A yellow bird with a crown on a blue background

Description automatically generated

Anglia Ruskin University

Semantic Modelling for Recipe Datasets

Module Code: MOD004979 TRI1 F01CAM

Academic Year: 2024/2025

Mohammad Ali Jaber 2365662

Contents

[Introduction 4](#_Toc184803213)

[Concept and Requirement Analysis 5](#_Toc184803214)

[1. Aim of the project 5](#_Toc184803215)

[2. Gap in the knowledge 5](#_Toc184803216)

[3. List of questions 6](#_Toc184803217)

[Design 8](#_Toc184803218)

[1. Ontology Summary 8](#_Toc184803219)

[2. Classes 8](#_Toc184803220)

[3. Properties 11](#_Toc184803221)

[4. Individuals 19](#_Toc184803222)

[Requirements Mapping 23](#_Toc184803223)

[1. Mapping Table 23](#_Toc184803224)

[2. Mapping in Protégé 25](#_Toc184803225)

[Evaluation and Use 28](#_Toc184803226)

[1. Use Case 1 28](#_Toc184803227)

[2. Use Case 2 29](#_Toc184803228)

[3. Use Case 3 30](#_Toc184803229)

[4. Use Case 4 31](#_Toc184803230)

[5. Use Case 5 32](#_Toc184803231)

[Implementation 33](#_Toc184803232)

[1. Libraries and Imports: 33](#_Toc184803233)

[2. Connection to the Dataset: 35](#_Toc184803234)

[3. NLP Techniques: 37](#_Toc184803235)

[4. Overall Code: 40](#_Toc184803236)

[5. Code Validation: 42](#_Toc184803237)

[Conclusion and Future Work 48](#_Toc184803238)

[References 49](#_Toc184803239)

[Appendix 50](#_Toc184803240)

[1.1 Concept and Requirement Analysis (10%) 50](#_Toc184803241)

[1.1.1 Aim of the Project 50](#_Toc184803242)

[1.1.2 Gap in the Knowledge (Rationale) 50](#_Toc184803243)

[1.1.3 List of Questions 50](#_Toc184803244)

[1.2 Design (Semantic Model for Recipes Dataset) (25%) 50](#_Toc184803245)

[1.2.1 Ontology Summary 50](#_Toc184803246)

[1.2.2 Classes 50](#_Toc184803247)

[1.2.3 Properties 50](#_Toc184803248)

[1.2.4 Individuals 50](#_Toc184803249)

[1.3 Requirements Mapping (05%) 50](#_Toc184803250)

# Introduction

This report presents, evaluates, and critically analyses a semantic-based application for recipe datasets. Semantic Data Technologies were applied to create a machine-readable structure, establishing meaningful relationships and hierarchies in recipe data for use by cooking enthusiasts, data analysts, and general users.

The Concept and Requirement Analysis section explains the application’s idea, addressing knowledge gaps and generating classes and properties, along with key questions about ontology properties. The Design section details the ontology’s structure, including its class hierarchy, properties, and individuals, while the Requirements Mapping section shows how the ontology meets the identified needs.

The Evaluation and Use section showcase SPARQL queries and their results, demonstrating the ontology’s ability to retrieve complex data. The Implementation section outlines the tools and methods used, including SPARQLWrapper, Stanza, NLTK, and Protégé, with examples of code and dataset integration.

Through these sections, the report provides a comprehensive exploration of how semantic data technologies were applied to the recipes dataset, emphasizing their value in enhancing data organization, querying, and user interaction.

# Concept and Requirement Analysis

## Aim of the project

This report focuses on a semantic-based application designed to organize and display a recipe dataset in a structured, meaningful way. Semantic data technologies enhance recipe databases by connecting data points like cuisine types, ingredients, and nutritional values, improving organization and retrieval.

The system aims to provide users with personalized recipe access based on dietary needs, allergies, or calorie goals, offering a richer and more tailored experience. Semantic models standardize recipe data, ensuring consistency and clarity. Applications include creating meal plans for dieticians, exploring cuisines and techniques for food enthusiasts, and uncovering relationships within the dataset, such as recipes sharing ingredients or belonging to the same cuisine. Advanced search features also support queries like finding recipes for specific nutritional goals.

## Gap in the knowledge

When building the recipe ontology, knowledge gaps in food and recipes weakened the structure. Websites like bbcgoodfood.com, allrecipes.com, and taste.com provided essential information on nutrition, cooking methods, and recipes to enrich the ontology.

