UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

DESIGN OF RECOMMENDATION SYSTEM FOR RESTAURANT CLIENTS

Trabajo Fin de Máster

Máster Universitario en Ingeniería Informática

Autor: Waad Alkhoury

Tutor: Antonio Garrido Tejero

2021/2022

Acknowledgment

I'd want to express my gratitude to Antonio Garrido Tejero, my supervisor, for making this work possible. His recommendations, remarks and advise helped me get through all of the stage of the project.

I'd also like to express my gratitude to my family and friends for their unwavering support.

Abstract

In this thesis we discuss the process of developing a recommendation system. The recommendation system is based on state-of-the-art artificial intelligence and machine learning techniques. The system will recommend the clients of a restaurant to consume food according to their interests and taste of certain ingredients. The system will be able to decide their next preferable dish by considering many features, such as their previously consumed food/ingredients, time and date, age, gender, drinks, weather, etc.

The purpose of this system is to facilitate the order process for clients. The idea came of the fact that people take long time to decide and choose which food they are looking for to satisfy their needs, in fact there are no similar application that offer this feature in the market.

The first step that was to perform a research on the tools and logarithms to build an AI to recommend an item to a user. Firstly, different methods were analyzed to find the best one. Secondly, testing the methods to determine the most accurate prediction for a product. Finally, analyzing the results to establish the accuracy rate of each method that was tested.

As a result, the user builds up a profile with various information such as age, gender, nationality, food preference, food allergy and tolerance. Furthermore, each time the user orders a product in the restaurant it is stored into a database belong to the restaurant that will help with predicting and suggesting food for the next time the user stops by the restaurant.

Keywords: Collaborative filtering, filtering, content-based filtering, recommendation system

Contents

List of Figures

List of Tables

CHAPTER 1
Introduction

Recently we have been seeing some applications that allow the clients of a restaurant to order food and pay for it using a mobile application while sitting at the table. These applications, such as Watson[1], Sirveme[2], and Pikotea[3], have made it so easy for the restaurant owners to manage the incoming orders and serve the food faster. However, these systems still lack a recommendation system that can help the clients of any restaurant in trying out new dishes. Such systems will dramatically improve the sales of any restaurant and will make it easier for the clients to choose their next dishes.

Starting of this point the idea of the food recommendation system came of, the idea provides a fundamental value to the user's experience by facilitating their ordering process and picking their meals based on their profile, previous orders, and other factors.

Ordering food and stay up to date with latest meals by a restaurant that meets your profile became a problem for all parts of the process, either for the user himself or the servants that would explain new meals, and the contents of the new meal to the user that leads to delay them of other tasks.

The most important component of the project is to provide an accurate prediction of the user's next meal, which may be done by looking at the user's previous orders.

This project entails the creation of a recommendation system that complements other ordering applications and acts as a starting point of evaluating the concept and future enhancements. Therefore, the development is carried out by two main parts, firstly the prediction of next meals to the users, and secondly the feedback that was provided by the users to measure the accuracy of the system.

1.1 Motivation

This chapter justifies the various motivational factors that were used to complete this project, both personal and technological. As well as a boost career opportunity for the future since it's a field that well required for upcoming future projects.

- Analyze existing recommendation systems and adding new feature to the existing systems. The main idea to improve an existing method of predicting product, by finding where other systems fail and cover up those weakness points in other systems as well adding new feature for POS applications.

- Learn Machine learning technologies. Due to the demands on this technology in many aspects of life, this project would open new research branches and new professional opportunities.

- Improve knowledge of programming languages. Improving in one of the most known and powerful languages on our day time " Python" and using it on a larger scale. This project would bring experience and opportunities for future projects.

- Learning about different libraries in python. By including some libraries to project that required a deep study for the most suitable way to get it done, and how wide and deep it is.

- Use the acquired knowledge learned in practice. Starting a project within the machine learning field and the complexity that it holds without any prior experience acquired a lot of research and studies to analyze, design and implement.

These listed points above explain the reasons that carried the project and establish the beginning of the objectives of it.

1.2 Objectives

The major purpose of this project is to build a food recommendation system, and the following sub-objectives must be completed to complete this implementation and meet the project's objectives:

- Build a database based on a real data by restaurants and users.
- Analyze the available recommendation systems and compare their strengths and weaknesses.
- The research about all the methods that are used in recommendation system applications.
- Perform tests between all the different methods.
- Implement the best method and improve it.
- Carry out tests based on the data we have.
- Receive feedback by users and compare the predictions results with user's expectations.

These objectives are in charge of steering the entire process of completing the project, as well as achieving the desired results and establishing the project's starting points.

1.3 Methodology

The motivation for this project stemmed from the necessity for a function in numerous food ordering apps, therefore the goals were apparent. The project is focused on offering the most precious next meal for the user. For this the project has various phases to be done in sequence in order to establish and implement a stable and reliable application with possibility of repetition on any phase.

The progress of this project is dependent on the results and the implementation approach chosen, despite the fact that it includes a research component. Finally, only one person was in charge of its advancement.

With all these considerations, the optimum methodology for completing the project is determined to be an Agile methodology[4], this method Allows to identify main object and sub objective to be done in sprint cycles within a specific timeline to reach the final product that provide this feature.

Agile development is an iterative procedure that is mainly characterized to provide high-quality software and a business approach align with the customer needs.

One of the Agile's subset is Scrum which is a lightweight framework, that means it runs into cycles which called "sprints" as the figure below figure 1.1 shows, that aim to keep the tasks as small as possible. Scrum usually used to manage complex software using iterative and incremental practices. Scrum process adjust to changes smoothly and rapidly.

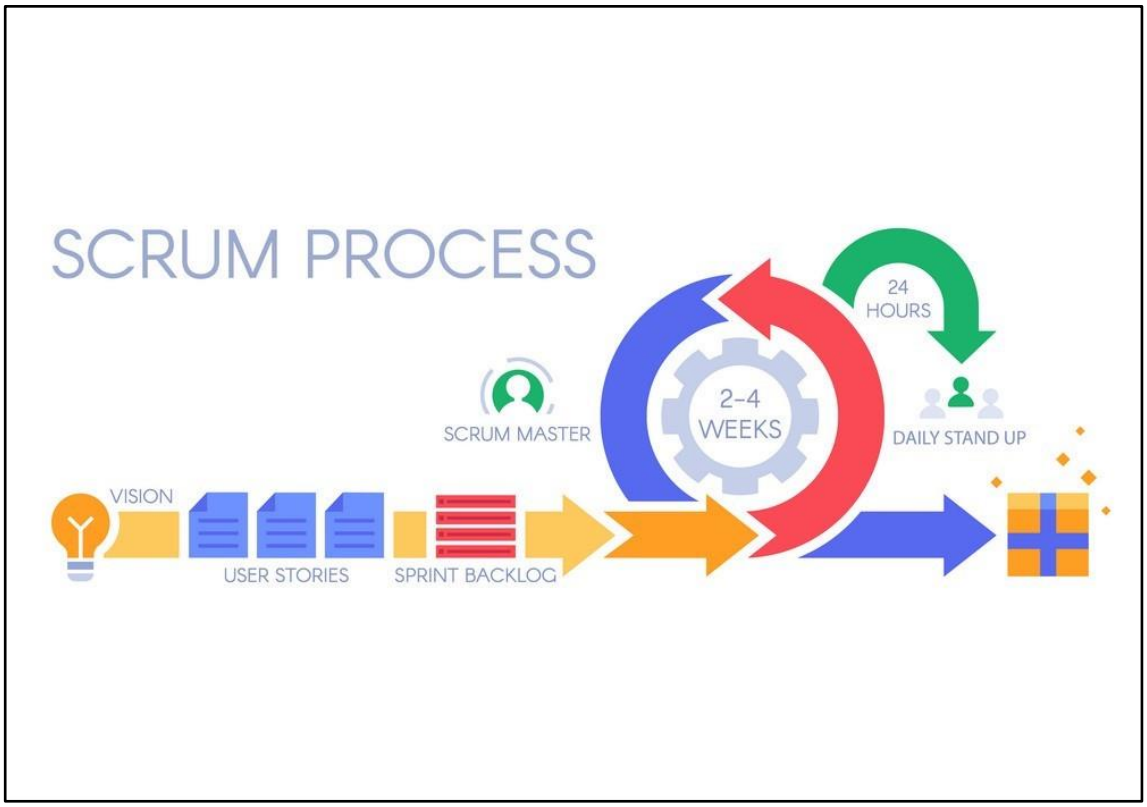


Figure 1.1: Phases of the scrum methodology

A final validation of the client is required during the development phase of this project, which increases the risk of updating functions. As a result, an evaluation and feedback from the user is required when reviewing the system functioning. Having the requirements stated from a project basis, on the other hand, minimizes the likelihood of errors being discovered

Furthermore, the phase of implementing and designing should be carried out, but since a research part consists of this project its slight modified. Research into recommendation system approaches necessitates tests to examine the outcomes, followed by a collaborative study, development, and testing stage. It is possible to evaluate the recommendation system by performing this in cyclical manner.

Furthermore, the development and testing phase are continued, with both phases being mixed in an intermediate stage for improved system quality and the ability to handle faults faster and more effectively.

1.4 Memory structure

The following chapters make up this article:

1. **Introduction:** This chapter outlines the work's overall backdrop, and the motive for its production and eventual goal.

2. **State of the art:** This chapter summarizes the existing technological condition in relation to the concept to be developed, assessing, and critiquing the possibilities already on the market.

3. **Proposal:** In order to determine the project's knowledge and technology area, this chapter examines the system requirements, along with the conceptual models and their use cases.

4. **Proposed solution:** The suggested solution, its development phase, the intricacies of its architecture, and how the system will be built and validated are all shown in this chapter.

5. **Implementation:** This chapter goes through the system project implementation, including the tests that were run and the results that were obtained.

6. **Conclusion:** Is the final chapter dedicated to determining whether the objectives were reached and discussing any issues that arose during the development process and offering future work opportunities.

CHAPTER 2
State of the art

Because of the rapid advancement of technologies in the ordering process, the restaurants has had to adapt substantially as a result, forcing a huge portion of the population to be registered on these types of applications[5], as shown in Figure 2.1

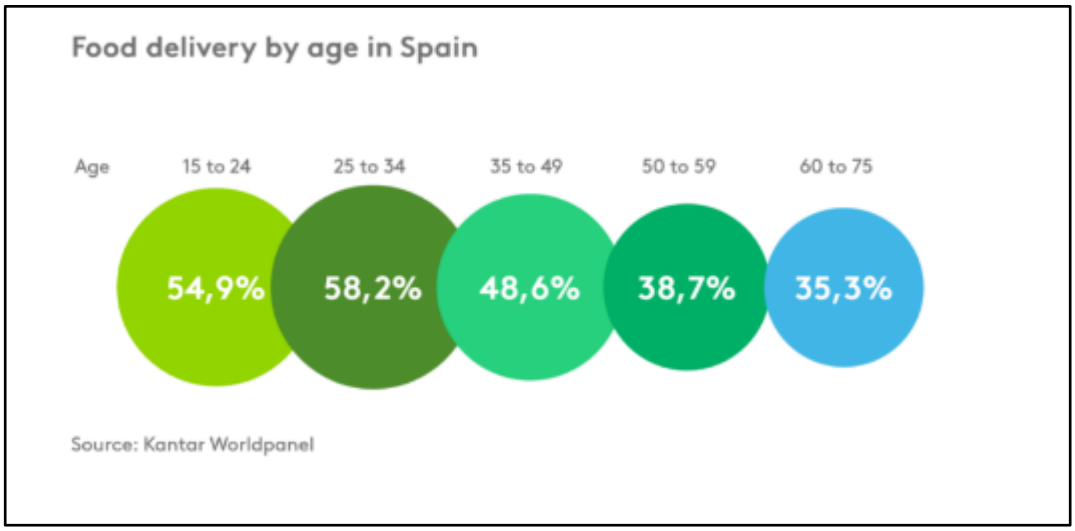


Figure 2.1: rate of Spanish population that use food delivery applications

The reason behind this, in recent years, the high number of companies that provide this service as you can see in the figure 2.2. These companies also add many offers due to the high competition between all delivery companies that would encourage users to order food. In addition to the variety of restaurants that are spreading fast world wide, Which lead to motivate the user to try different foods from different countries regardless of his incapability of cooking or due to his busy schedule that would be a huge obstacle for the user to cook.

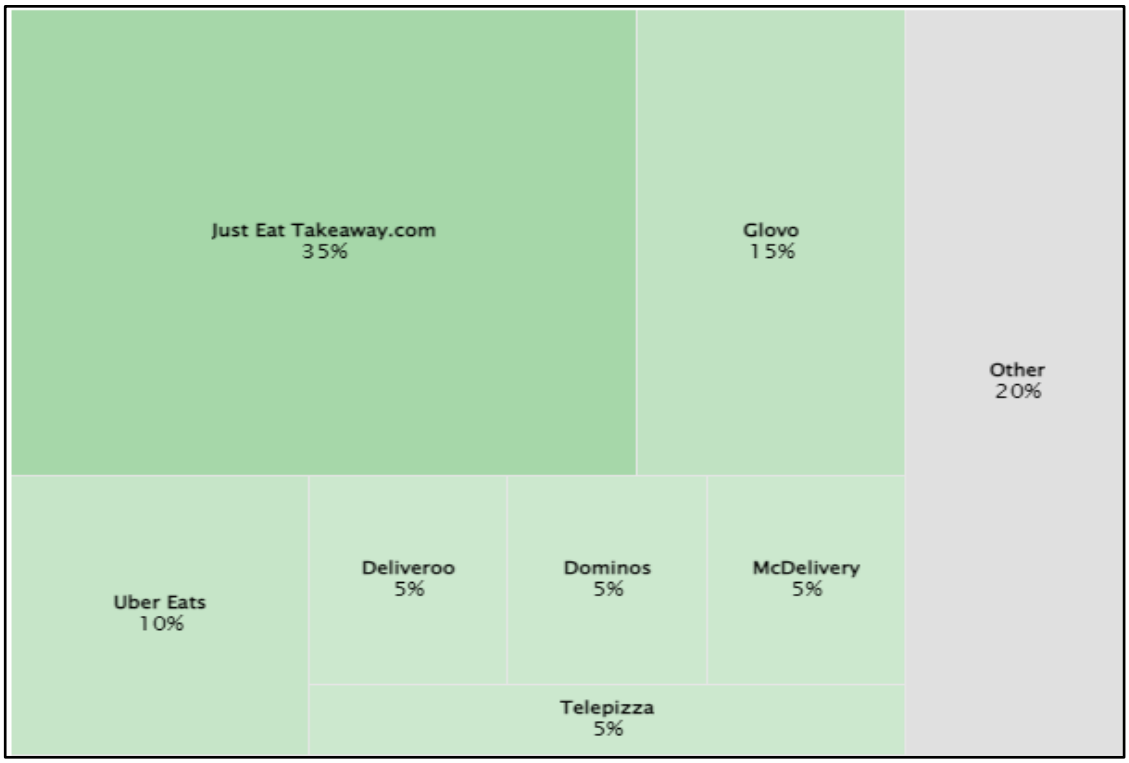


Figure 2.2: Percent of users that order food of food delivering companies

Along with these figures, another significant factor to consider is the increase in Spain's monthly minimum wage. Figure 2.3 depicts the rise in the monthly minimum wage over the last few years.

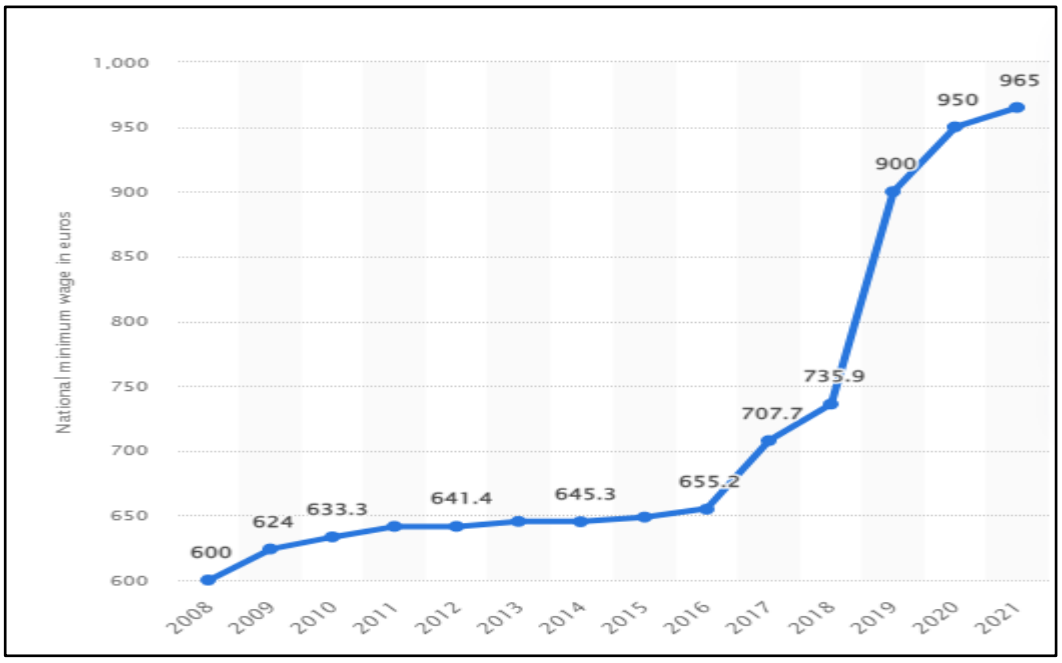


Figure 2.3: National minimum wages in euros in Spain

If we analyze these data together with the graph of number of companies that deliver food, we observe that the increase in the number of delivery food companies and the increase of the wages lead to the increase in the number of people that use online ordering food applications. Also, As demonstrated in Figure 2.4, the primary motivation for consumers ordering food is to save time.