Analysing these websites helped identify missing concepts and relationships in the ontology. For example, as shown in Figures 1 and 2, Allrecipes provided information on occasions and cuisines, which were added as main classes in the hierarchy.

|  |  |
| --- | --- |
| A screenshot of a recipe  Description automatically generated  Figure 1-screenshot of allrecipes website | A screenshot of a recipe  Description automatically generated  Figure 2-screenshot of allrecipes website |

Also the BBC Good Food website was pivotal in shaping the recipe ontology, offering valuable insights for structuring the class hierarchy. Classes like ‘meal\_type,’ ‘nutrition,’ ‘diet\_type,’ and ‘tags’ were directly inspired by its categories. Figures 3 and 4 highlight how these concepts were derived from the platform.

|  |  |
| --- | --- |
| A screenshot of a food website  Description automatically generated  Figure 3-screenshot of bbc goodfood website | A screenshot of a food menu  Description automatically generated  Figure 4-screenshot of bbc goodfood website |

Moreover, the Taste website contributed to enhancing the ontology by suggesting new properties and classes for recipes. As shown in Figure 5, the properties ‘has\_rating,’ ‘preparation\_time,’ ‘cooking\_time,’ ‘description,’ and ‘serving\_size,’ along with the classes ‘instructions,’ ‘ingredients,’ and ‘time,’ were derived from this platform.

|  |
| --- |
| A screenshot of a recipe  Description automatically generated  Figure 5-screenshot for taste website |

## List of questions

|  |  |
| --- | --- |
| Number | Question |
| 1 | What should I make for dinner? |
| 2 | What is the rating score of “Buttermilk Pie with Gingersnap Crumb Crust” recipe? |
| 3 | What can be cooked on Christmas? |
| 4 | How many grams of chicken are needed in the Carnation Lean Fettuccine Chicken Alfredo recipe? |
| 5 | Are there any Mexican or Italian recipes? |
| 6 | What is the cuisine type of sushi? |
| 7 | What are the ingredients needed for the Crock Pot Chicken recipe? |
| 8 | What is the difficulty level of making Calzone? |
| 9 | What are the nutritional values of Cabbage Soup? |
| 10 | Which recipes have blueberry as an ingredient? |
| 11 | How long does it take to cook Biryani? |
| 12 | 20 minutes recipes? |
| 13 | Which recipes are gluten-free or sugar-free? |
| 14 | List of recipes for beginners |
| 15 | What can be cooked for 4 people? |
| 16 | Which desserts contain less than 500 calories? |
| 17 | How much protein is in Biryani? |
| 18 | Recipes created by Dancer |
| 19 | Who created the “Butterflied Lamb with Garlic Butter” recipe? |
| 20 | How much fiber does Carina’s Tofu-Vegetable Kebabs contain? |
| 21 | What can be prepared in under 45 minutes? |
| 22 | How to cook Bread Pudding? |
| 23 | Ingredients for Betty Crocker's Southwestern Guacamole Dip |
| 24 | List number of ingredients for each recipe |

# Design

## Ontology Summary

The semantic web relies heavily on formal ontologies to structure data for transferable and comprehensive machine understanding. Consequently, the success of the semantic web is closely tied to the widespread use of ontologies, which are applied in agent systems, knowledge management systems, e-commerce platforms, and many other applications to clearly define the knowledge they contain.

As shown in Figure 6 from Protégé, the ontology is made of 5998 axioms, 64 classes, 20 object properties, 14 data properties and 973 individuals.

|  |
| --- |
| A screenshot of a computer  Description automatically generated  Figure 6-ontology metrics |

## Classes

The recipe ontology’s class hierarchy includes main classes like beverages, cooking\_methods, cuisine, meal\_type, nutrition, recipe, and others, as shown in Figures 7 and 8. These classes support effective organization and retrieval of information within the dataset.

|  |  |
| --- | --- |
| A chart of food items  Description automatically generated with medium confidence  Figure 7-protégé class hierarchy | A diagram of a diagram  Description automatically generated with medium confidence  Figure 8-class hierarchy ontograf |

The ‘beverage’ class represents drinkable items that almost have the same properties as a recipe, and it is divided into ‘alcoholic’ and ‘non\_alcoholic’ sub classes. The ‘cooking\_methods’ class define different techniques for cooking, and it has ‘baking’, ‘boiling’, ‘grilling’ and ‘frying’ as sub classes, moreover, the ‘frying’ subclass contains two subclasses ‘air\_frying’ and ‘deep\_frying’.

Recipes are organized by ‘course’ class having subclasses ‘starter’, ‘main\_dish’, ‘side\_dish’ and ‘dessert’. Similarly, the ‘meal\_type’ class support categorizing recipes, it has five sub classes which are ‘breakfast’, ‘brunch’, ‘lunch’, ‘dinner’ and ‘snack’. Also, ‘cuisine’ class helps organizing recipes by representing cultural or regional origins such as Lebanese, Italian or Indian.

The ‘diet\_type’ class covers all dietary preferences or constraints such as being vegan, vegetarian, or gluten-free, while ‘flavor’ class identifies the taste of a recipe or a beverage like sweet, savoury, or spicy. Furthermore, The hierarchy for the ‘food’ class consists of many levels to provide a detailed structure, at the top level ‘ingredients’ sub class includes several branches as ‘diary’ which involve ‘cheese’, ‘milk’ and ‘yogurt’, ‘fruits’, ‘meat’ that is divides into ‘beef’, ‘lamb’, ‘pork’ and ‘poultry’, ‘seafood’, ‘spices’, ‘vegetables’ and ‘others’ which represents items that do not fit neatly into these sub classes. Also, ‘instructions’ class provides step by step guidelines for preparing for and cooking a recipe.

Class ‘location’ is a parent class of ‘country’ class which represents the origin of and geographical information relevant to a cuisine. The ‘measurments\_units’ class is responsible for ingredients and nutritional units such as grams for meat, litter for liquids, kcal for calories, cups, teaspoons etc. Meanwhile, the ‘nutrition’ class provides nutritional information about a certain recipe, it splits to ‘calories’, ‘carbohydrate’, ‘cholesterol’, ‘fat’, ‘saturated\_fat’, ‘fiber’, ‘protein’, ‘sodium’ and ‘sugar’.

Recipes are further linked to ‘occasion’ class, this connects a recipe to specific occasion or occasions because one recipe can be suitable for birthdays and Christmas for example. Additionally, ‘person’ class acts as a parent class of ‘creator’ which is a class representing creators who creates a recipe, and ‘reviewer’ class that represents users who rate or comment a recipe.

The ’recipe’ class is the central node in this ontology, it is connected to almost every other class to represent a complete dish. User feedback is captured through ‘review’ class linked directly to ‘reviewer’ class. Moreover, the ‘skill\_level’ class illustrates the difficulty level of a recipe to show if it is suitable for beginners, intermediate or advanced cooks.

To improve discoverability, the ‘tags’ class is added to the ontology to provide keywords and labels for recipes, so is ‘texture’ class which describes the physical feel of food such as crispy, crunchy, or creamy. The ‘time’ class classifies the preparation and cooking duration as well as total time for a recipe. The entire sub class hierarchy, as discussed earlier, is illustrated in Figures 9 and 10 below. These figures, captured as screenshots from Protégé, provide a detailed view of how all classes are logically organized within the ontology.

|  |  |
| --- | --- |
| A diagram of food items  Description automatically generated  Figure 9-protégé class hierarchy showing sub classes | A chart of nutrition  Description automatically generated  Figure 10-protégé class hierarchy showing sub classes |

## Properties

* 1. Object Properties

Object Properties are the next crucial components of an ontology. Class is connected to another class through an object property in the form *class\_A object\_property class\_B* where class A is the domain and class B is the range, this relationship forms a triple in the ontology.

|  |
| --- |
| A screenshot of a computer  Description automatically generated  Figure 11-WebVOWL presenting object properties |
| A diagram of a network  Description automatically generated  Figure 12-WebVOWL presenting object properties |

The two graphs above Figures 11 and 12, screenshots captured from WebVOWL website, are representing the connection between classes throughout object properties.

All twenty of the recipe ontology object properties are shown below in Figure 13 below.

|  |
| --- |
| A list of blue and black text  Description automatically generated  Figure 13-protégé object property hierarchy |

A brief explanation of the object properties:

|  |  |
| --- | --- |
| **Property** | **Protégé screenshot** |
| * belongs\_to\_cuisine: recipe belongs to a certain cuisine   + Domain: recipe   + Range: cuisine | A screenshot of a computer  Description automatically generated |
| * created\_by (inverse of: has\_recipe): recipe is created by a creator (inverse of: a creator creates a recipe)   + Domain: recipe (for has\_recipe: creator)   + Range: creator (for has\_recipe: recipe) |  |
| * has\_cooking\_method: recipe has a cooking method   + Domain: recipe   + Range: cooking\_method |  |
| * has\_course: recipe belongs to a course   + Domain: recipe   + Range: course |  |
| * has\_diet\_type: recipe and beverage have a certain diet type   + Domain: recipe or beverages   + Range: diet\_type |  |
| * has\_difficulty\_level: recipe and beverage require a certain cooking skill level   + Domain: recipe or beverages   + Range: skill\_level |  |
| * has\_flavor: recipe or beverage has a specific flavour   + Domain: recipe or beverages   + Range: flavor |  |
| * has\_ingredient (inverse of: is\_ingredient\_of): recipe or beverage consists of certain ingredients (inverse of: an ingredient belongs to a recipe or beverage)   + Domain: recipe or beverages (for is \_ingredient\_of: food)   + Range: food (for is\_ingredient\_of: recipe or beverages) |  |
| * has\_instructions: recipe and beverages have instructions and steps to follow   + Domain: recipe or beverages   + Range: instructions |  |
| * has\_meal\_type: recipe and beverage belong to a meal type/s   + Domain: recipe or beverages   + Range: meal\_type |  |
| * has\_nutrition: recipe has nutritional information   + Domain: recipe   + Range: nutrition |  |
| * has\_occasion: recipe and beverage belong to an occasion/s   + Domain: recipe or beverages   + Range: occasion |  |
| * has\_origin: cuisine has an origin   + Domain: cuisine   + Range: location or country |  |
| * has\_review: recipe and beverage have reviews (ratings and comments)   + Domain: recipe or beverages   + Range: review |  |
| * has\_tag: recipe and beverage have a tag   + Domain: recipe or beverage   + Range: tags |  |
| * has\_texture: recipe has a texture   + Domain: recipe   + Range: texture |  |
| * has\_unit: ingredients and nutrition categories have specific unit   + Domain: ingredients (+subclasses) or nutrition   + Range: measurements\_units |  |
| * reviewed\_by: review is reviewed by a reviewer   + Domain: review   + Range: reviewer |  |