Another aspect that are many channels on different platforms, such as YouTube, contains specific categories only for food that would make people curious and in passion to try out different cultures food that they can see throughout the "influencers", by ordering them from their original restaurants in order to try out the original taste of the food.

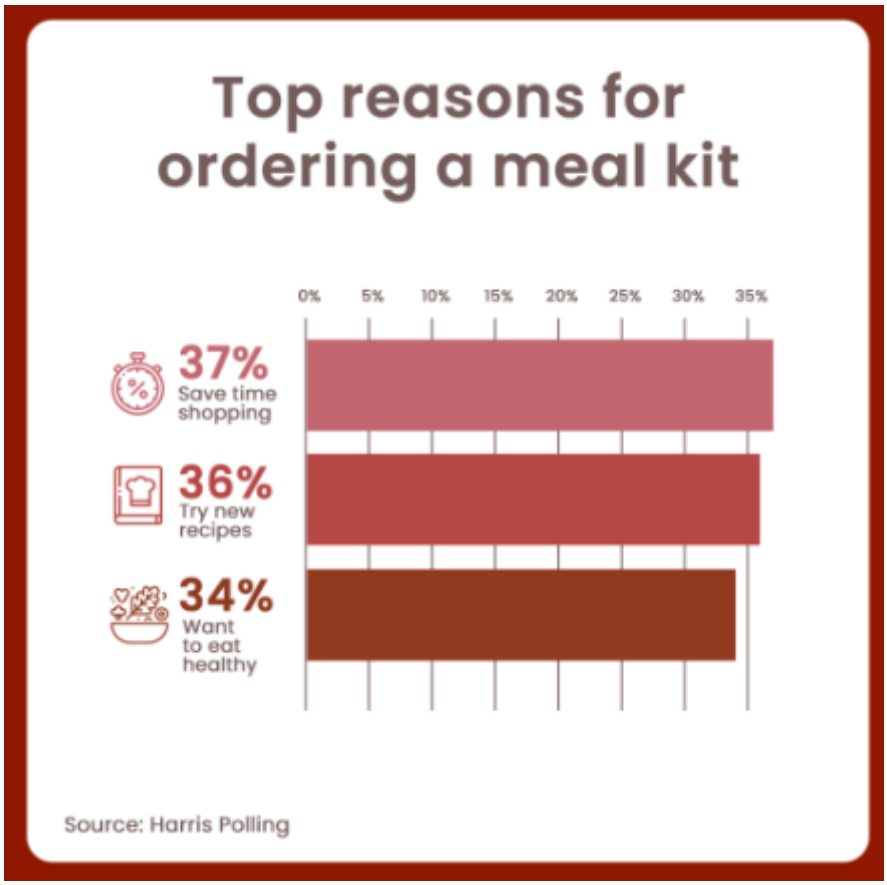


Figure 2.4: Top reasons for ordering a meal

All these variants of foods have led to develop new methods of recommendation systems. Specially with the increased number in the delivery companies and the offers that are made between restaurants, and the delivery companies due to the high competition between each other, encouraged the user to use these applications and to be benefited by the feature of recommending food to try out different food from different cultures that would meet their profiles.

2.1 Existing solutions

In general, we can distinguish between all existing food recommendation system solutions for this project.

2.1.1. Watson

An application[1] that exists for mobile and web contain multiple features for ordering food, either by ordering from home or by sitting on the table of the restaurants. The application requires to be a registered user to order your food, and only ask for the location where food should be delivered.

We can highlight the filtering methods:

1. **Categories**: You can specify category to look through based on your need.
2. **Distance**: You can search by the closest restaurant to your location.
3. **Recent product**: You can find the most recent products that have been added to the application.
4. **Letters**: You can filter the products based on their alphabetical.
5. **Order time**: You can search for products based on their preparation time.
6. **Delivery time:** You can search for products based on the time it takes to be delivered.

Figure 2.5: Watson app

2.1.2. Sirveme online

An application[2] that is only for mobiles either Android or IOS. The application is divided into 2 parts, one for users and other for restaurants. Restaurant applications allow restaurants to add their menu as an online menu. And for users it requires them to be registered in order to make their orders, either from their location or at the restaurant.

We can highlight the features for user application:

1. **Delivery:** User can order by the application and it will be delivered.
2. **Take Away:** User can order by the application and you have to pick it up.
3. **Check the menu:** User able to check the menu of the restaurant from the application itself.
4. **Call the waiter:** A feature where you can call the waiter from the application.
5. **Pay by the application:** User able to pay by the application without the need of waiter interactive.

Figure 2.6: Sirveme Online app

2.1.3. Pikotea Go

An application[3] that is only for mobiles either Android or IOS. The application is divided into 2 parts, one for users and other for restaurants. Restaurant app allows restaurants to add their menu as an online menu. And for users it requires them to be registered and insert their postal code in order to find close restaurants in their area.

We can highlight the features for user application:

1. **Categories**: User can sort restaurants by categories.

2. **Check the menu**: User able to check the menu of restaurants online.

3. **Allergies**: User can add specific allergies to filter out the food within this list.

4. **QR scan**: User can scan the QR code of the restaurant to check out the menu.

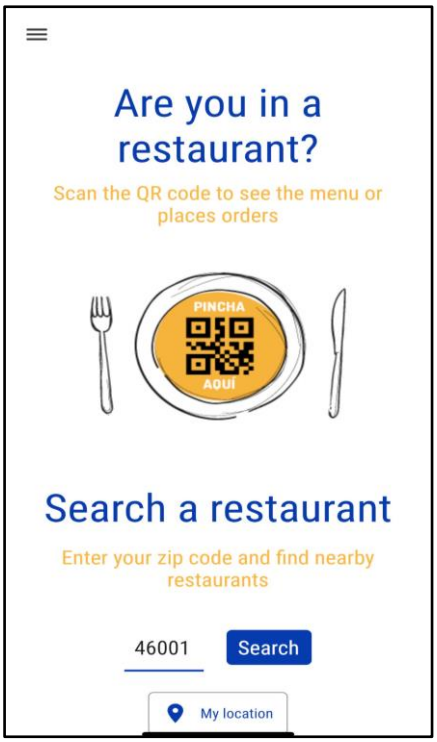


Figure 2.7: Pikotea Go app

2.1.4. Future ordering

An application[6] that's available for mobile and web, contains multiple features for ordering food, either by ordering from home or by sitting on the table of the restaurants. The application requires users to be registered in order to take action into the application.

We can highlight the features for user application:

1. **Restaurant menu**: User can check out the restaurant menu.

2. **Promise time**: Calculate the estimated time for delivering food either for your location or table.

3. **Food statue**: User will be up to date about the statue of his food.

4. **Gift cards**: User can use gift cards to order their food.

5. **Offers**: User can checkout restaurant offers.

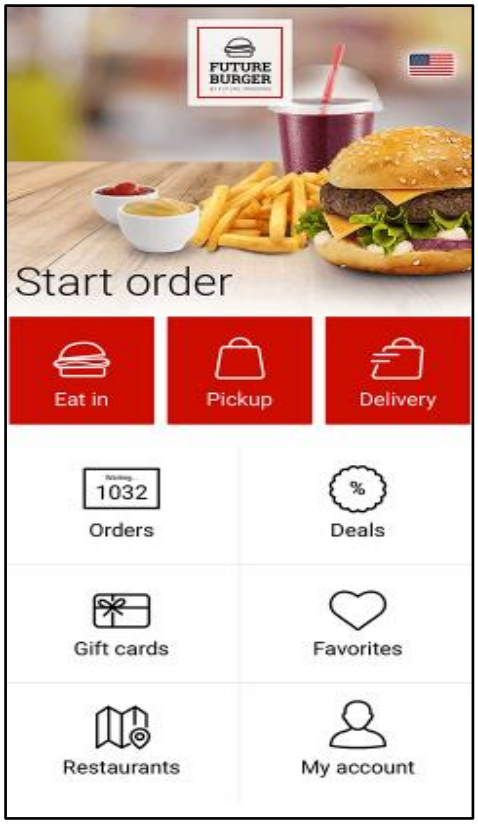


Figure 2.8: Future Ordering app

2.1.5. Brainium

An application[7] that's available for mobile and web, contains multiple features for ordering food, either by ordering from home or by sitting on the table of the restaurants. The application requires users to be registered in order to take action into the application.

We can highlight the features for user application:

1. **Categories**: User can sort out by cuisine, average costing, offers, popularity, food type.

2. **Order statue**: User can track out their orders in real time and current location.

3. **In-app Payment**: User can pay by the application.

4. **Offers**: User can check out the offers that are provided by the restaurant.

5. **Favorite**: User can add their favorite meals to liked list.

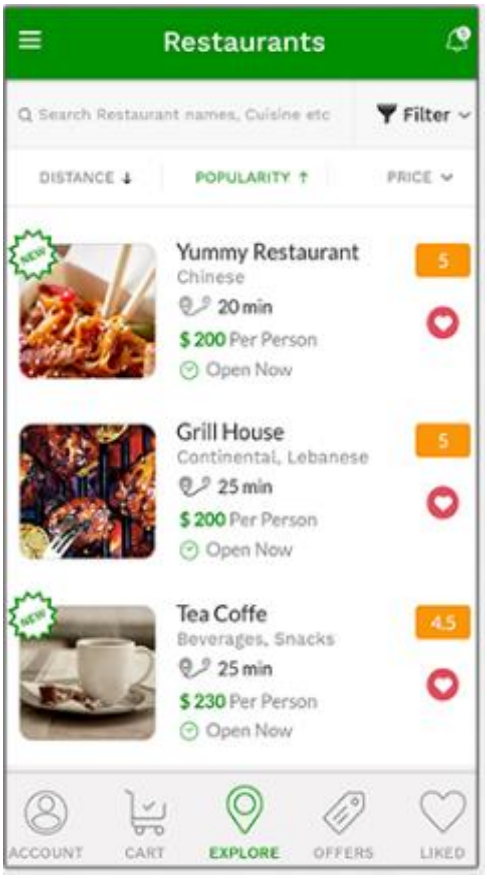


Figure 2.9: Brainium app

2.2 Analyze the existing solutions critically

Criticizing the applications that have been mentioned in the 2.1 subsection would focus on the missing features of each of these applications and analyze the existing features of each application.

2.2.1 Watson

Watson app available as web application and mobile application, which is a positive point for this application, which also contain a lot of filtering and sorting to find what you are looking for of the products. In other hand for restaurants owners its expensive 50€ to integrate some features like ordering from the table. Finally, its missing out some features such as, taking allergies and food preferences of clients in count, as well as recommendation system.

2.2.2 Sirveme online

Sirveme online app available as web application and mobile application either for IOS or android which is a positive point for the application. Also, the application has some features that would be counted as positive points such as, paying from the application, calling the waiter using the application and shows the menu of the restaurants in a digital way. in another hand the application does not have any kind of filtering or sorting out for the products of the menu, which as well missing the recommendation system of the products for the clients.

2.2.3 Pikotea Go

Pikotea go app available only for mobile application either IOS or android which is a negative point compared to other applications were analyzed in the subsections 2.2.1 and 2.2.2. Moreover, the application is expensive for simple package which goes around 70€. In other hand some features can be counted as positives points such as, checkout out the menu, being able to filter out food with specific allergies and QR scan the QR code on the restaurant tables.

2.2.4 Future ordering

Future ordering app available on both mobile and web application, which is a positive point, moreover it has some other features that can be positively counted for the application such as, gift cards, offers made by restaurants, track up the food statue and show the restaurant menu digitally. However, it’s missing some other features such as filtering out food preferences or allergies or recommend some product for the clients based on their profiles.

2.2.5 Brainium

Brainium app available on both mobile and web application, also the user can do some other actions such as, sorting out products based on their categories, pay in application, track order statue, add products to their favorite list and check out offers of restaurants. However, its missing out filtering by food preferences and allergies as well as missing out system recommendation.

In summary of the section 2.2 the table below shows the advantages and the disadvantages of each existing solution.

Application	Advantages	Disadvantages
Watson	➢ Developed for both mobile application and web application. ➢ Multiple options to filter by.	➢ Expensive ➢ Can't filter based on allergies, or preferences ➢ Missing recommendation system
Sirveme online	➢ Developed for both mobile application and web application ➢ Check the menu digitally ➢ Pay by the application ➢ Call the waiter by the application	➢ Can't filter products ➢ Missing recommendation system
Pikotea Go	➢ QR scanner implemented into the application ➢ Filter out by categories ➢ Add allergies to filter products out ➢ Check the menu digitally	➢ Only available as mobile application ➢ Expensive ➢ Missing recommendation system
Future ordering	➢ Developed for both mobile application and web application ➢ Able to give gift cards ➢ Check offers ➢ Check food statue and estimated time for delivery	➢ Can't filter products based on allergies or preferences ➢ Missing recommendation system
Brainium	➢ Developed for both mobile and web application ➢ Sorting by categories ➢ Check order status ➢ Save favorite meals ➢ Pay by the application	➢ Can't filter products based on allergies or preferences ➢ Missing recommendation system

Table 2.1: Comparison between the existing solutions

As an analysis to the table 2.1, most applications of food delivery are missing recommendation systems and don't take the user allergy or preferences in count. Therefor a recommendation system into an application will bring many benefits for the user and the restaurant, either by make the process and making decision for the user easier or increasing the revenue for the restaurant.

2.3 Recommendation system available methods

2.3.1 Graph neural network

Graph Neural Networks[8] are a type of deep learning algorithms that can be applied to graphs and give an easy way to conduct node-level, edge-level, and graph-level predictions tasks. GNN's primary principle is to get a representation for a node by recursively aggregating the representations of surrounding nodes to specified depth[9].

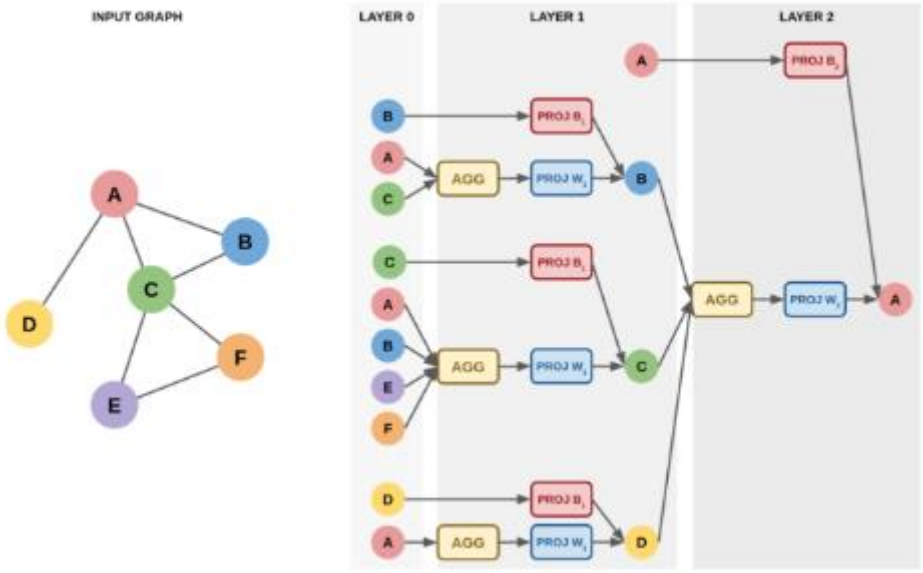


Figure 2.10: GNN obtain representation node of an input graph

Starting with A, take the average and projection a matrix multiplication with a learned weight matrix to get a representation of a neighborhood of the node at one hope distance from A, the neighborhood is combined of information about the node itself and projected by matrix multiplications with a learned weight matrix B, this happens recursively to capture both properties of node A and information about the neighborhood. For example UberEats[9] employ graph neural network for their recommendation system.

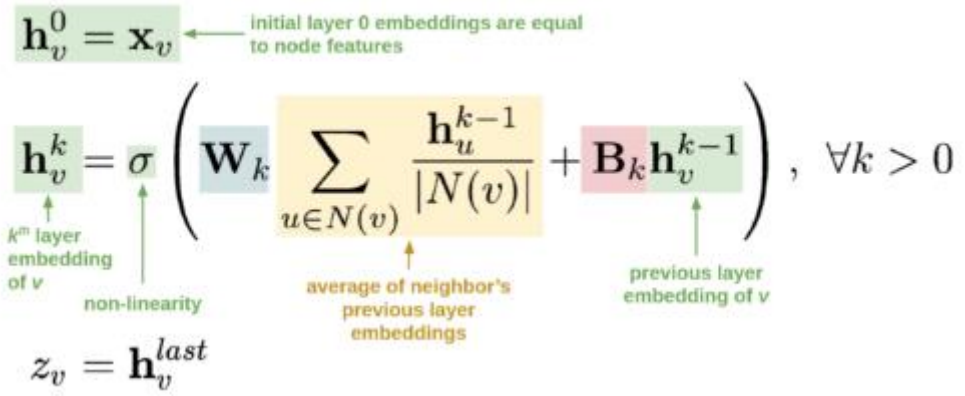


Figure 2.11: Equation represent the computation graph in figure 2.10

2.3.2 Content-Based filtering

The system tries to figure out how similar objects features are. A content-based recommendation system creates a user profile based on the user's previously rated things, allowing him to adapt to new interests by matching the user profile with content attributes. Very accurate user profile is needed to give best results[11]. For example, one of the most online dating applications use this method which is Tinder.

Steps that the recommendation goes through to process:

1. Content Analyzer: Because some data is unstructured, the content analyzer reflects the information from the source in a meaningful way.

2. Learner: This module creates a user profile by combining the data from the previous component.

3. Filtering Component: This module predicts similar things based on the user's profile.

Disadvantages of the content-based filtering:

1. Over-specializations: It doesn't have essential method to predict thing unpredicted, it will look through high score rated items that are related to user profile.

2. New user: Content-based recommendation will not work for new user since it requires for some data to be collected of the user to build up his profile.

2.3.3 Collaborative filtering

For recommendations, collaborative filtering uses “user behavior”, which means there is no feature related to users or things. It makes use of a matrix. The content-based recommendations system’s shortcoming can be overcome since collaborative filtering predicts items to recommend based on the feedback of other users. Peer reviews are used to evaluate items. Also, it suggests other items if the user have shown interest in them[11]. Many famous applications use this method such as Amazon and Netflix.

Collaborative filtering can be divided into two types:

- **2.3.3.1 Memory-based approach**

  The utility matrix here is learnt and suggestions are given by asking the given user with the rest of the utility matrix.

$$\bar{y}_i = \frac{1}{|I_i|} \sum_{j \in I_i} y_{ij}$$

This equation will yield the customer’s average ranking for all the products, whereas the product rating can be computed using this equation.

$$\hat{y}_{ik} = \bar{y}_i + \frac{1}{\sum_{a \in U_k} |w_{ia}|} \sum_{a \in U_k} w_{ia} (y_{ak} - \bar{y}_a)$$

regardless of if the data becomes spares, we will have poor performance, however, to calculate the similarities we can use methods like:

- Cosine similarity:

  Cosine similarity is the cosine of the angel between 2 dimensional vectors in number of dimensional spaces. It’s the product of the two vector’s lengths divided by the dot product of their lengths[10].

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Figure 2.12 Cosine similarity formula

- Pearson's correlation:

The strength of a linear link between two variables is measured by pearson's correlation. That has the value between -1 and 1, -1 means it's a total of negative linear correlations, while 0 means no correlation and +1 means it's a total positive of linear correlations[12].

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

Figure 2.13 Pearson's correlation formula

Furthermore, memory-based collaborative filtering is classified into two groups, which we will compare on a few topics below.

1. **2.3.3.1.1 Item-based filtering:**
    a. Accuracy: Analyze the scores given by the same user to determine the similarity between two items.
    b. Efficiency: When there are more users than items, item-based filtering is more reliable in terms of memory and time necessary to calculate the similarity than user-based filtering.
    c. Stability: Item-based filtering has more stability in case the items are static because the similarity weight of items be be compute irregular time intervals.
    d. Justifiably: The reason for the recommendation can be explained by looking at the list of neighboring items and their similarity weights.
    e. Serendipity: The item-based filter has a drawback in this situation because it only recommends goods that are comparable to those that the user has previously enjoyed.

2. **2.3.3.1.2 User-based filtering**
    a. Accuracy: User's similarity is determined by examining the scores given to the same item by different users.
    b. Efficiency: When there are more users than objects, user-based filtering is less accurate in terms of memory and time necessary to calculate the similarity.
    c. User-based filtering has more stability in case the items are dynamic and change because the similarity weight of items be computed irregular time intervals.
    d. Justifiably: Because the user is unaware of other people's preferences, the user-based filter can't explain why the recommendation is produced.
    e. Serendipity: The algorithm provides surprising recommendation by studying the neighbors who gave the same rating as the user and checking the ratings on different things by the neighbor user.

Memory-based collaborative filtering has the following advantages:

- **Easiness**: Neighborhood-based strategies are intuitive and simple to execute.
- **Justifiably**: It offers explanations for the calculated results.
- **Efficiency**: It does not necessitate costly training phases, and memory usage is minimal.
- **Stability**: The constant insertion of the consumers has a minor impact on it.

- **2.3.3.2 Model-based approach**

  This model-based collaborative filtering use machine learning to find user ratings of unrated items. However, this approach can deal with missing sparse unlike Memory based collaborative filtering as discussed in subsection 2.3.3.1 which can be a positive point for this approach. However, it has some other disadvantages like memory consumption is high for training data and the inference is intractable because of the hidden latent.

2.4 Technological context

This chapter describes and explains the technologies utilized in this project, including their benefits and drawbacks, as well as why they were chosen.

2.4.1 Python

Python[13] is a general-purpose, adaptable, and powerful programming language that is also quite easy to read. For being object oriented supportive, imperative programming, and functional programming. In summary, it may be utilized for a variety of tasks like web development, machine learning, and data research. This language has been chosen over other programming languages for some reasons:

- **Community**
  Python is known worldwide with huge community[14] which provide a value when looking for solution for some language problems.

- **Powerful and flexible**
  Python is powerful and flexible since it wasn't created for specific need therefore it can be used for whatever can be done.

- **Available libraries**
  Python offers a large library repository that can be utilized specifically for machine learning tasks.

- **Documentation**
  Python is known worldwide therefore it is required to have a well written documentation for all libraries that it contains.

- **Previous knowledge**.
  As a programmer that I already have experience in work, I have been using python as a programming language for several projects, which I feel comfortable developing with it.

2.4.2 Snowflake

Snowflake[15], is a data platform and data warehouse, it supports standardized version of SQL. The main advantages of snowflake warehouse the performance and speed, durability and reliability of the data, and the high flexibility that the user can use the warehouse and query services in the same data lake.

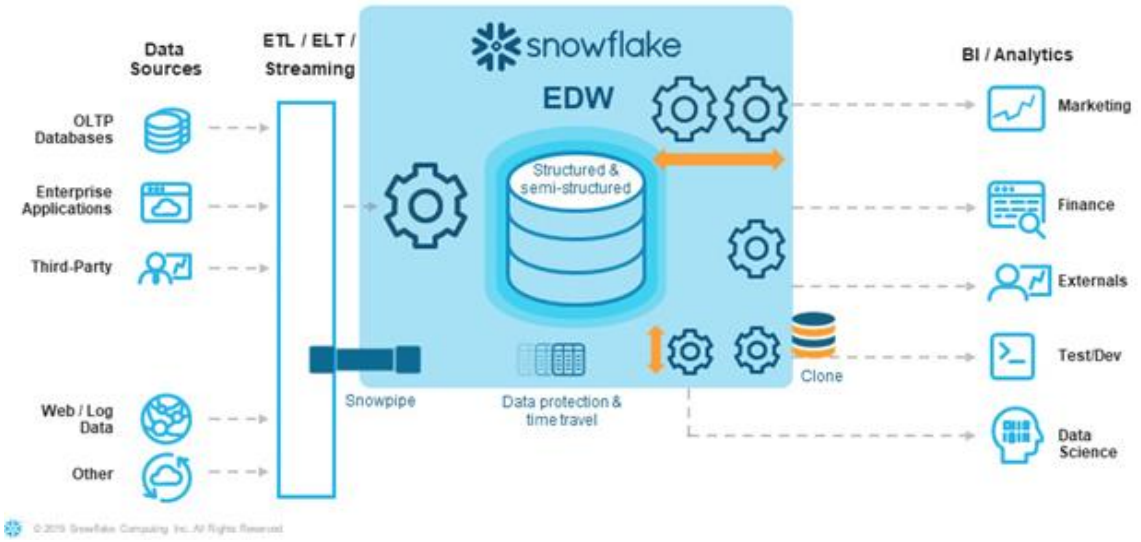


Figure 2.14: Snowflake platform and data warehouse

However, the experiments I am doing using csv files, but I would recommend using snowflake for the system in production for the mentioned advantages above in subsection 2.4.2.

2.4.3 Git

Git[16] is a free and open source version control system created by Linus Torvalds. It can manage projects of any size, from tiny to huge, and can assist in the versioning of many sources. The following are properties of Git:

- Nonlinear development
  Git allows users to perform operations from all over the world remotely on a project, by creating branches of the project.

- Reliable
  It provides a central repository that is cloned each time the user makes a pull operation and the central repository is always backed up in every user local repository.

- Secure
  it keeps a record of commits each user has made on the local copy of the developer.

- Repository can be published by using HTTP, SSH or TCP/IP protocols.

Git was used to manage different methods of recommendation systems, having a repository for each of them: Content-based filtering and Collaborative filtering. The figure below shows the GitFlow methodology[17].

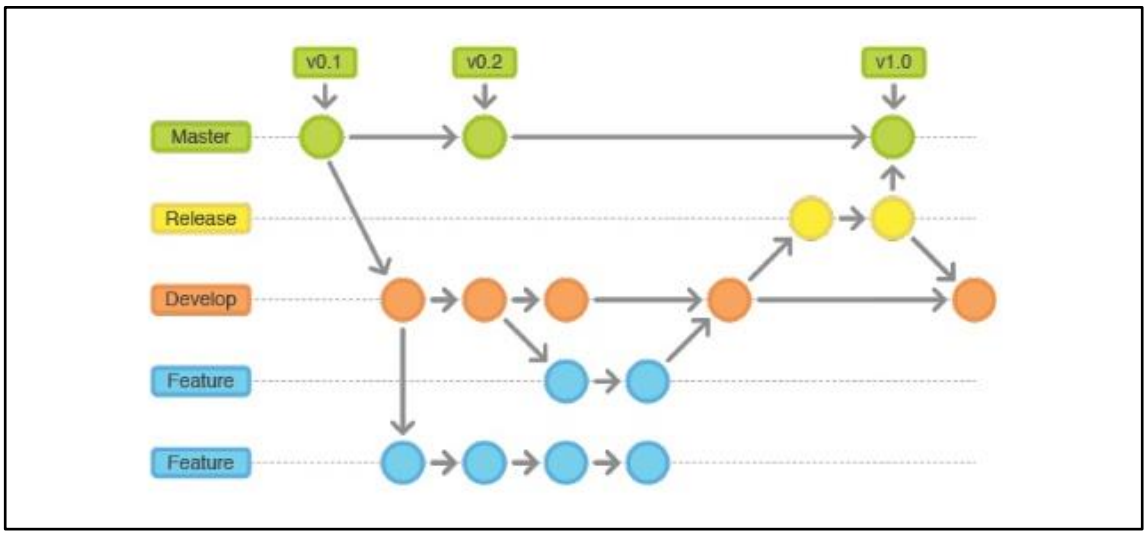


Figure 2.15: Gitflow schema

Moreover, Github is the most popular web platform for developers and programmers to collaborate on projects. It is used for managing repositories and includes features like CI/CD, pull request management, and code reviews.

CHAPTER 3
Proposal

The main goal of this project is to create a food recommendation system, specifically for restaurants, as opposed to other industries such as movies and online shopping.

There are numerous methods for recommending items to consumers, as described in Chapter 2. Some of these methods, such as the one described in the subsection 2.3.2, recommend based on the user's profile, which necessitates collecting data about the user in order to process the recommendation system, however, the more accurate data collected about the user, the more accurate the system becomes. In this case, improving the recommending system is an objective.

Other methods, such as collaborative filtering described in section 2.3.3, are based on item ratings and other user's ratings to forecast the next item for the user, whereas content-based methods focus on the user profile to predict the next item for the user.

This system is proposing the recommender system using hybrid filtering to recommend users based on criteria such as ratings, user age, user gender, food description, food origin country, food allergy type, food preparation time, food ingredients, and food type. This system will recommend food based on the user's preferences and allergies.

The rating is provided on a separate scale by the user, the system uses these ratings for analysis and recommendation of foods. The purpose of one part of the system is to estimate the rating of the food for a customer where he hasn't rated the item before.

Whenever a customer wants to order a specific type of food, such system will recommend him the right food, it will assist the customer select the food taking in concern price, discount, and taste. This system will also safe effort and time to find the best food that meet the user profile. The application of such a system will help to create loyal customers and make them attend the restaurant more often. As a result, the restaurant will appear competent in the market. Recommendation system will play a great role play in increasing revenues[18].

3.1 User definition

The users are the people who interact with the system. It is vital to define the users who will interact with the food recommendation system in order to analyze it.

3.1.1 New user

This user must create their profile first to be able to perform some analyzing and build over their profile.

3.1.2 Registered user

This user has previously generated a profile in the system, allowing the system to assess their profile and generate a recommendation for them.

3.2 Analysis of requirements

The specification that a recommendation system must meet in order for it to forecast and recommend products.

Because the recommendation system is build on a the accuracy of the requirements specification, it must be accurate.

3.2.1 Functional requirements

A software function is defined by a functional need. A set of inputs, behaviors, and output is referred to as a function. These are the kinds of requirements that a recommendation system would examine and evaluate. Below is a list of the functional criteria.

New user

Code	FR-01
Dependencies Name	Create profile
Description	The application must allow the user to create their profile
Restrictions Importance	Medium

Table 3.1: Functional requirement FT-01 Create profile

Registered users

Code	FR-02
Dependencies Name	Display main menu
Description	The user must be able to see the primary menu of the application.
Restrictions Importance	Medium

Table 3.2: Functional requirement FT-02 display main menu

Code	FR-03
Dependencies Name	Display Food list
Description	The user must be able to see the food list in the application.
Restrictions Importance	Medium

Table 3.3: Functional requirement FT-03 display food list

Code	FR-04
Dependencies Name	Display recommended food
Description	The user must be able to see the recommended food on the application.
Restrictions Importance	Medium

Table 3.4: Functional requirement FT-04 Recommended food

Code	FR-05
Dependencies Name	Display ordered item history
Description	The user must be able to see the history of orders in the application.
Restrictions Importance	low

Table 3.5: Functional requirement FT-05 display ordered item history

3.2.2 Not Functional requirements

These requirements are characteristics that is used to assess a system's functioning rather than its specific behaviors, they can be related to issues like accuracy. Below is a list of non-functional requirements

Recommendation system methods

- **NFR-01:** The data filled should be correct and accurate.
- **NFO-02:** User must be created, and profile must be filled.
- **NFO-03:** Rating should be positive.
- **NFO-04:** To consider product for rating it should have minimum amount of voting by users.

Database

- **NFR-05:** Snowflake must be used to implement the database.

3.2.3 Business rules

These are not functional requirements that the system must follow in order to meet quality standards, such as behaviors and rules. The business rules for this system are listed below.

- **BR-01:** Many products are related to many restaurants.
- **BR-02:** Many products are related to many Orders.
- **BR-03:** Many orders are related to one user.
- **BR-04:** Many allergies are related to many users.
- **BR-05:** Many product types are related to many products.
- **BR-06:** Many product types are related to many users.
- **BR-07:** Many products are related to one country.
- **BR-08:** Many ingredients are related to many products.

3.2.4 Data requirement

These are a type of not functional requirement that specifies the information that will be used in the system. The system's information requirements are stated below.

- **DR-01**: For each user will be stored:

    - User ID
    - Name
    - Gender
    - Age
    - Nationality
    - Allergies
    - Preferences

- **DR-02**: For each product will be stored:
    - Product ID
    - Name
    - Allergy type
    - Origin country
    - Preparation time
    - Discount
    - Ingredients
    - Description
    - Vote average
    - Vote count
    - Type

- **DR-03**: For each order will be stored:
    - Order ID
    - User ID
    - Product ID
    - Day
    - Weather
    - Time

3.3 Use cases

We proceed with the graphical case of the system to finish the functional requirements specification provided in the subsection 3.2.1. The interplay of the entities that interfere in the system is the emphasis of this illustration.

Actors: They are the elements in the system that perform the activities. They represent function rather than a specific item.

Use cases: They are the system's functionalities.

Relationships: They depict the relationships between Actors and Use cases, as well as Use cases within Use cases.

System boundary: It establishes the scope of the system's operation.