* 1. Data Properties

Data properties are vital for linking individuals to literal values such as strings, numbers, or dates, thereby enriching the structure of the ontology. Figure 14 shows the hierarchy of data properties in the recipe ontology.

|  |
| --- |
| A screenshot of a computer  Description automatically generated  Figure 14-protégé data property hierarchy |

A brief explanation of the object properties:

|  |  |
| --- | --- |
| **Property** | **Protégé screenshot** |
| * description: recipe and beverages and instructions have a description * Domain: beverages or instructions or recipe |  |
| * first\_name and last\_name: properties for creator and reviewer classes (person) that represent their names. * Domain: person |  |
| * has\_rating: each review has a rating from 0 to 5 * Domain: review |  |
| * has\_comment: each review has a comment * Domain: review |  |
| * has\_quantity: ingredients (food) and nutrition have a quantity * Domain: food or ingredients or nutrition |  |
| * name: recipe, beverages, food, meal\_type, course, location, nutrition, occasion, skill\_level and measurements\_units have a name * Domain: recipe or beverages or food or meal\_type or course or location or nutrition or occasion or skill\_level or measurements\_units |  |
| * release\_date: date the recipe/beverages was released in or the date the review was posted * Domain: recipe or beverages or review |  |
| * serving\_size: the size of servings for each recipe * Domain: recipe or beverages |  |
| * time: * preparation\_time: time to prepare for a recipe * cooking\_time: time needed to cook a recipe after preparing for it * total\_time: preparation + cooking time * Domain: recipe |  |

## Individuals

Individuals are the ground level components of an ontology; they provide a real world meaning for the ontology. An individual is a specific element of a class and holds real values for its associated object and data properties.

|  |
| --- |
| A computer screen shot of a network  Description automatically generated  Figure 15-recipe class individuals ontograf |
| Figure 16-nutrition class individuals ontograf |

Two Excel sheets, one for recipes and another for reviews, were downloaded from the Food.com-Recipes and Reviews dataset on Kaggle, as shown in Figure 17 below.

|  |
| --- |
| Figure 17-kaggle food.com page |

The recipes excel sheet initially contained over 500,000 recipes organized into 28 columns. A screenshot of the updated version is shown below in Figure 18. Updates included adding a recipe ID, separating the creator's name into first and last name, and removing most records that were not needed. This process reduced the dataset to around 100 recipes, which were then inserted into the ontology using celfie Protégé 🡪 Tools 🡪 Create axioms from Excel workbook... (A used celfie rule example: Individual: @A\* Types: recipe Facts: name @B\*, preparation\_time @G\*, cooking\_time @F\*, total\_time @H\*, serving\_size @M\*, release\_date @I\*, description @J\*)

|  |
| --- |
| Figure 18-recipes excel sheet |

Similarly, the reviews excel sheet was initially huge, it contained more than 1,400,000 reviews organized into 8 columns. As shown in Figure 19 below, updated were made by adding a review ID, splitting the reviewer’s name into first and last name, and including the recipe ID to ensure clear linking of reviews to their perspective recipe when inserting the data into the ontology, as each review is associated with a single recipe. (Some used celfie rules examples:

-Individual: @C\* Types: reviewer Facts: first\_name @G\*, last\_name @H\*

-Individual: @A\* Types: review Facts: reviewed\_by @C\*, release\_date @F\*, has rating D\*, has\_comment @E\*)

|  |
| --- |
| A screenshot of a computer  Description automatically generated  Figure 19-reviews excel sheet |

Additionally, another excel sheet was created, containing multiple tabs, to add some more individuals to various classes such as nutrition, ingredients, cuisine, occasions, country… This sheet was manually prepared to be compatible with Protégé using celfie rules. Figures 20 and 21 below represents this excel sheet.

|  |
| --- |
| Figure 20-other classes excel sheet |

Some used celfie rules examples:

- Individual: @A\* Types: nutrition, @B\* Facts: has\_quantity @C\*

- Individual: @F\* Types: food, ingredients, @G\*, @H\* Facts: name @I\*

- Individual: @A\* Types: measurments\_units Facts: name @C\*

- Individual: @C\* Types: recipe Facts: has\_nutrition @A\*

|  |
| --- |
| Figure 21-other classes excel sheet |

# Requirements Mapping

## Mapping Table

|  |  |  |
| --- | --- | --- |
| **No** | **Question** | **Property** |
| 1 | What should I make for dinner? | recipe has\_meal\_type meal\_type -> dinner (object property) |
| 2 | What is the rating score of "Buttermilk Pie with Gingersnap Crumb Crust" recipe? | recipe has\_review review (object property)  review has\_rating rating (data property) |
| 3 | What can be cooked on Christmas? | recipe has\_occasion occasion (object property) |
| 4 | How many grams of chicken are needed in the Carnation Lean Fettuccine Chicken Alfredo recipe? | recipe has\_ingredient food –> ingredients (object property)  ingredients has\_unit measurements\_units (object property)  food –> ingredients has\_quantity quantity (data property) |
| 5 | Are there any Mexican or Italian recipes? | recipe belongs\_to\_cuisine cuisine (object property) |
| 6 | What is the cuisine type of sushi? | recipe belongs\_to\_cuisine cuisine (object property) |
| 7 | What are the ingredients needed for the Crock Pot Chicken recipe? | recipe has\_ingredient ingredients (object property) |
| 8 | What is the difficulty level of making Calzone? | recipe has\_difficulty\_level skill\_level (object property) |
| 9 | What are the nutritional values of Cabbage Soup? | recipe has\_nutrition nutrition (object property)  nutrition has\_unit measurements\_units (object property)  nutrition has\_quantity quantity (data property) |
| 10 | Which recipes have blueberry as an ingredient? | Food->ingredient is\_ingredient\_of recipe/beverages (object property) |
| 11 | How long does it take to cook Biryani? | recipe total\_time time (data property) |
| 12 | 20 minutes recipes? | recipe total\_time time (data property) |
| 13 | Which recipes are gluten-free or sugar-free? | recipe has\_diet\_type diet\_type (object property) |
| 14 | List of recipes for beginners | recipe has\_difficulty\_level skill\_level (object property) |
| 15 | What can be cooked for 4 people? | recipe serving\_size serving size (data property) |
| 16 | Which desserts contain less than 500 calories? | recipe has\_course course->dessert (object property)  recipe has\_nutrition nutrition->calories (object property)  nutrition has\_unit measurements\_units (object property)  nutrition->calories has\_quantity quantity (data property) |
| 17 | How much protein is in Biryani? | recipe has\_nutrition nutrition->protein (object property)  nutrition has\_unit measurements\_units (object property)  nutrition->protein has\_quantity quantity (data property) |
| 18 | Recipes created by Dancer | recipe created\_by person -> creator (object property)  creator has\_recipe recipe (object property ‘inverse of created\_by’)  person->creator first\_name name (data property)  person->creator last\_name surname (data property) |
| 19 | Who created the “Butterflied Lamb with Garlic Butter” recipe? | recipe created\_by person (object property)  person->creator first\_name name (data property)  person->creator last\_name surname (data property) |
| 20 | How much fiber does Carina's Tofu-Vegetable Kebabs contain? | recipe has\_nutrition nutrition->fiber (object property)  nutrition->fiber has\_quantity quantity (data property)  nutrition has\_unit measurements\_units (object property) |
| 21 | What can be prepared in under 45 minutes? | recipe total\_time time (data property) |
| 22 | How to cook Bread Pudding? | recipe has\_instructions instructions (object property)  Instructions description … (data property) |
| 23 | Ingredients for Betty Crocker's Southwestern Guacamole Dip | recipe has\_ingredient ingredients (object property) |
| 24 | List number of ingredients for each recipe | recipe has\_ingredient ingredient (object property) |

## Mapping in Protégé

|  |  |
| --- | --- |
| +r2 is a recipe individual  +r2 has\_review rev73  +when clicking on rev73 |  |
| +rev73 is a review individual  +rev73 reviewed\_by 3575  --data properties:  +rev73 has\_rating 5  +rev73 has\_comment Thanks…  +rev73 release\_date 2000…  +when clicking on 3575 |  |
| +3575 is a reviewer individual  --data properties:  +3575 first\_name Lynn  +3575 last\_name Walters |  |
| +r2 has\_meal\_type m1  +when clicking on m1 |  |
| +m1 is a meal\_type individual  --data properties:  +m1 name dinner |  |
| +r2 has\_nutrition n18  +when clicking on n18 |  |
| +n18 is a protein individual  +n18 has\_unit u1  --data properties:  +n18 has\_quantity 280.1  +when clicking on u1 |  |
| +u1 is a measurements\_units individual  --data properties:  +u1 name gram |  |
| +r2 created\_by 1567  +when clicking on 1567 |  |
| +1567 is a creator individual  --data properties:  +1567 first\_name elly9812 |  |
| +r15 is a recipe individual  --data properties:  +r15 name Betty Crocker…  +r15 serving\_size 4  +r15 description Make and…  +r15 release\_date 1999…  +r15 preparation\_time 5M  +r15 cooking\_time 2H  +r15 total\_time 2H 5M |  |

# Evaluation and Use

## Use Case 1

* Question: How many grams of chicken are needed in the Carnation Lean Fettuccine Chicken Alfredo recipe?