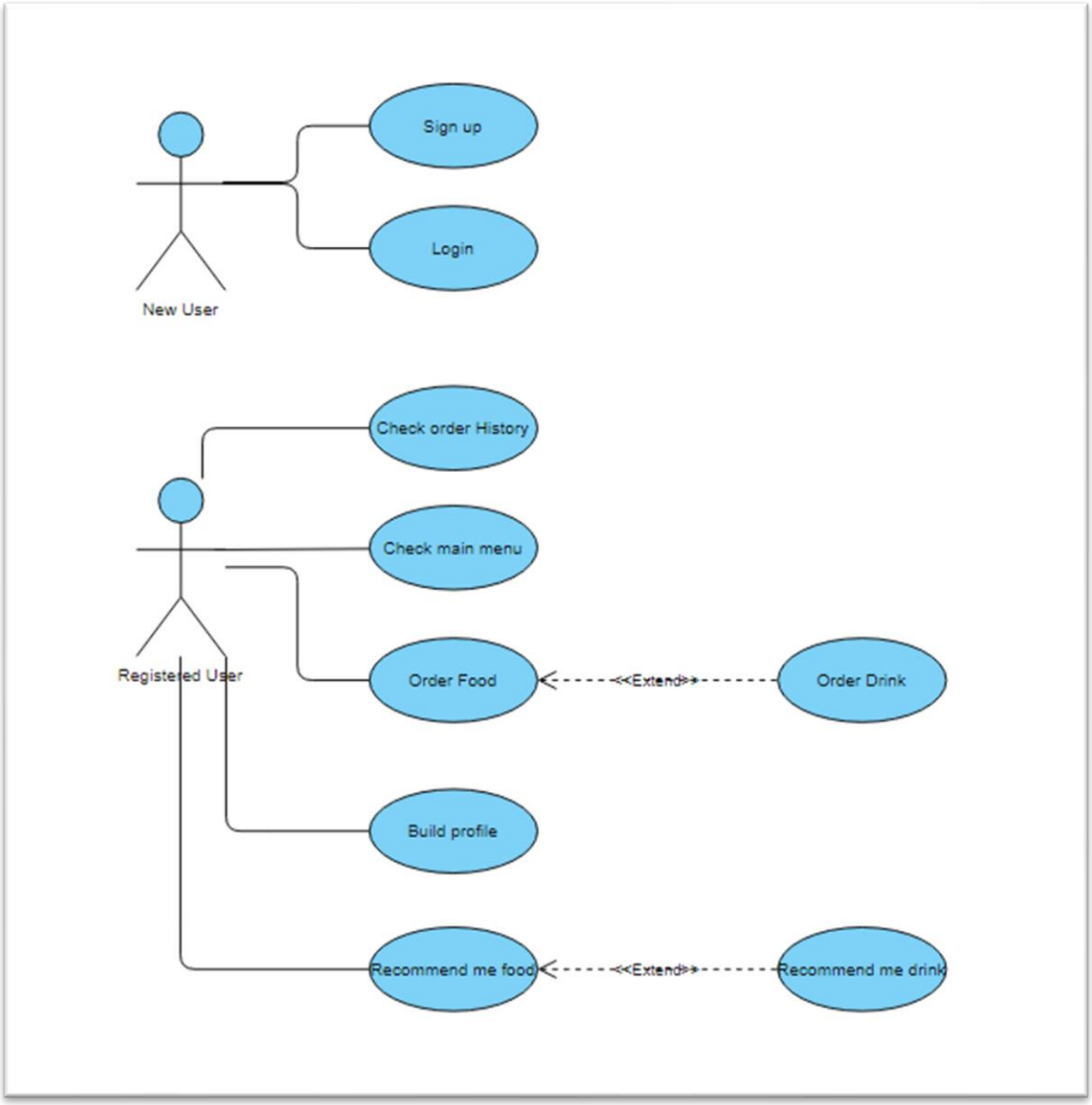


Figure 3.1: Food Recommendation system use case graph

In the diagram shown in the Figure 3.1, there are two actors: new users and registered users. For an application with a food recommendation system, it explains the activities that each Actor can take.

In this case there are no relations between the use cases and that for the reason the user is not restricted with following steps to have a food recommended for him.

3.4 Conceptual modeling

The purpose of a conceptual model is to depict the system's essential components and their relationships. This has been taken from the data requirements and business rules in subsections 3.2.3 and 3.2.4, and is depicted in figure 3.2 as a diagram.

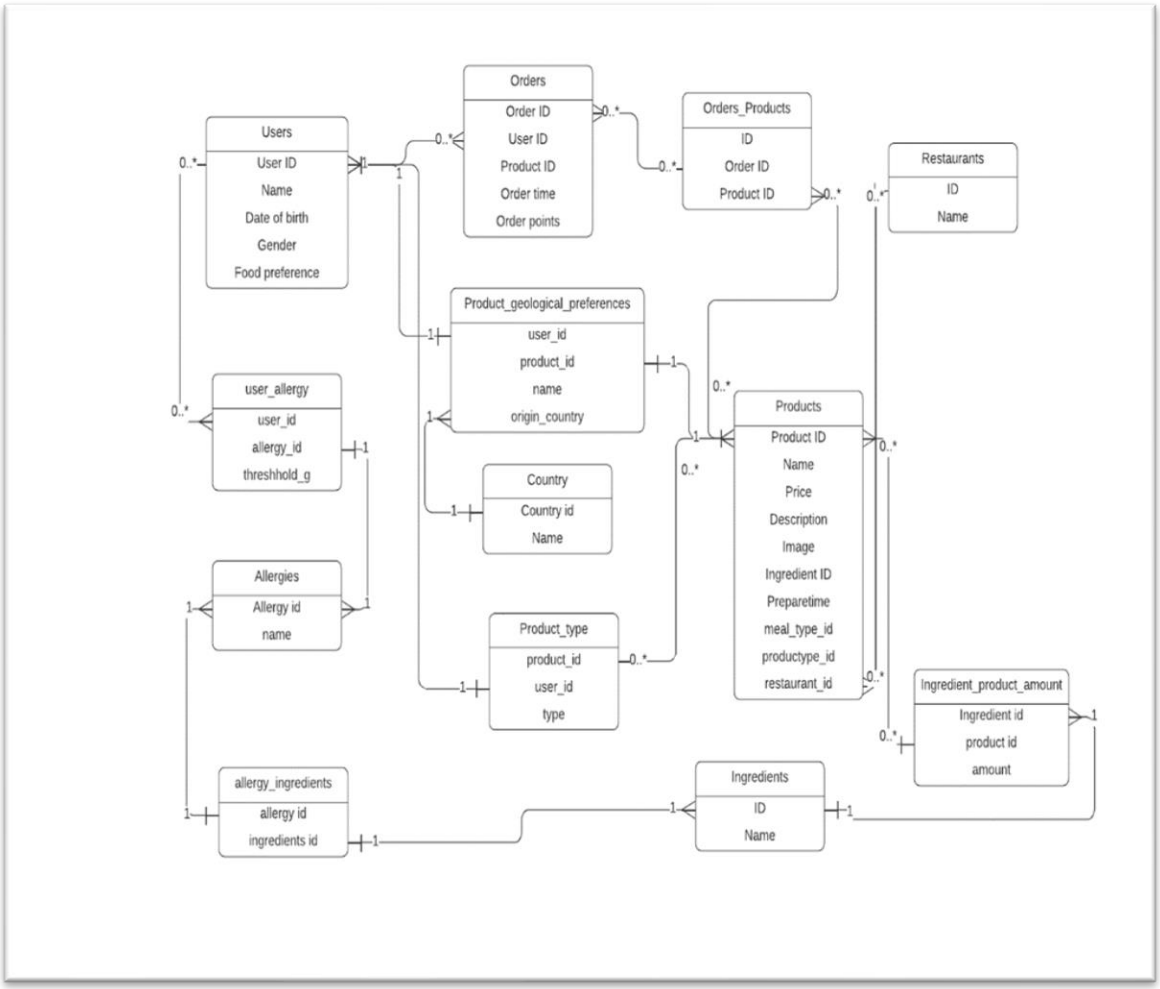


Figure 3.2: Food Recommendation system conceptual model graph

As the figure 3.2 shows, the system is based on 3 entities, User, Product and Order. The system has relation between the product and user within the order entity, that each product the user order is saved into different table and create an order id for that action. Each product can be into many orders, and 1 user can make many orders as the relations between entities show. It is necessary to comment on some of the attributes that require clarification:

Weather: which it might be helpful to know the weather of the day that the order was made on, would make the recommended product meet the need of the user. For example, a user would prefer to order hot drink in a cold weather than having freezing coca cola.

Day: Users would change their orders based on the day. For example, user would go for light dinner on a day during the weekdays, unlike in weekends which followed by a day off.

Date and time: User would order different food or drink based on the time of the day. For example, the user wouldn't order food on lunch same as dinner.

CHAPTER 4
Proposed solution

This phase will focus on the management aspects of the project's detailed design, as the initial idea of the offered solution was discussed in the previous chapter. In addition, the architecture or design of each part's implementation will be detailed in detail.

4.1 Project plan

Due to the emphasis of research and testing development over the rest of the system, the planning for this project has been set according to the plan schedule in figure 4.1.

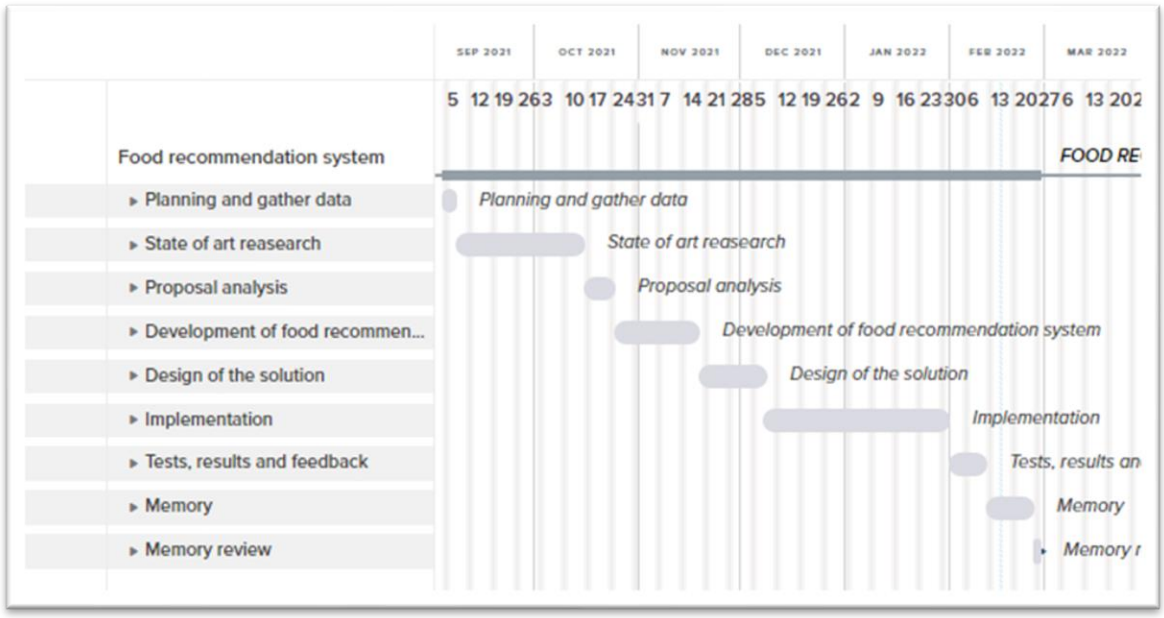


Figure 4.1: Food Recommendation system planning diagram

Working on the project began on the 5th of September 2021, the day the proposal was discussed, and the work plan was established, as indicated in figure 4.1. The first stage was to collect information and data on recommendation systems as well as dataset of items to recommend to users. Then adapt the memory structure to the final master's thesis template, which served as the foundation for documenting the entire process.

Following the initial phase, the second step began with some state-of-the-art study. This research phase would serve as the foundation for the rest of the project, thus it was completed in a slow and methodical manner, taking 39 days to develop and test the recommendation system methods and applications that would be employed in that field.

It was divided into 2 different parts as the figure 4.2 shows, on a part of looking for existing solutions to criticize and compare between them. And other part to check all other methods that are used for recommendation system to choose the best for food recommendation system, which took 14 days to finish with existing solutions and another 18 days for the part of recommendation system methods.

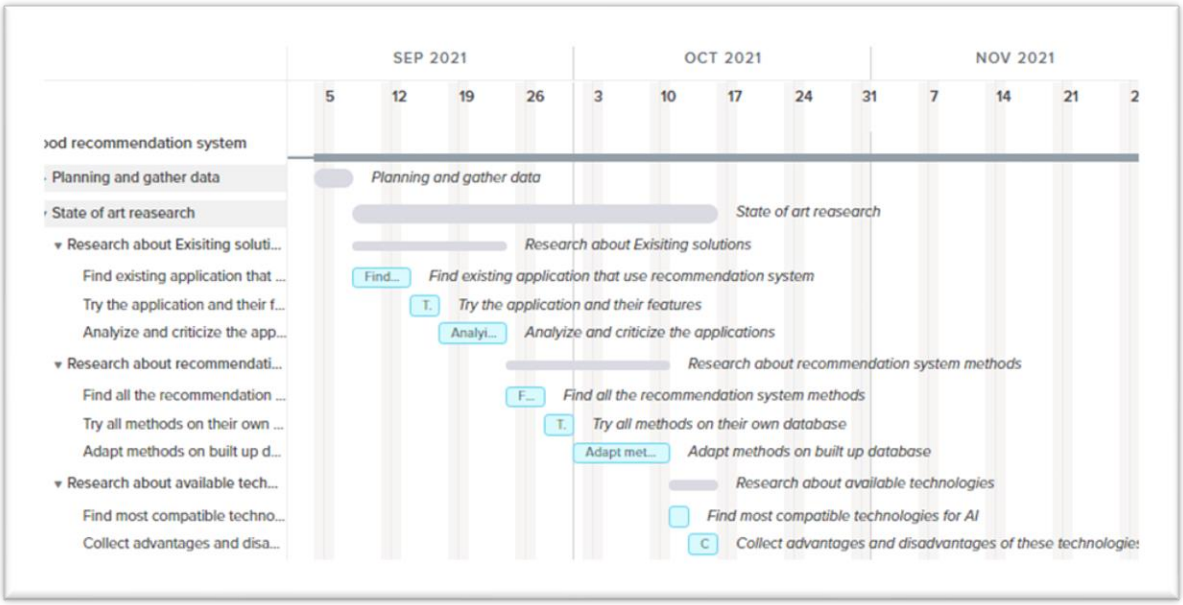


Figure 4.2: Food Recommendation System State of art planning

Once this step was over, the next step was to check out the best environments and technologies to use to implement the recommendation system, therefore in this step the best technologies were selected to carry out the project which it took about 5 days.

Later on, the development phase commenced, this phase began the process of inquiry, trial and error to determine the optimum way for developing a food recommendation system. It was divided into 3 different parts using different methods based on the database that was build after gather data in the initial phase as the figure 4.3 shows, in which different methods were studied and tested, to figure out that hybrid method would reduce the error and give the best accurate result of the prediction to the next product. This step was the longest since it lasted for 75 days to finish it. It was period in which numerous mistakes and changes occurred as a result of many problems posed by having no knowledge of the technology.

The next step was designing and detailing the architecture of the system which carried almost 10 days.

The next step was taking all in action and implementing the hybrid recommendation system an other API's such as weather API that would take specific date into concern and extract the weather on specific day and daytime. Implementing the system carried out for 11 days due to having examples of other recommendation system and the studies were done in previous steps which can be seen in figure 4.4.

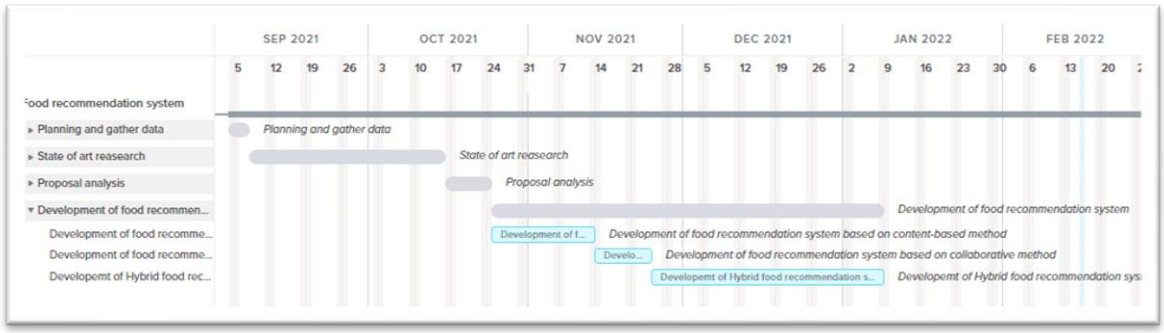


Figure 4.3: Food Recommendation system development diagram

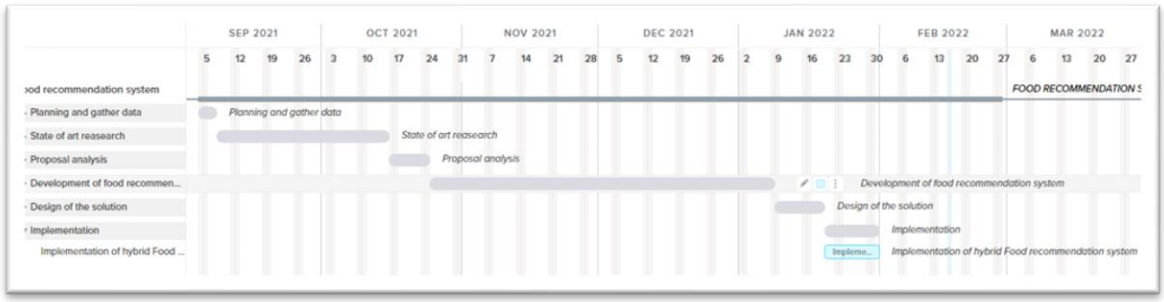


Figure 4.4: Food Recommendation system implementation diagram

Finally, more processes were completed for testing, analysis, and receiving user input. Also, concluding all of the work that had been completed, reviewing the memory, with the project will be completed on Sunday February 27th.

4.2 Budget

4.2.1 Information requirements

Because only me was in charge of all tasks during the execution of this project, there were no external charges. And that the master's project was completed by a student without income, the funding was cut. Which it has led to free usage of open source tools in order to save money on technology.

Aside from that, student plans have been adopted from the platforms like GitHub[19], which provide a student pack for non-commercial product development. In terms of the system's implementation, it was done utilizing a free environment called Visual Studio Code, which includes all of the Python language's and machine learning libraries extensions.

Finally, a trial license has been used for APIs such as the Weather API to retrieve weather of specific dates and times.