A white screen with text

Description automatically generated with medium confidence

Figure 22-SPARQL query 1 and result

* Explanation: This SPARQL query retrieves information about recipes, including their names, ingredients, the names and quantities of those ingredients, and the measurement units associated with them. The FILTER starts searching for a recipe with a name matching “Fettuccine Chicken Alfredo” and ensures that one of its ingredients contains the name “chicken” as an ingredient name, with the “i” flag in REGEX ignoring case sensitive comparison. The && condition is applied in this FILTER to combine these two requirements for precise results.

## Use Case 2

* Question: Are there any Mexican or Italian recipes?

A screenshot of a computer

Description automatically generated

Figure 23-SPARQL query 2 and result

* Explanation: This SPARQL query retrieves information about recipes and their associated cuisines, specifically their names. The FILTER checks if the cuisine's name is either "Mexican" or "Italian," using the LCASE function to ensure a case-insensitive comparison. The results are ordered in descending order based on the cuisine name for better readability.

## Use Case 3

* Question: List number of ingredients for each recipe

A screenshot of a computer

Description automatically generated

Figure 24-SPARQL query 3 and result

* Explanation: This query fetches the names of recipes along with the count of their ingredients. The COUNT function is used to count the number of ingredients of each recipe. The GROUP BY modifier ensures that the count of ingredients is grouped by each recipe name, providing a list of recipe name and the number of ingredients associated to it.

## Use Case 4

* Question: How much protein is in Biryani?

A white rectangular object with black lines

Description automatically generated

Figure 25-SPARQL QUERY 4 AND RESULT

* Explanation: This SPARQL query fetches information about the protein content in the recipe “Biryani.” It extracts the recipe name, the specific nutrition type (protein), the quantity of protein, and the measurement unit. The FILTER condition ensures that only the recipe with a name matching "Biryani" (case-insensitive after using LCASE) is included in the results.

## Use Case 5

* Question: How to cook Bread Pudding?

A screenshot of a computer

Description automatically generated

Figure 26-SPARQL QUERY 5 AND RESULT

* Explanation: This query retrieves the cooking instructions for the recipe “Bread Pudding”. It extracts recipes’ names and description of the instructions associated with the recipe. The FILTER condition uses REGEX to match the recipe name containing “bread pud” ignoring case sensitivity (“i”).

# Implementation

## Libraries and Imports:

#nltk.download('averaged\_perceptron\_tagger')

averaged\_perceptron\_tagger is downloaded to help with POS (part of speech) tagging.

#nltk.download('maxent\_ne\_chunker')

maxent\_ne\_chunker is named entity identifier.

#nltk.download('words')

From NLTK ‘words’ is a word corpus that contains commonly used English words.

#nltk.download('punkt')

punkt is a tokenizer it is essential for breaking down a text into words.

#nltk.download('stopwords')

NLTK’s stopwords is a list of common stop words is English; it is used to remove unnecessary words from a text.

#nltk.download('wordnet')

wordnet is a lexical database which provides synonyms, definitions, and relations between words.

#pip install SPARQLWrapper

The SPARQLWrapper library is commonly used to interact with SPARQL endpoints.

#pip install stanza

The stanza library is an NLP toolkit in Python, it is useful for processing natural language queries.

#stanza.download('en')

Downloading the English language model from the stanza library because queries used are in English language.

#from SPARQLWrapper import SPARQLWrapper2

SPARQLWrapper2 is an improved version of SPARQLWrapper, it is used to connect to the recipe dataset.

#from nltk.tokenize import word\_tokenize

word\_tokenize is a function from the NLTK library, it is used to split a sentence into words or tokens.

#from nltk. stem import WordNetLemmatizer

WordNetLemmatizer is a function from NLTK library, it is used to return a word to its base root (lemma).

#from nltk.corpus import stopwords

Importing the stopwords after downloading it.

#import stanza

Importing the stanza library after downloading it.

#import re

re is a Python built in module that is used to normalize a text.