4.2.2 Developer budget

Time was the only cost associated with implementing the project. When we look at the typical wage for a junior software engineer in Spain[20], we get € 18,795 gross per year for a full time work of 40 hours per week, or € 1,566 monthly in 12 payments.

When the price per hour for a junior software developer is subtracted, we arrive to a gross wage of € 9.03 per hour. If we simply consider the time spent on development and implementation, we can estimate that it took an average of 4 hours per day to complete.

That would give a total of 87 days * € 4h/day * € 9.03/h = € 3,142 gross salary of the junior software engineer.

4.3 System architecture

To begin, a recommendation system's overall architecture must be described in detail. An architecture which is divided into 4 parts, recommendation services, training inference, data processing and storage, and underlying basic data. As you can see in the figure 4.5.

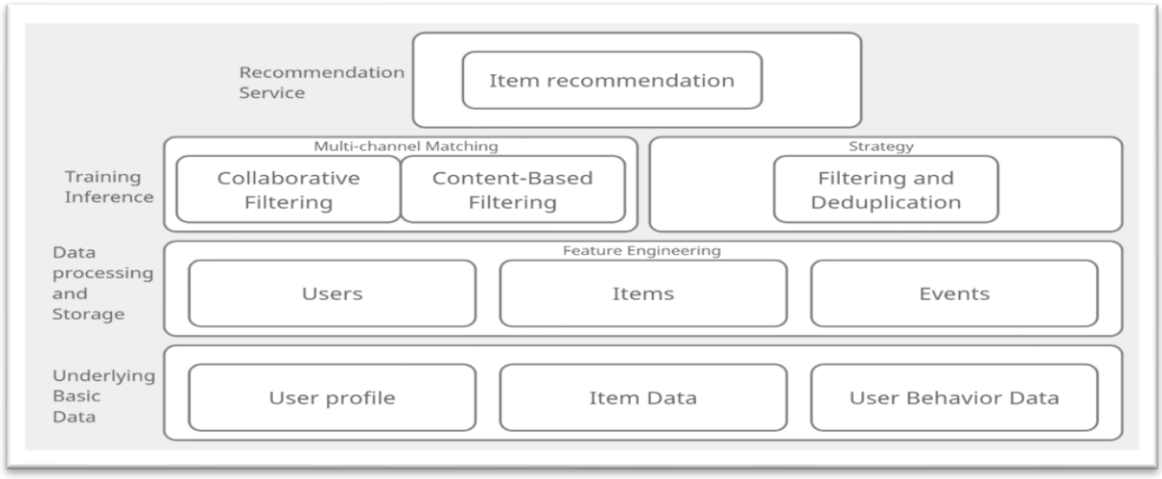


Figure 4.5: Food Recommendation system overall architecture

All of these components are found in the backend strand, which houses all of the business logic and persistence access.

4.3.1 Backend architecture

Python was used to implement this component of the project. Figure 4.5 depicts the architecture's several levels, which include:

- **Underlying basic data:** User profile, item data, and user activity data are all stored at this layer. The user profile may include information such as the user's name, gender, age, allergies, preferences, and nationality. The product name, allergy type, origin country, preparation time, discount, ingredients, description, average vote, vote count, and product type are all included in the item data. The behavior data relates to interaction between the user and the item, The behavior data relates to interaction between the user and the object, in this case, the item is food or drink, and when a user orders a product, they can add it to their favorite list, pay for it, and it will also be added to their order history. The user behavior data is made up of these acts.

- **Data processing and data storage:** We can utilize this layer to perform data processing such as identifying user characteristics, product characteristics, and order characteristics.

- **Training Inference:** Matching and ranking are the two most significant modules in this layer. In this project since it uses a hybrid filtering it use two different types to filtering which are collaborative filtering and content-based filtering for matching then set up score for each product based on the matching. Next must filter and deduplicate the recommendation results based on the user's profile if he has any allergies or food preferences.

- **Recommendation service:** This is the top layer will can recommend a product. For example in this project case the system will recommend either food or drink for the user based on the details well mentioned in the Training inference layer.

4.4 Detailed design

This chapter describes the design that was created to complete the project while following all of the specifications and requirements laid out in the preceding chapters. The design

here lays forth the groundwork for future development and implementation. The design will be broken down into sections in the following sections.

4.4.1 Database design

A final design of the database that the developing system will use is required. The scheme of the real database built for this system, as shown in figure 4.6, will be used by this system.

The established association related to connections such as primary and foreign keys because this database is relational. Each table represent a collection, and the relationships between them are their keys, each of which can refer to documents from other collections. Finally, each table's main key is a set of unique keys that distinguishes it from other tables.

The data types that appear in the tables that have been used in the system were only user table, allergy table, products table, ingredients table, and orders table. However, the rest of tables are planned for a future work such as adding ingredients amount in each product.

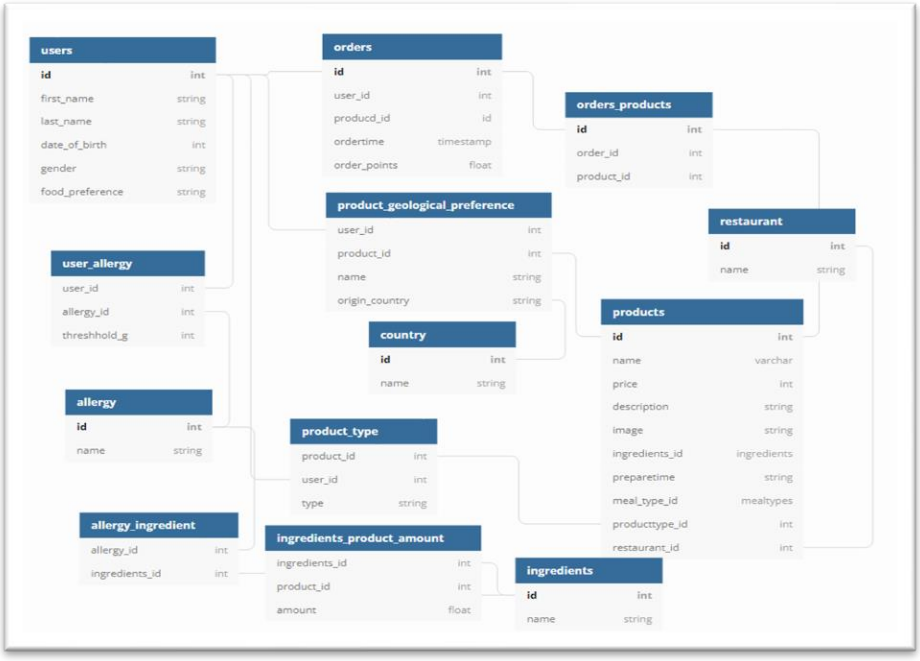


Figure 4.6: Food Recommendation database diagram

As the figure 4.6 shows some relations the most important relations are between users and orders, and products and orders.

4.4.2 Backend design

Because there aren't many services to supply to an application, the backend doesn't have any special or too complex logic. This design is based on the architecture described in subsection 4.3.1. Furthermore, the backend can be divided into two parts.

- **Food recommendation system design:**

  This design of food recommendation system is divided into 2 parts, and that's due to using hybrid filtering recommendation. Which we will separately detail how each recommending method works.

Content-based filtering:

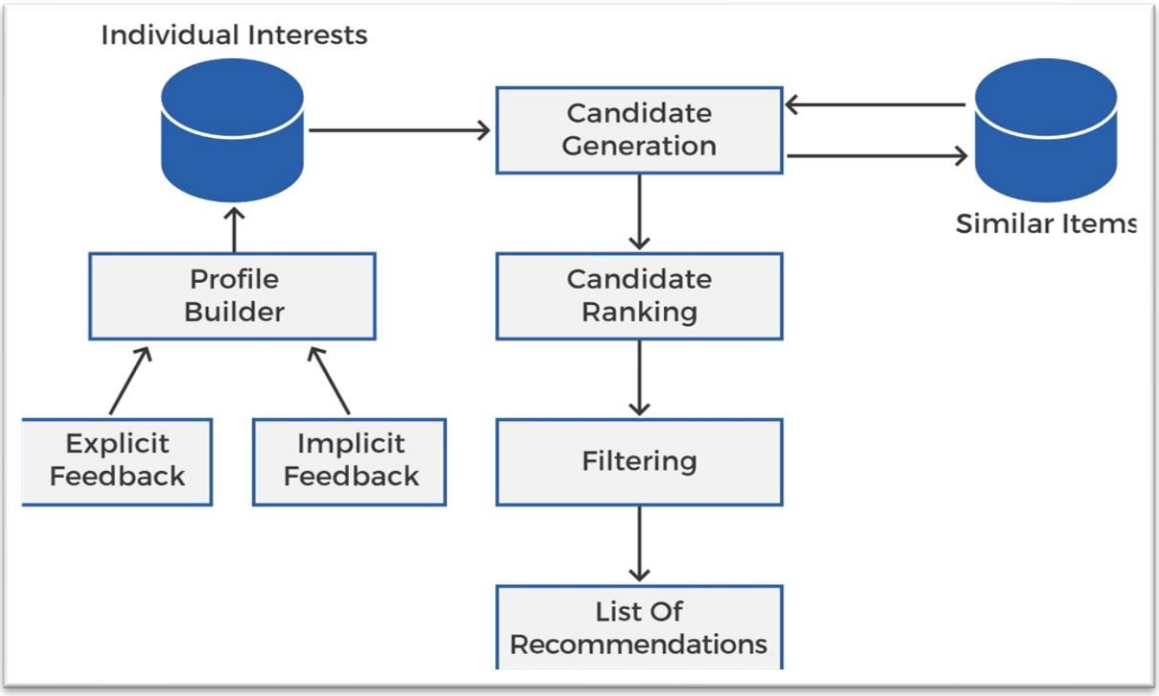


Figure 4.7: Content-based filtering design diagram

As the figure 4.7 shows, the following design focus on having a profile for the user, and the user history orders, from this point the system find the similarities between each order and other products and score these products, then sort them out. Later the system filters out the product based on the user's allergies and preferences. Finally, it lists out the list of recommendations for the user.

Collaborative filtering:

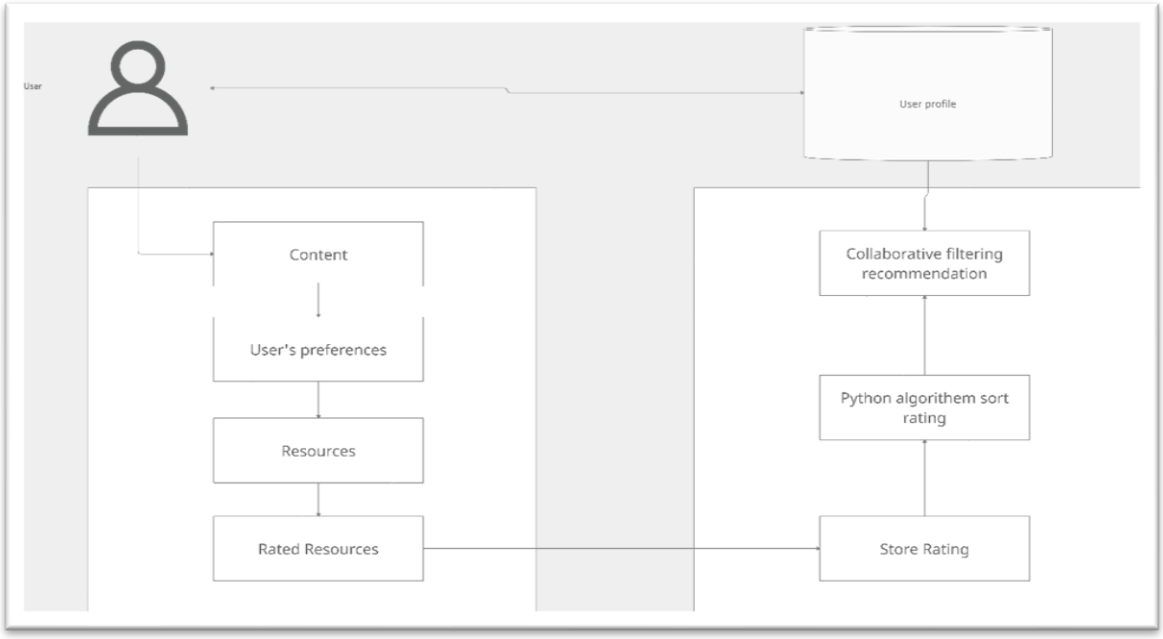


Figure 4.8: Collaborative filtering design diagram

As the figure 4.8 shows, the following design focus on the products and their rating to suggest a product for the user. Therefore, the system checks out the user's profile and his interests. After that it check out the items with their rates that meet the user profile and store them. Later, it sorts out the items using python algorithm. Finally, it filters out the products based on user's allergies and food type preference and list them out of the user.

However, in this system since it is a hybrid recommendation system that mix between both filtering, Before the system reach the point where to sort out the score of the items, it takes the score of each product from content-based filtering and the score of each product from collaborative filtering then multiply them up after that it comes to the point to filter out based on the user's profile preferences and allergies and list them out to the user as the Figure 4.9 shows.

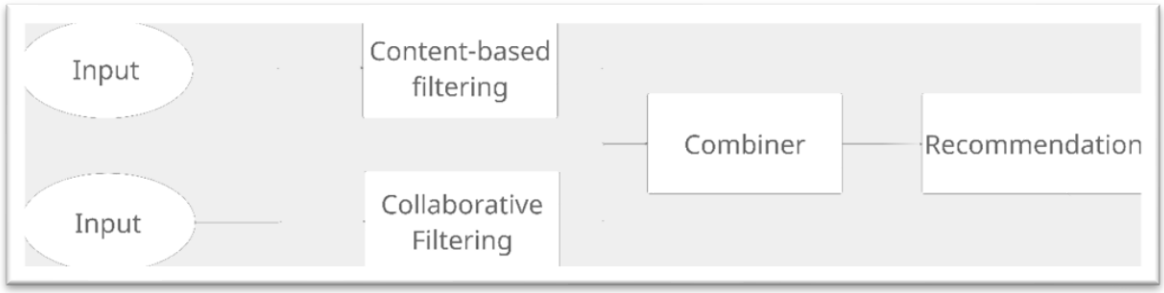


Figure 4.9: Hybrid filtering design diagram

- **Weather API design**

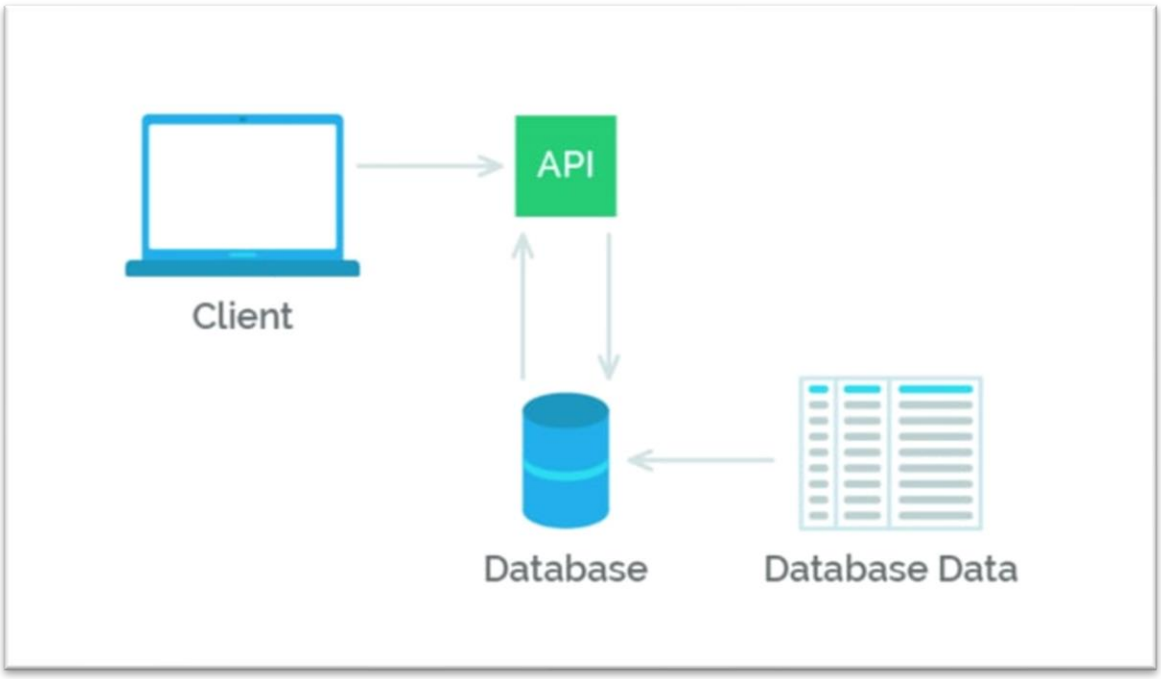


Figure 4.10: Weather API design diagram

As the figure 4.10 shows, the weather API[21] retrieve data from worldweatheronline, which require various inputs such as, api key, location, start date and end date. Later it save the data into a csv file, that the csv file contain various output but the most important and that serve this system are, max temperature, min temperature, humidity. That can be converted into general weather of the day.

4.4.3 Design of a user interface flowchart

The flow a user will follow with user input is an important part of design. This identifies various business rules that the program must follow, as well as the actions that the user is permitted to take when navigating the system. Figure 4.11 illustrates this navigation. Even though no UI application was created for this project, this would be the most basic flowchart a user would follow to get a recommended item.

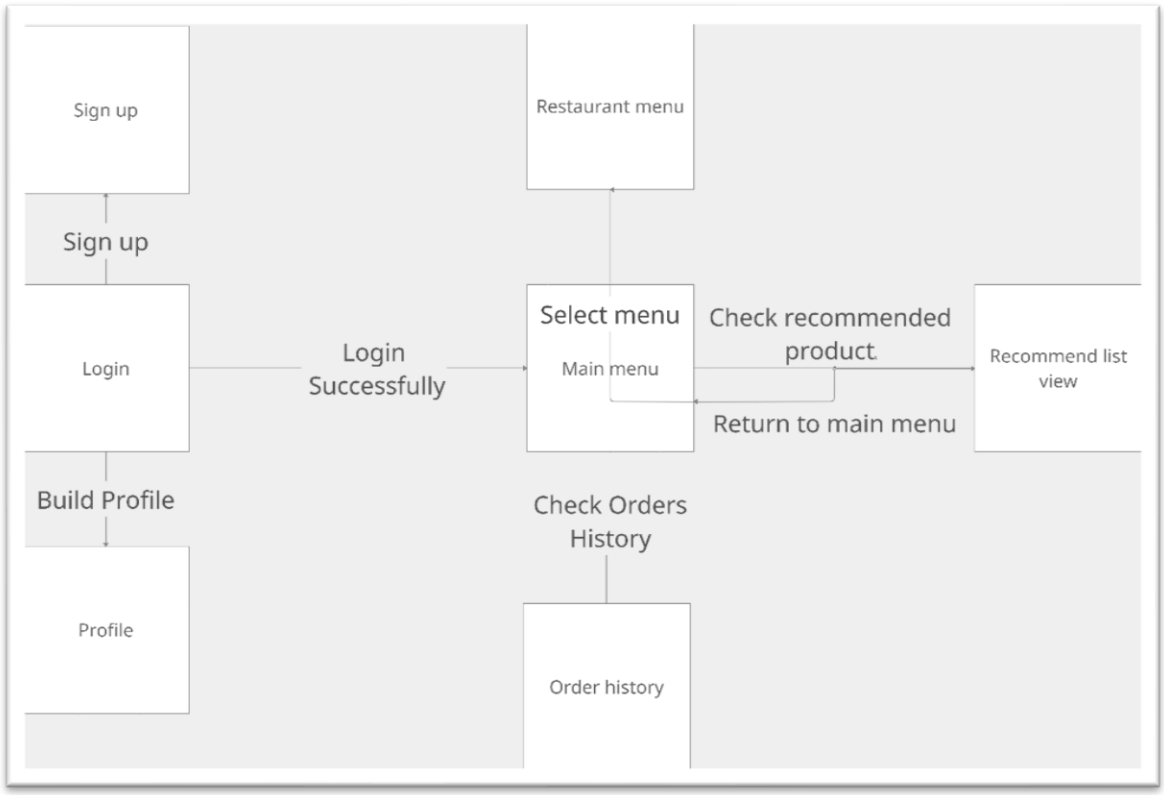


Figure 4.11: Application flowchart design

It's worth mentioning that once you've signed up, you'll need to login and create a profile.

4.5 Development of the proposed solution

This part will go over the steps involved in building the proposed solution that was discussed in the previous chapter.

4.5.1 Development of the food recommendation system

This phase of the project is the most difficult and where the logic lies, with less clarity in terms of the length of time it will take to complete and the quality of the outcomes that can be expected.

Because of the great complexity of the tasks, as well as the lack of understating in the field of machine learning, the granularity of the tasks had to be kept very tiny to build a depth of knowledge and thoroughly evaluate each step.

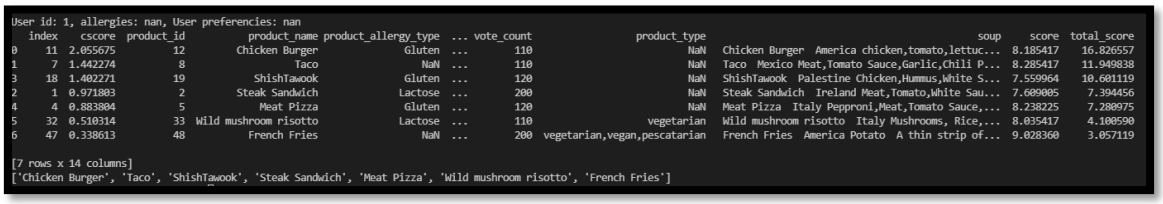
Version control and project creation

The initial step was to build a GitHub repository to hold the implementation while searching for a solution to the recommendation system.

Following that, a separate branch was developed for the purpose of testing certain procedures. Code conflicts are uncommon as a single user of the repository, but it provides a tool to track development and to try different methods of recommendation system.

Developing Food recommendation System

The first step is to be able to build up a profile and store it into our database, that we can retrieve it for learning and recommending methods. Let's assume we have a user retrieved of the database that has no food preference and no allergies, with order history of Nuggets, Chicken tender, and Schnitzel which all these products are Chicken recipe based. Moreover, these products will be given two separated scores using two filtering methods that will be explain in depth later on this chapter then multiply these two scores to give a total score of each product that will be sorted at the end.



```
User id: 1, allergies: nan, User preferencies: nan
   index    cscore  product_id          product_name product_allergy_type  ... vote_count               product_type                                                soup     score  total_score
0     11  2.055675          12        Chicken Burger              Gluten  ...        110                        NaN  Chicken Burger  America chicken,tomato,lettuc...  8.185417    16.826557
1      7  1.442274           8                  Taco                 NaN  ...        110                        NaN  Taco  Mexico Meat,Tomato Sauce,Garlic,Chili P...  8.285417    11.949838
3     18  1.402271          19           ShishTawook              Gluten  ...        120                        NaN  ShishTawook  Palestine Chicken,Hummus,White S...  7.559964    10.601119
2      1  0.971803           2        Steak Sandwich             Lactose  ...        200                        NaN  Steak Sandwich  Ireland Meat,Tomato,White Sau...  7.609005     7.394456
4      4  0.883804           5            Meat Pizza              Gluten  ...        120                        NaN  Meat Pizza  Italy Pepproni,Meat,Tomato Sauce,...  8.238225     7.280975
5     32  0.510314          33  Wild mushroom risotto             Lactose  ...        110                  vegetarian  Wild mushroom risotto  Italy Mushrooms, Rice,...  8.035417     4.100590
6     47  0.338613          48          French Fries                 NaN  ...        200  vegetarian,vegan,pescatarian  French Fries  America Potato  A thin strip of...  9.028360     3.057119

[7 rows x 14 columns]
['Chicken Burger', 'Taco', 'ShishTawook', 'Steak Sandwich', 'Meat Pizza', 'Wild mushroom risotto', 'French Fries']
```

Figure 4.12: Hybrid recommendation system output

Some data will be missing that we will get it using some API which is:

Weather API

This API use a library called wwo_hist which retrieve weather data from Worldweatheronline. Then it opens a file to save all the weather information retrieved from worldweatheronline, by passing api key, start and end date, location of the city as the figure 4.12 shows. By doing this step we will have the required weather information to pass it as feature for each order to take the weather as feature to recommend a product.

```
from wwo_hist import retrieve_hist_data
import os

os.chdir("D:\Food Recommendation System")

frequency = 3
start_date = '09-OCT-2021'
end_date = '25-OCT-2021'
api_key = 'd269af8a479e424a9f8234819221101'
location_list = ['valencia']
hist_weather_data = retrieve_hist_data(api_key,
                                       location_list,
                                       start_date,
                                       end_date,
                                       frequency,
                                       location_label = False,
                                       export_csv = True,
                                       store_df = True)
```

Figure 4.13: Weather API implementation

Hybrid method

After gathering all the data needed and retrieving it from the database, now it's the time to rank the data by different ranking methods which it comes into two phases in this project:

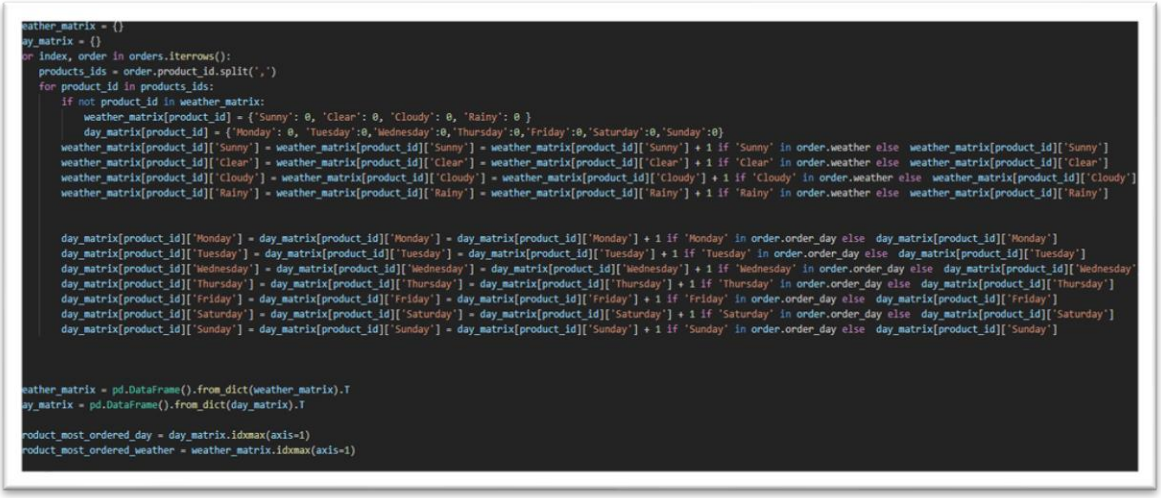
Content-based filtering was carried through these steps:

1. First step is to have a clean data, by converting all keyword instance into lowercase then strip all out of spaces between them as the figure 4.13 shows.

```
def clean_data(x):
    if isinstance(x, list):
        return [str.lower(i.replace(" ", "")) for i in x]
    else:
        #Check if director exists. If not, return empty string
        if isinstance(x, str):
            return str.lower(x.replace(" ", ""))
        else:
            return ''
```

Figure 4.14: Clean data code

2. Second step is to get the most day and weather the product was ordered in, so we can include them as metadata and give weight for each product.



```
eather_matrix = {}
ay_matrix = {}
or index, order in orders.iterrows():
   products_ids = order.product_id.split(',')
   for product_id in products_ids:
      if not product_id in weather_matrix:
         weather_matrix[product_id] = {'Sunny': 0, 'Clear': 0, 'Cloudy': 0, 'Rainy': 0 }
         day_matrix[product_id] = {'Monday': 0, 'Tuesday':0,'Wednesday':0,'Thursday':0,'Friday':0,'Saturday':0,'Sunday':0}
      weather_matrix[product_id]['Sunny'] = weather_matrix[product_id]['Sunny'] = weather_matrix[product_id]['Sunny'] + 1 if 'Sunny' in order.weather else  weather_matrix[product_id]['Sunny']
      weather_matrix[product_id]['Clear'] = weather_matrix[product_id]['Clear'] = weather_matrix[product_id]['Clear'] + 1 if 'Clear' in order.weather else  weather_matrix[product_id]['Clear']
      weather_matrix[product_id]['Cloudy'] = weather_matrix[product_id]['Cloudy'] = weather_matrix[product_id]['Cloudy'] + 1 if 'Cloudy' in order.weather else  weather_matrix[product_id]['Cloudy']
      weather_matrix[product_id]['Rainy'] = weather_matrix[product_id]['Rainy'] = weather_matrix[product_id]['Rainy'] + 1 if 'Rainy' in order.weather else  weather_matrix[product_id]['Rainy']


      day_matrix[product_id]['Monday'] = day_matrix[product_id]['Monday'] = day_matrix[product_id]['Monday'] + 1 if 'Monday' in order.order_day else  day_matrix[product_id]['Monday']
      day_matrix[product_id]['Tuesday'] = day_matrix[product_id]['Tuesday'] = day_matrix[product_id]['Tuesday'] + 1 if 'Tuesday' in order.order_day else  day_matrix[product_id]['Tuesday']
      day_matrix[product_id]['Wednesday'] = day_matrix[product_id]['Wednesday'] = day_matrix[product_id]['Wednesday'] + 1 if 'Wednesday' in order.order_day else  day_matrix[product_id]['Wednesday']
      day_matrix[product_id]['Thursday'] = day_matrix[product_id]['Thursday'] = day_matrix[product_id]['Thursday'] + 1 if 'Thursday' in order.order_day else  day_matrix[product_id]['Thursday']
      day_matrix[product_id]['Friday'] = day_matrix[product_id]['Friday'] = day_matrix[product_id]['Friday'] + 1 if 'Friday' in order.order_day else  day_matrix[product_id]['Friday']
      day_matrix[product_id]['Saturday'] = day_matrix[product_id]['Saturday'] = day_matrix[product_id]['Saturday'] + 1 if 'Saturday' in order.order_day else  day_matrix[product_id]['Saturday']
      day_matrix[product_id]['Sunday'] = day_matrix[product_id]['Sunday'] = day_matrix[product_id]['Sunday'] + 1 if 'Sunday' in order.order_day else  day_matrix[product_id]['Sunday']


eather_matrix = pd.DataFrame().from_dict(weather_matrix).T
ay_matrix = pd.DataFrame().from_dict(day_matrix).T

roduct_most_ordered_day = day_matrix.idxmax(axis=1)
roduct_most_ordered_weather = weather_matrix.idxmax(axis=1)
```

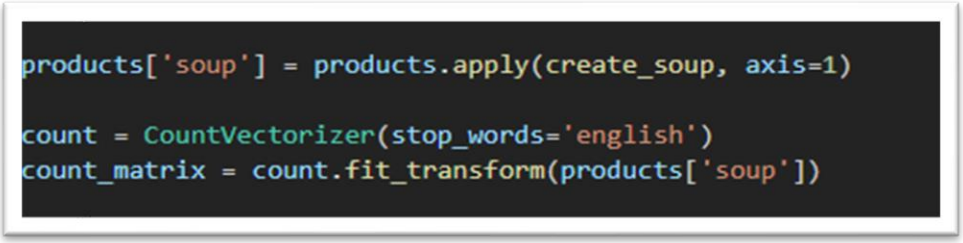
Figure 4.15: Get most weather and day of product order

3. Third step would be creating the soup of all the metadata, which is a string that contain all the metadata that we need to feed our vectorizer as the figure 4.14 shows. By passing these metadata such as product name, product origin country, product ingredients, product most ordered day, product most ordered weather. This function will join all required columns by a space. This is the final preprocessing step.

```
Compute the cosine similarity matrix
ef create_soup(x):
   return ' ' + x['product_name'] + ' ' + ' ' + x['product_origing_country'] + ' ' + x['product_ingredients']+ ' '+ x['product_description'].replace(",", " ") + \
      x['product_most_ordered_day'] + ' ' + x['product_most_ordered_weather']
```

Figure 4.16: Create soup code

4. Fourth step would be feeding our vocabulary with all the products with taking all metadata in concern by using CounterVectorizer[22] and removing all stop words in English such as, ‘the’, ‘a’ as the figure 4.15 shows. By this step we will have vocabulary contain most important words to compare each product and it’s feature with other products.



```
products['soup'] = products.apply(create_soup, axis=1)

count = CountVectorizer(stop_words='english')
count_matrix = count.fit_transform(products['soup'])
```

Figure 4.17: CounterVectorizer

5. Fifth step, by using the cosine similarity formula to calculate a numeric quantity that denotes the similarity between two products and their features that has been fed to the vectorizer in the previous step.

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}^{\mathsf{T}}}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|} = \frac{\sum_{i=1}^{n} \mathbf{x}_i \cdot \mathbf{y}^{\mathsf{T}}{}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{x}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{y}_i)^2}}$$

That will give out an output as the figure 4.16 shows. And the figure 4.17 shows the implementation of the code.