#from string import punctuation

Importing punctuation from Python’s string which contains all standard punctuations characters; it is used to remove all unnecessary punctuation characters from a text.

|  |
| --- |
| A screenshot of a computer  Description automatically generated  Figure 27- Jupyter notebook code snippet |

## Connection to the Dataset:

To demonstrate the connection to the dataset, below (Figure 28) is a screenshot from the Apache Fuseki server which shows the server status as green, confirming that the server is running successfully. It also displays the URL of the recipe dataset, which verifies that the SPARQL queries are being executed against the correct endpoint. This URL was copied from the “edit” tab on the Fuseki page under “List Current Graphs” 🡪 “default” as shown in Figure 29 down.

|  |
| --- |
| A screenshot of a computer  Description automatically generated  Figure 28-apache fuseki server query tab |
| A screenshot of a computer  Description automatically generated  Figure 29-apache fuseki server edit tab |

Figure 30 down displays a code snippet from Jupyter Notebook demonstrating how the connection to the dataset was established using SPARQLWrapper2, and then sending the query to SPARQL end point.

|  |
| --- |
| A screenshot of a computer program  Description automatically generated  Figure 30-Jupyter notebook code snippet |

## NLP Techniques:

1. Tokenization:

Tokens are obtained using the NLTK tokenize library’s function: word\_tokenize

tokens = word\_tokenize(text) #this line is used in get\_lemmas function.

1. Lemmatization:

WordNetLemmatizer is used to identify lemmas, it is placed in the function below

get\_lemmas function:

{

def get\_lemmas(text):

text = normalize(text)

lemma = WordNetLemmatizer()

tokens = word\_tokenize(text)

clean\_tokens = remove\_stop\_words(tokens)

return [lemma.lemmatize(token, 'v') for token in clean\_tokens]

}

This function calls the normalize function and stores the result in “text” variable 🡪 “lemma” variable stores results from WordNetLemmatizer 🡪 “tokens” variable stores tokens from word\_tokenize 🡪 remove stop words and store them in “clean\_tokens” variable 🡪 lemmatize each token in clean\_tokens then returns the lemmatized form

1. Stopword Removal:

remove\_stop\_words function:

{

def remove\_stop\_words(tokens):

#Remove common stopwords

stop\_words = stopwords.words("english")

stop\_words.append('list') # Add custom stopwords if needed

stop\_words.remove('by') # Remove custom stopwords if needed

return [word for word in tokens if word not in stop\_words]

{

The function above takes tokens as an input 🡪 add the word “list” to the stopwords list 🡪 remove “by” word from the stopwords list (it is used to identify properties) 🡪 looks for a match in the NLTK’s stopwords list 🡪 returns words that are not a match

1. Named Entity Recognition (NER):

First process the input “text” through the stanza pipeline and store the results in “doc” variable

{

nlp = stanza.Pipeline(“en”, download\_method=stanza.DownloadMethod.NONE)

doc = nlp(text)

}

Then search for entities using hasattr() condition as shown below:

{

entities\_arr = []

if hasattr(doc, 'entities') and len(doc.entities) > 0:

entity = doc.entities[0].text

entities\_arr.append(entity)

new\_text = text.replace(entity, '')

else:

new\_text = text

}

The named entities (if any found) are stored in “entities\_arr” variable.

Then a bag of words list was created to identify other named entities that could be used in our queries.

{

entities\_BOW = ['beginner', 'bread', 'pudding']

for word in lemmatized\_words:

if word in entities\_BOW:

entities\_arr.append(word) # Keep the term itself for SPARQL filtering

}

1. Text Normalization:

re.sub and string.punctuation were used to remove punctuation, special characters, and extra spaces from the input text.

The function “normalize” in the code takes a text (string) as an input 🡪 remove the found punctuations or other characters by substituting it with “” 🡪 then returning the cleaned text ensuring that only one single space is between words by using

“ “.join(processed\_text.split())

-normalize function:

{

def normalize(text):

#Remove punctuation and extra spaces

processed\_text = re.sub(f"[{re.escape(punctuation)}]", "", text)

print(processed\_text)

return " ".join(processed\_text.split())

}

1. Ontology Term Mapping:

Two “bag of words” lists were created to ensure accurate mapping between the user’s input question and the recipe dataset. These lists contain potential keywords related to properties and classes within the ontology. The purpose is to identify the relevant classes or properties by comparing the words in the question against the keywords in the lists.

{  
 prop\_list = [

['belongs\_to\_cuisine', 'cuisine'],

['created\_by', 'created', 'creator', 'by'],

['has\_difficulty\_level', 'skill\_level', 'level', 'difficulty', 'beginner', 'advanced'],

['has\_ingredient', 'food', 'ingredient', 'ingredients', 'contain'],

['has\_instructions', 'instructions', 'how to make'],

['has\_nutrition', 'nutrition', 'calories', 'protein'],

['has\_diet\_type', 'diet', 'allergen', 'allergies', 'vegan', 'veg', 'gluten']

]

class\_list = [

['recipe', 'recipes', 'cook'],

['food', 'ingredient', 'chicken', 'fish', 'beef', 'poultry'],

['measurments\_units', 'unit', 'gram', 'kcal'],

['cuisine', 'origin'],

['skill\_level', 'difficulty level', 'skill level', 'hard', 'easy', 'beginner', 'beginners']

]

}

Then the function match\_property\_and\_class below is required for searching for a match and returning matched classes and matched properties.