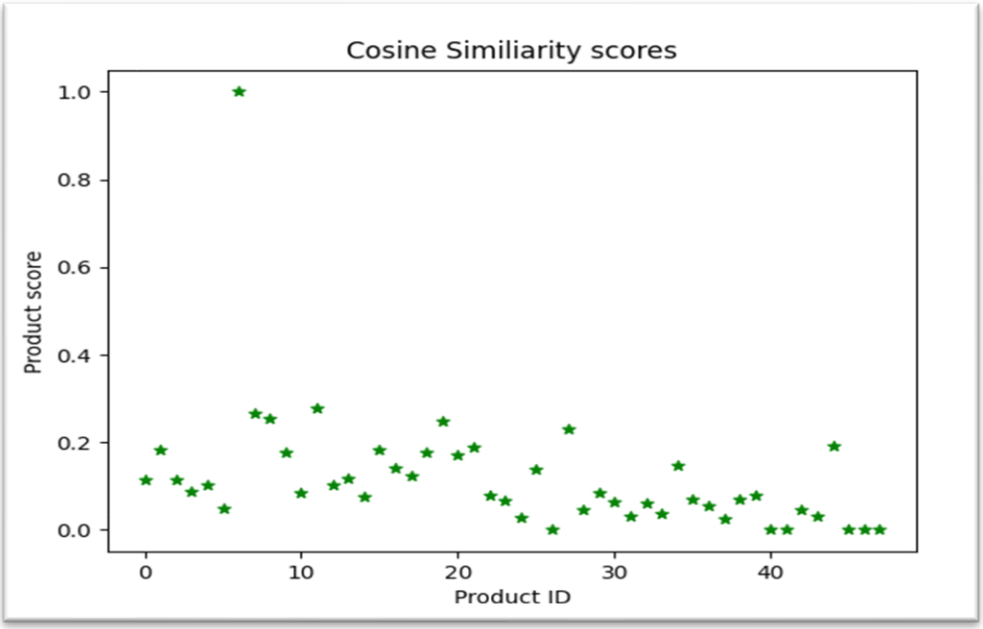


Figure 4.18: Cosine similarity scores to products

```
cosine_sim2 = cosine_similarity(count_matrix, count_matrix)
```

Figure 4.19: Cosine similarity code implementation

6. Last step is to get all the products indexes and the score of each product then sort the highest 10 products using python sorting algorithms as the figure 4.18 shows. In order to list them for the user.

```
def get_recommendations_score(product_name, cosine_sim=cosine_sim2):
    # Get the index of the product that matches the product_name
    idx = indices[product_name]

    # Get the pairwsie similarity scores of all products with that product
    sim_scores = list(enumerate(cosine_sim[idx]))

    # Sort the products based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get the scores of the 10 most similar products
    sim_scores = sim_scores
    # Get the product indices
    product_indices = [i[0] for i in sim_scores]

    return sim_scores
```

Figure 4.20: Sorting products based on score content-based filtering

Collaborative filtering was carried through these steps:

1. We have already retrieved the data from the database in the previous method, therefor our first step here will be calculating the mean of vote average of each product as the figure 4.19 shows. These ratings of products are gathered by the users rating on each product.

```
C = products['vote_average'].mean()
```

Figure 4.21: Calculate the mean of vote average

2. Second step was calculating the minimum number of votes required as the figure 4.20 shows which in this case it products that has more than 90 votes. By doing this step we avoid having high rated products with low votes, to give more weight for the products with more vote counts.

```
m = products['vote_count'].quantile(0.70)
```

Figure 4.22: Calculate the minimum number of votes required

3. Third step would be having a copy of the original metadata then calculate the score of each score as the figure 4.21 shows. Calculating the score by using this formula. In order to rate each product and sort them out.

$$WeightedRating(\mathbf{WR}) = \left(\frac{\mathbf{v}}{\mathbf{v}+\mathbf{m}} \cdot \mathbf{R}\right) + \left(\frac{\mathbf{m}}{\mathbf{v}+\mathbf{m}} \cdot \mathbf{C}\right)$$

```
def collaborative():
    q_products = products.copy().loc[products['vote_count'] >= m]
    q_products['score'] = q_products.apply(weighted_rating, axis=1)
    return q_products

def weighted_rating(x, m=m, C=C):
    v = x['vote_count']
    R = x['vote_average']
    # Calculation of the score
    return (v/(v+m) * R) + (m/(m+v) * C)
```

Figure 4.23: Calculate the score of product collaborative rating

Calculation of the score of both methods

After we got the scores of the products by 2 different methods, collaborative and content-based filtering, now we will multiply the score of both scores into a total score to give the most accurate prediction to the user as the figure 4.22 shows the implementation and figure 4.23 shows example of the output possible.

```
for product_id in product_id_array:

    product_row = products.loc[int(product_id)]
    product_name = product_row['product_name']
    scores = get_recommendations_score(product_name, cosine_sim2)
    for score in scores:
        product_index = score[0]
        product_score = score[1]
        if product_index in sum_product_score:
            sum_product_score[product_index] = sum_product_score[product_index] + product_score
        else:
            sum_product_score[product_index] = product_score
sum_product_score_sorted = sorted(sum_product_score.items(), key=lambda item: item[1], reverse=True)

# Multiplying collaborative by content-based
c_based_products = pd.DataFrame.from_dict(sum_product_score.items())
c_based_products.columns=["index","cscore"]

q_products = collaborative()
two_scores = pd.merge(c_based_products, q_products, how='inner', on='index')
two_scores['total_score'] = two_scores['score'] * two_scores['cscore']
sorted_products = two_scores.sort_values('total_score', ascending=False)
print(sorted_products)
```

Figure 4.24: Multiply collaborative filtering score by content-based filtering score

11	2.055675	12	Chicken Burger	Gluten	...	110	NaN	Chicken Burger America chicken,tomato,lettuc...	8.185417	16.826557
7	1.442274	8	Taco	NaN	...	110	NaN	Taco Mexico Meat,Tomato Sauce,Garlic,Chili P...	8.285417	11.949838
18	1.402271	19	ShishTawook	Gluten	...	120	NaN	ShishTawook Palestine Chicken,Hummus,White S...	7.559964	10.601119
1	0.971883	2	Steak Sandwich	Lactoze	...	200	NaN	Steak Sandwich Ireland Meat,Tomato,White Sau...	7.609005	7.394456
4	0.883804	5	Meat Pizza	Gluten	...	120	NaN	Meat Pizza Italy Pepproni,Meat,Tomato Sauce,...	8.238225	7.280975
32	0.510314	33	Wild mushroom risotto	Lactoze	...	110	vegetarian	Wild mushroom risotto Italy Mushrooms, Rice,...	8.035417	4.100590
47	0.338613	48	French Fries	NaN	...	200	vegetarian,vegan,pescatarian	French Fries America Potato A thin strip of...	9.028360	3.057119

Figure 4.25: Output of multiply filtering methods

Python sorting and filtering algorithm

Finally, once we have all the scores sorted of all products its time to meet the user's profile of allergies and preferences with the products by filtering out the products that the user has allergy to and filter out the products type doesn't meet the user's preference type as the figure 4.24 shows and the figure 4.25 shows an output example.

```
recomended_products = []
for index, product in sorted_products.iterrows():

    product_allergies = [val for val in str(product['product_allergy_type']).lower().split(',') if not val == 'nan']
    if user_allergies in product_allergies:
        continue

    if not pd.isna(user_preferencies) and not user_preferencies in str(product["product_type"]).lower().split(","):
        continue

    if not product['product_name'] in recomended_products:
        recomended_products.append(product['product_name'])
```

Figure 4.26: Implementation of filtering algorithm

```
User id: 1, allergies: lactoze, User preferencies: nan
['Chicken Burger', 'Taco', 'ShishTawook', 'Meat Pizza', 'French Fries']
```

Figure 4.27: Output of food recommendation system

As the figure 4.25 shows the output contain the user id, the allergies of the user and the user preferences. Also, it shows the recommended product for the user based on his profile.

CHAPTER 5
Implementation

This chapter describes the tests used to ensure that the developed recommendation system working properly.

5.1 Food recommendation system

This section will detail all of the tests carried out to determine whether the recommendation system fits the requirements and is capable of making an accurate forecast.

Because each test section has a hypothesis to test, the tests in this subsection will be evaluated using a specified criterion, also it will be tested by three different methods to compare between the results of each method.

5.1.1 Food recommendation system user preferences and allergies tests

The tests in this area are aimed at determining different dietary preferences and allergies.

- **Food type preferences:** These attributes represent the user's food preferences, and by that the recommendation system can recommend the food type the user should have.
    - **Without preferences:** This attribute means that the user has no specific food preferences, therefore there would be no filtering over food preferences on recommending products.
    - **Vegetarian:** This attribute means that the user does not eat meats, therefore the system should list only the food that are specified as vegetarian or vegan food.

- **Vegan:** This attribute means that the user does not eat meats or their productions, therefore the system should only list the food that are specified as vegan food.

- **Pescatarian:** This attribute means that the use does not eat any animal meat other than the sea food, therefore the system should exclude all the meats that are no considered as sea food.

- **User Allergies:** These attributes represent the user's allergies, and by that the recommendation system should avoid recommending any product that consider danger to the user's life. In this system few of allergies/intolerance were mentioned such as, lactose, fructose, and gluten.

    - **Lactose:** This attribute means that the user intolerant to lactose. Which should avoid products that contain any product of milk and that's for the sugar in the milk.

    - **Fructose:** This attribute means that the user intolerant to fructose. Which should avoid "Fruit sugar".

    - **Gluten:** This attribute means that the user intolerant to gluten. Gluten is a structure protein naturally found in grains such as wheat, barley, and rye.

- **Number of votes for a product:** In these test cases we will change the number of products valid for collaborative filtering, and that's based on the number of votes counted to rate the product. As the it was described in the subsection 4.5.1 collaborative filtering method.

Test resources

For the tests there are many users with different allergies and food preferences, also order history for each user. And number of vote counts on each product.

Test cases

Test case 1 will represent a user without preferences and without allergies. The test will be carried out by 3 different method and compare the results between all of them. Starting with content-based, then collaborative, and finishing with hybrid filtering.

The User profile and order history are listed below:

User ID	1
Name	User1
Gender	Male
Age	18
Nationality	Palestinian
Allergies	No Allergies
Preferences	No preference

Table 5.1: User profile Test case 1

Order ID	1	2	3	4	5	6	7	8
Product ID	6	42	15	39	17	25	12	16
Product name	Hot dog	Fish and Chips	Wings	Quinoa	Nuggets	Caesar salad	Chicken burger	Chicken tender
Order day	Saturday	Sunday	Wednesday	Sunday	Wednesday	Sunday	Monday	Sunday
Order date	2021-05-15	2021-05-16	2021-05-19	2021-05-23	2021-05-26	2021-05-30	2021-05-31	2021-06-06
Order time	15:19	21:20	15:00	14:00	19:30	12:00	13:00	16:30
Weather	Rainy	Cloudy	Sunny	Sunny	Sunny	Sunny	Cloudy	Sunny

Table 5.2: Order history for User 1 of test case 1

Content-based filtering

Product ID	Product Name	Product allergy	Product type	Score
12	Chicken Burger	Gluten	None	2.055675
8	Nuggets	None	None	1.442274
19	Chicken curry	Gluten	None	1.402271
2	Broasted	Lactose	None	0.971803
5	Chicken burger	Gluten	None	0.883804
33	Calamari	Lactose	Vegetarian	0.510314
48	Schnitzel	None	Vegetarian, Vegan, Pescatarian	0.338613

Table 5.3: Test case 1 Content-based filtering

Collaborative filtering

Product ID	Product Name	Product allergy	Product type	Score
48	French Fries	None	Vegetarian, Vegan, Pescatarian	9.028360
8	Taco	None	None	8.285417

5	Meat Pizza	None	None	8.238225
12	Chicken Burger	Gluten	None	8.185417
33	Wild mushroom risotto	None	Vegetarian	8.035417
2	Steak Sandwich	None	None	7.609005
19	ShishTawook	None	None	7.559964

Table 5.4: Test case 1 Collaborative filtering

Hybrid Filtering

Product ID	Product Name	Product allergy	Product type	Score
12	Chicken Burger	None	None	16.826557
8	Taco	None	None	11.949838
19	ShishTawook	None	None	10.601119
2	Steak Sandwich	None	None	7.394456
5	Meat Pizza	Gluten	None	7.280975
33	Wild mushroom risotto	None	Vegetarian	4.100590
48	French Fries	Gluten	Vegetarian, Vegan, Pescatarian	3.057119

Table 5.5: Test case 1 Hybrid filtering

Analysis test case 1

If we observe table 5.2 which is the historical order of the user, we realize that most orders of the user are products that include chicken as ingredient. First test was carried out on a content-based to calculate the similarity between the products were ordered and all other products as table 5.3 shows, later it was performed on collaborative filtering as the table 5.4 shows, to give these scores based on a formula takes the vote count and the average rate into account. Fully detailed in subsection 4.5. If we compare the results in the two methods, we find that content-based filtering has better results than collaborative. That is, the recommended products in the content-based filtering are more related to the products that contain chicken as ingredient, which is better than the recommended products that were given by the collaborative filtering. However, another test was carried out using a hybrid method to include rating of products and the similarities between the products by multiplying the score of each product by each method. Results are shown in table 5.5. This approach gives the best recommended products among the other two approaches.

Test case 2 will represent a user with preferences and no allergies.

The User profile and order history are listed below:

User ID	3
Name	User3
Gender	Male
Age	18
Nationality	Palestinian
Allergies	Lactose
Preferences	No preference

Table 5.6: User profile Test case 2

Order ID	1	2	3	4	5	6	7	8
Product ID	6	42	15	39	17	25	12	16
Product name	Hot dog	Fish and Chips	Wings	Quinoa	Nuggets	Caesar salad	Chicken burger	Chicken tender
Order day	Saturday	Sunday	Wednesday	Sunday	Wednesday	Sunday	Monday	Sunday
Order date	2021-05-15	2021-05-16	2021-05-19	2021-05-23	2021-05-26	2021-05-30	2021-05-31	2021-06-06
Order time	15:19	21:20	15:00	14:00	19:30	12:00	13:00	16:30
Weather	Rainy	Cloudy	Sunny	Sunny	Sunny	Sunny	Cloudy	Sunny

Table 5.7: Order history for User 3 of test case 2

Content-based filtering

Product ID	Product Name	Product allergy	Product type	Score
17	Popcorn Chicken	None	None	2.625219
16	Nuggets	None	None	2.583744
19	Chicken curry	None	None	2.353550
12	Broasted	None	None	2.310922
11	Chicken burger	Gluten	None	2.055675
42	Calamari	None	Pescatarian	1.826165
22	Schnitzel	None	None	1.786756

Table 5.8: Test case 2 Content-based filtering

Collaborative filtering

Product ID	Product Name	Product allergy	Product type	Score
48	French Fries	None	Vegetarian, Vegan, Pescatarian	9.193269
10	Shawerma	Gluten	None	8.992038
41	Dynamite shrimp	None	Pescatarian	8.728333
21	Msahab	Gluten	None	8.685127
37	Crispy vegan quinoa cakes	None	Vegetarian, Vegan	8.580657
29	Vegetarian Pasta	None	Vegetarian	8.435417
31	Cauliflower curry	None	Vegetarian	8.435417

Table 5.9: Test case 2 Collaborative filtering

Hybrid Filtering

Product ID	Product Name	Product allergy	Product type	Score
17	Nuggets	None	None	20.071387
18	Popcorn chicken	None	None	19.448500
20	Chicken curry	None	None	18.283158
13	Broasted	None	None	17.454739
12	Chicken Burger	Gluten	None	16.835375
43	Calamari	None	Pescatarian	14.953529
21	Msahab	Gluten	None	14.512310

Table 5.10: Test case 2 Hybrid filtering

Analysis test case 2

In this case the same approaches were done on the three methods of recommendation but with having a user allergy to lactose as table 5.6 shows. We will achieve the following results as table 5.10 shows. Which we can come with a conclusion that the system gives more weight for the products that are lactose free and eliminate the product that contain lactose.

Test case 3 will represent a user with preferences and no allergies.

The User profile and order history are listed below:

User ID	7
Name	User7
Gender	Female
Age	34
Nationality	Swedish
Allergies	None
Preferences	Vegetarian

Table 5.11: User profile Test case 3

Order ID	1	2	3	4
Product ID	39	33	32	35
Product name	Quinoa	Wild mushroom risotto	Kung pao tofu	Vegetable biryani
Order day	Saturday	Sunday	Wednesday	Sunday
Order date	2021-05-15	2021-05-16	2021-05-19	2021-05-23
Order time	15:19	21:20	15:00	14:00
Weather	Rainy	Cloudy	Sunny	Sunny

Table 5.12: Order history for User 7 of test case 3

Content-based filtering

Product ID	Product Name	Product allergy	Product type	Score
40	Gluten-Free Pizza	Lactose	Vegetarian	1.436927
33	Wild mushroom risotto	Lactose	Vegetarian	1.404663
29	Vegetarian Pasta	None	Vegetarian	0.637530
31	Cauliflower curry	None	Vegetarian	0.544186
35	Vegetable biryani	Gluten	Vegetarian, Vegan	0.442747
23	Greek Salad	None	Vegetarian, Vegan	0.433908
32	Kung pao tofu	None	Vegetarian, Vegan	0.385968

Table 5.13: Test case 3 Content-based filtering

Collaborative filtering

Product ID	Product Name	Product allergy	Product type	Score
48	French Fries	None	Vegetarian, Vegan, Pescatarian	9.193269
37	Crispy vegan quinoa cakes	None	Vegetarian, Vegan	8.580657
27	Hummus	None	Vegetarian, Vegan	8.435417
29	Vegetarian Pasta	None	Vegetarian	8.435417
39	Quinoa	None	Vegetarian, Vegan	8.435417
31	Cauliflower curry	None	Vegetarian	8.435417
24	Arabic Salad	None	Vegetarian, Vegan	8.301786

Table 5.14: Test case 3 Collaborative filtering

Hybrid Filtering

Product ID	Product Name	Product allergy	Product type	Score
40	Gluten-Free Pizza	Lactose	Vegetarian	11.679821
33	Wild mushroom risotto	Lactose	Vegetarian	11.231105
29	Vegetarian Pasta	None	Vegetarian	5.377828
31	Cauliflower curry	None	Vegetarian	4.590433
23	Greek Salad	None	Vegetarian, Vegan	3.660197
35	Vegetable biryani	None	Vegetarian, Vegan	3.600727
32	Kung pao tofu	Gluten	None	3.178603

Table 5.15: Test case 3 Hybrid filtering

Analysis test case 3

In this case the same approaches were done on the three methods of recommendation but with a vegetarian user as table 5.11 shows. We will achieve the following results as table 5.15 shows. Which we can come with a conclusion that the system gives more weight for the products that considered as vegetarian or vegan products and eliminate the products that contain meat.

Test case 4 will represent a user with preferences and no allergies.

The User profile and order history are listed below:

User ID	8
Name	User8
Gender	Male
Age	60
Nationality	Polish
Allergies	None
Preferences	Vegan

Table 5.16: User profile Test case 4

Order ID	1	2	3	4
Product ID	39	38	37	27
Product name	Quinoa	Gluten-Free Lettuce Wrap	Crispy vegan quinoa cakes	Hummus
Order day	Saturday	Sunday	Wednesday	Sunday
Order date	2021-05-15	2021-05-16	2021-05-19	2021-05-23
Order time	15:19	21:20	15:00	14:00
Weather	Rainy	Cloudy	Sunny	Sunny

Table 5.17: Order history for User 8 of test case 4

Content-based filtering

Product ID	Product Name	Product allergy	Product type	Score
39	Quinoa	None	Vegetarian, Vegan	1.197606
26	Falafel	None	Vegetarian, Vegan	0.636622
23	Greek Salad	None	Vegetarian, Vegan	0.567527
37	Crispy vegan quinoa cakes	None	Vegetarian, Vegan	0.544186

Table 5.18: Test case 4 Content-based filtering

Collaborative filtering

Product ID	Product Name	Product allergy	Product type	Score
48	French Fries	None	Vegetarian, Vegan, Pescatarian	9.193269
37	Crispy vegan quinoa cakes	None	Vegetarian, Vegan	8.580657
27	Hummus	None	Vegetarian, Vegan	8.435417
39	Quinoa	None	Vegetarian, Vegan	8.435417
23	Greek Salad	None	Vegetarian, Vegan	8.435417
24	Arabic Salad	None	Vegetarian, Vegan	8.301786

Table 5.19: Test case 4 Collaborative filtering

Hybrid Filtering

Product ID	Product Name	Product allergy	Product type	Score
39	Quinoa	None	Vegetarian, Vegan, Pescatarian	10.102302
26	Falafel	Lactose	Vegetarian, Vegan	5.213339
23	Greek Salad	None	Vegetarian, Vegan	4.787325
36	Crispy vegan quinoa cakes	None	Vegetarian, Vegan	3.713912
24	Arabic Salad	None	Vegetarian, Vegan	3.152383
27	Hummus	None	Vegetarian, Vegan	1.124677
35	Vegetable biryani	Gluten	Vegetarian, Vegan	1.089831

Table 5.20: Test case 4 Hybrid filtering

Analysis test case 4

In this case the same approaches were done on the three methods of recommendation but with a vegan user as table 5.16 shows. We will achieve the following results as table 5.20 shows. Which we can come with a conclusion that the system gives more weight for the products that considered as vegan products and eliminate the products that contain meat or any product that contain meat products.

Test case 5 will represent a user with preferences and allergies.

The User profile and order history are listed below.

User ID	10
Name	User10
Gender	Female
Age	61
Nationality	Spanish
Allergies	Fructose
Preferences	Pescatarian

Table 5.21: User profile Test case 5

Order ID	1	2	3	4
Product ID	42	44	43	47
Product name	Fish and chips	Crab Salad	Calamari	Smoked Salmon canapes
Order day	Saturday	Sunday	Wednesday	Sunday
Order date	2021-05-15	2021-05-16	2021-05-19	2021-05-23
Order time	15:19	21:20	15:00	14:00
Weather	Rainy	Cloudy	Sunny	Sunny

Table 5.22: Order history for User 10 of test case 5

Content-based filtering

Product ID	Product Name	Product allergy	Product type	Score
45	Mussels	None	Pescatarian	1.150603
43	Calamari	None	Pescatarian	1.132682
48	French Fries	None	Vegetarian, Vegan, Pescatarian	1.000000
26	Falafel	None	Vegetarian, Vegan, Pescatarian	0.578823

Table 5.23: Test case 4 Content-based filtering

Collaborative filtering

Product ID	Product Name	Product allergy	Product type	Score
48	French Fries	None	Vegetarian, Vegan, Pescatarian	9.193269
41	Dynamite shrimp	None	Pescatarian	8.728333
46	Smoked Salmon salad	None	Pescatarian	8.435417
42	Fish and chips	None	Vegetarian, Vegan	8.435417
39	Quinoa	None	Vegetarian, Vegan, Pescatarian	8.435417
43	Calamari	None	Vegetarian, Vegan	8.188487

Table 5.24: Test case 4 Collaborative filtering

Hybrid Filtering

Product ID	Product Name	Product allergy	Product type	Score
45	Mussels	None	Pescatarian	9.418165
43	Calamari	None	Pescatarian	9.274955
48	French Fries	None	Vegetarian, Vegan, Pescatarian	9.193269
39	Quinoa	None	Vegetarian, Vegan	4.100912
46	Smoked Salmon salad	None	Pescatarian	1.848862
42	Fish and chips	None	Pescatarian	1.830929
41	Dynamite shrimp	None	Pescatarian	1.679769

Table 5.25: Test case 4 Hybrid filtering

Analysis test case 5

In this case the same approaches were done on the three methods of recommendation but with a pescatarian user and has allergy to fructose as table 5.21 shows. We will achieve the following results as table 5.25 shows. Which we can come with a conclusion that the system gives more weight for the products that considered are fructose free and contain sea food as ingredients, as well eliminates the products that are contain fructose into its ingredients.

Test case 6 will represent a user without preferences and without allergies and high number of votes counts for product to be valid. The test will be carried out by 3 different method and compare the results between all of them. But focusing on collaborative filtering and the number of vote counts for each product to be counted as a valid product for rating. Starting with content-based, then collaborative, and finishing with hybrid filtering.

The User profile and order history are listed below:

User ID	11
Name	User11
Gender	Male
Age	21
Nationality	Spanish
Allergies	None
Preferences	No preference

Table 5.26: User profile Test case 6

Order ID	1	2	3	4	5	6	7	8
Product ID	6	42	15	39	17	25	12	16
Product name	Hot dog	Fish and Chips	Wings	Quinoa	Nuggets	Caesar salad	Chicken burger	Chicken tender
Order day	Saturday	Sunday	Wednesday	Sunday	Wednesday	Sunday	Monday	Sunday
Order date	2021-05-15	2021-05-16	2021-05-19	2021-05-23	2021-05-26	2021-05-30	2021-05-31	2021-06-06
Order time	15:19	21:20	15:00	14:00	19:30	12:00	13:00	16:30
Weather	Rainy	Cloudy	Sunny	Sunny	Sunny	Sunny	Cloudy	Sunny

Table 5.27: Order history for User 11 of test case 6

Content-based filtering

Product ID	Product Name	Product allergy	Product type	Score
12	Chicken Burger	Gluten	None	2.055675
8	Taco	None	None	1.442274
19	ShishTawook	Gluten	None	1.402271
2	Steak Sandwich	Lactose	None	0.971803

5	Meat Pizza	Gluten	None	0.883804
33	Wild mushroom risotto	Lactose	Pescatarian	0.510314
38	French Fries	None	None	0.338613

Table 5.28: Test case 6 Content-based filtering

Collaborative filtering

Product ID	Product Name	Product allergy	Product type	Score
48	French Fries	None	Vegetarian, Vegan, Pescatarian	9.028360
8	Taco	None	None	8.285417
5	Meat Pizza	Gluten	None	8.238225
12	Chicken Burger	Gluten	None	8.185417
33	Wild mushroom risotto	Lactose	Vegetarian	8.035417
2	Steak Sandwich	Lactose	None	7.609005
19	ShishTawook	Gluten	None	7.559964

Table 5.29: Test case 6 Collaborative filtering

Hybrid Filtering

Product ID	Product Name	Product allergy	Product type	Score
12	Chicken Burger	Gluten	None	16.826557
8	Taco	None	None	11.949838
19	ShishTawook	Gluten	None	10.601119
2	Steak Sandwich	Lactose	None	7.394456
5	Meat Pizza	Gluten	None	7.280975
33	Wild mushroom risotto	Lactose	Pescatarian	4.100590
38	French Fries	None	None	3.057119

Table 5.30: Test case 6 Hybrid filtering

Test case 7 will represent a user without preferences and without allergies and low number of votes counts for product to be valid.

The User profile and order history are listed below:

User ID	11
Name	User11
Gender	Male
Age	21
Nationality	Spanish
Allergies	None
Preferences	No preference

Table 5.31: User profile Test case 7

Order ID	1	2	3	4	5	6	7	8
Product ID	6	42	15	39	17	25	12	16
Product name	Hot dog	Fish and Chips	Wings	Quinoa	Nuggets	Caesar salad	Chicken burger	Chicken tender
Order day	Saturday	Sunday	Wednesday	Sunday	Wednesday	Sunday	Monday	Sunday
Order date	2021-05-15	2021-05-16	2021-05-19	2021-05-23	2021-05-26	2021-05-30	2021-05-31	2021-06-06
Order time	15:19	21:20	15:00	14:00	19:30	12:00	13:00	16:30
Weather	Rainy	Cloudy	Sunny	Sunny	Sunny	Sunny	Cloudy	Sunny

Table 5.32: Order history for User 11 of test case 7

Content-based filtering

Product ID	Product Name	Product allergy	Product type	Score
18	Popcorn chicken	None	None	2.625219
17	Nuggets	None	None	2.583744
20	Chicken curry	None	None	2.353550
13	Chicken Burger	None	None	2.310922
12	Burrito	Gluten	None	2.055675
7	Calamari	Lactose	Pescatarian	1.875865
43	Schnitzel	None	None	1.826165

Table 5.33: Test case 7 Content-based filtering

Collaborative filtering

Product ID	Product Name	Product allergy	Product type	Score
48	French Fries	None	Vegetarian, Vegan, Pescatarian	9.193269
10	Shawerma	Gluten	None	8.992038
41	Dynamite shrimp	None	Pescatarian	8.728333
21	Msahab	Gluten	None	8.685127
37	Crispy vegan quinoa cakes	None	Vegetarian, Vegan	8.580657
29	Vegetarian Pasta	None	Vegetarian	8.435417
31	Cauliflower curry	Gluten	None	8.435417

Table 5.34: Test case 7 Collaborative filtering

Hybrid Filtering

Product ID	Product Name	Product allergy	Product type	Score
17	Nuggets	Gluten	None	20.071387
18	Popcorn chicken	None	None	19.448500
20	Chicken curry	Gluten	None	18.283158
13	Broasted	Lactose	None	17.454739
12	Chicken Burger	Gluten	None	16.835375
43	Calamarie	Lactose	Pescatarian	14.953529
21	Msahab	None	None	14.512310

Table 5.35: Test case 7 Hybrid filtering

Analysis test case 6 and test case 7

By observing Tables 5.26, 5.27, 5.28, 5.29, 5.30, 5.31, 5.32, 5.33, 5.34, and 5.35 we will realize that on high vote number the content-based filtering and hybrid having the same results and that due to have few products counted in the collaborative filtering, in the other hand when we have low vote number for the product to be counted and valid product for recommendation we will have better various options and better results and that's due to having a low votes on products in our database.

CHAPTER 6
Conclusions

In this work, a food recommendation system has been developed, with having highest accurate prediction as goal of this system. I have learned to integrate a machine learning system that require more than technological knowledge, a lot of research and tests. As a result, the design and architectures have been studied in greater depth.

I learnt how to design a recommendation system utilizing various recommendation methods while also grappling with issues such as which approach would be most appropriate for this system and how much redundancy to maintain, both of which were key components of the system.

A small-scale inquiry was done, and the findings were merged. This has helped me to gain a better understanding of the documentation, testing, and validation of hypothesis. Through test to validate this hypothesis, feedback of users has been asked for, to obtain the most possible accurate results to predict a product. I learned about machine learning and artificial intelligence while working on this project, a field that is leading the road to a future world with more job options.

Switching from one technology to another has been a challenge in the beginning. This has allowed me to exceed myself and improve my capacity to adapt to new technology far more quickly than previously.

Finally, because of the scope of this project, I've had to learn to manage my work on my own, which was first challenging, but now I know how to manage my time and set boundaries to organize my time.

6.1 Relationship of the work developed with the studies completed

The following are some of the aspects of the completed studies that were relevant to the completion of this project:

- **Governance of information technology, project planning and management**: The information gained in this course served as the foundation for management.

- **Information System Audit, Quality, and Management:** Design patterns and clean code have made development considerably reliable and more fluid over time.

- **High-Performance Recommender System:** Machine learning is a resource-intensive process, and being able to employ a variety of methodologies allowed me to test a variety of hypotheses in order to asses more development choices.

6.2 Future work

Recommender systems can be very powerful tools, and future developments are going to increase business value and revenue even further. As future work for this system more features are planned to be implemented such as, adding a new feature of taking in concern the amount of ingredients in each product which would help the users with allergies, that they are allergic to the percentage amount of ingredients in each product. And by filtering out only the products that are over the limits for these users, would allow them to try more options of products in restaurants.

References

[1] Watson Tech SL 2021, Accessed 2 February 2022, https://thewatsonapp.com/

[2] Sirveme online 2022, Accessed 2 February 2022, https://www.sirveme.online/wp/

[3] Pikotea software 2022, Accessed 2 February 2022, https://pikotea.com/

[4] Scrum 2022, Accessed 5 February 2022. https://www.scrum.org/resources/what-is-scrum

[5] Statista 2022, Accessed 5 February 2022. https://www.statista.com/outlook/dmo/eservices/online-food-delivery/spain

[6] Future ordering 2010, Accessed 6 February 2022. https://www.futureordering.com

[7] Brainium 2022, Accessed 6 February 2022. https://www.brainiuminfotech.com/

[8] Neptune labs 2022, Accessed 7 February 2022. https://neptune.ai/blog/graph-neural-network-and-some-of-gnn-applications

[9] Uber Technologies 2022, Accessed 7 February 2022. https://eng.uber.com/uber-eats-graph-learning/

[10] Neo4j Inc 2022, Accessed 7 February 2022. https://neo4j.com/docs/graph-data-science/current/alpha-algorithms/cosine/

[11] Melese, A. (2021 APRIL). Food and Resturant Recommendation System using Hybrid Filtering Mechanism. *NORTH AMERICAN ACADEMIC RESEARCH (NAAR)*, 281.

[12] Elsevier B.V 2022, Accessed 7 February 2022. https://www.sciencedirect.com/topics/computer-science/pearson-correlation

[13] Codecadamey 2022, Accessed 8 February 2022. https://www.codecademy.com/catalog/language/python

[14] Python software Foundation 2022, Accessed 8 February 2022. https://www.python.org/community/

[15] Snowflake Inc 2022, Accessed 8 February 2022. https://www.snowflake.com/data-warehousing-glossary/sql/

[16] Software Freedom Conservary 2021, Accessed 8 February 2022. https://git-scm.com/

[17] Atlassian 2022, Accessed 8 February 2022. https://www.atlassian.com/git/tutorials/comparing-workflows

[18] Mohanty, S. N. (2020). *Recommender System with Machine Learning and Artificial Intelligence: Practical Tools and Applications in Medical, Agricultural and Other Industries.* Hoboken: LLC.

[19] GitHub Inc 2022, Accessed 16 February 2022. GitHub Student Developer Pack - GitHub Education

[20] PayScale 2022, Accessed 17 February 2022. https://www.payscale.com/research/ES/Job=Junior_Software_Engineer/Salary

[21] Python Software Foundation 2022, Accessed 19 February 2022. https://pypi.org/project/wwo-hist/

[22] Scikitlearn 2022, Accessed 20 February 2022. https://scikitlearn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

Annexes

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Procede**
ODS 1. **Fin de la pobreza.**				**X**
ODS 2. **Hambre cero.**				**X**
ODS 3. **Salud y bienestar.**	**X**			
ODS 4. **Educación de calidad.**				**X**
ODS 5. **Igualdad de género.**	**X**			
ODS 6. **Agua limpia y saneamiento.**				**X**
ODS 7. **Energía asequible y no contaminante.**		**X**		
ODS 8. **Trabajo decente y crecimiento económico.**	**X**			
ODS 9. **Industria, innovación e infraestructuras.**		**X**		
ODS 10. **Reducción de las desigualdades**.		**X**		
ODS 11. **Ciudades y comunidades sostenibles**.				**X**
ODS 12. **Producción y consumo responsables.**	**X**			
ODS 13. **Acción por el clima.**		**X**		
ODS 14. **Vida submarina.**				**X**
ODS 15. **Vida de ecosistemas terrestres.**				**X**
ODS 16. **Paz, justicia e instituciones sólidas.**				**X**
ODS 17. **Alianzas para lograr objetivos.**				**X**

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

The food recommendation system can be related to many of the ODS objectives. The system can be useful to obtain healthy life by recommending healthy food for the user as a priority in a restaurant. That's in a hand, and in another, it helps with reducing with sharing material of restaurants such as menu, by ordering on a personal device and get recommended by that person's device. Another aspect would be reducing the discrimination, that some clients would feel mistreated by the waiters or the business owner, and by ordering online using your mobile, business owner wouldn't be able to differentiate between the clients. As well as reducing the energy that all it takes is a personal device for the user. It can be useful to increase the restaurant owner revenue by providing a way for the clients to order in a friendly and rapid way, without hesitating and long thinking of what they should order, also providing the users a modern way of ordering instead of waiting for waiters to take their orders. And by that it would also increase the economy of the country by improving the cycle money around the citizens. And provide more job opportunities by opening new restaurants based on the increase of the revenue of the restaurant owners. Which as well lead to better lifestyle for the citizens. Environmental wise, it reduces the creation of papers and that leads to reduce of cutting down trees and keep the earth a green place. Also, it would reduce the noise that made by factories of papers.