{

def match\_property\_and\_class(lemmatized\_words, prop\_list, class\_list):

#Match lemmatized words to properties and classes

prop\_dict = {alias: prop[0] for prop in prop\_list for alias in prop}

class\_dict = {alias: cls[0] for cls in class\_list for alias in cls}

matched\_properties = []

matched\_classes = []

for word in lemmatized\_words:

if word in prop\_dict:

matched\_properties.append(prop\_dict[word])

if word in class\_dict:

matched\_classes.append(class\_dict[word])

# Remove duplicates, if any

matched\_properties = list(set(matched\_properties))

matched\_classes = list(set(matched\_classes))

return matched\_properties, matched\_classes

}

## Overall Code:

1. The user’s question is read using input() and stored in “text” variable

{

text = input("What is your question? ")

}

1. After preprocessing the text using the techniques discussed in section 3 above (NLP Techniques), three results are obtained:

matched\_properties storing the matched properties between text and prop\_list

matched\_classes storing the matched classes between text and class\_list

entities\_arr storing the named entities

1. These results are used in nested if else conditions to obtain the best query possible as shown in Figure 31 below.

|  |
| --- |
| A screenshot of a computer  Description automatically generated |
| A screen shot of a computer code  Description automatically generated  Figure 31-Jupyter notebook code snippet |

The above code dynamically generates SPAQRL queries based on the user’s input, aiming to match classes, properties, and entities identified through NLP processing.

* + 1. First condition (if matched\_classes and matched\_properties and len(entities\_arr)>0)

This condition applies when a class and a property and an entity at least are found during processing.

🡺 query retrieves instances of the matched class type

* Relation between this instance and another variable is found using the matched property
* FILTER condition is applied on the found entity
  + 1. Second condition (if matched\_classes and len(entities\_arr)>0)

This condition applies when at least one class and one entity are found.

* Query retrieves instances of the matched class
* Looks for relation between this instance and another instance through an unknown property
* FILTER condition is applied on the found entity
  + 1. Third condition (if matched\_classes and matched\_properties)

This condition is applied when at least one class and one property are found.

* Query retrieves instances of the matched class
* Relation between this instance and another variable is found using the matched property
  + 1. Fourth condition (if matched\_classes 🡪 if len(matched\_classes) < 2)

This condition is applied when only one class is found.

* Query retrieves instances of the matched class
* Looks for relation between this instance and another instance through an unknown property
  + 1. Fifth condition (else of (if len(matched\_classes) < 2) condition)

This condition is applied when more than one class is found.

* Query retrieves instances of the matched classes (two instances from two classes)
* Looks for a relation between these two instances through an unknown property
  + 1. Sixth condition (else of (if matched\_classes) condition)

This condition applies when only property is found.

* Query looks for a relation between two instances through the matched property
  + 1. Last condition is applied when no classes or properties or entities are found

🡪it displays an error message: “Sorry no queries found”

## Code Validation:

1. Question 1:

“list all Italian recipes”

The input will be ‘lisT all "Italian – recipes’

|  |
| --- |
|  |
| A screenshot of a computer  Description automatically generated |
| A screenshot of a computer  Description automatically generated |
| A screenshot of a computer  Description automatically generated |
| A screenshot of a computer  Description automatically generated |

Figure 32-Jupyter notebook code snippet

1. Question 2:

“list all the recipes that contain Blueberry”

The input will be ‘List\_\_all recipes that contain Blueberry’

|  |
| --- |
|  |
|  |
| A screenshot of a computer  Description automatically generated  Figure 33-Jupyter notebook code snippet |

1. Question 3:

“what recipes are for beginner cook?”

|  |
| --- |
|  |
|  |
| A screenshot of a computer  Description automatically generated  Figure 34-Jupyter notebook code snippet |

1. Question 4

“what are the ingredients of Crock Pot Chicken recipe?”

|  |
| --- |
|  |
|  |
| Figure 35-Jupyter notebook code snippet |

1. Question 5:

“show the diets for each recipes”

The input will be ‘show the diets for @@@each recipes!!!!!!!!’

|  |
| --- |
|  |
|  |
| Figure 36-Jupyter notebook code snippet |

# Conclusion and Future Work

This project demonstrates the practical application of semantic data technologies. This model aims to enhance the searchability and usability of recipe dataset.

Future work on this project can focus on several improvements such as expanding the ontology to include additional classes and properties and enhancing the natural language processing pipeline with more advanced models that can improve the understanding of complex user queries. Additionally, creating a graphical user interface can be something to investigate later to provide non-technical users to interact with the system.

# References

* BBC Studios Distribution Limited (2024) *goodfood*. Available at: <https://www.bbcgoodfood.com/> (Accessed: 15 November 2024).
* Dotdash meredith (2024) *allrecipes*. Available at: <https://www.allrecipes.com/> (Accessed: 17 November 2024).
* Taste.com.au (2024) *Real beef burritos*. Available at: <https://www.taste.com.au/recipes/real-beef-burritos/e8157364-323d-44ac-af59-be3bad827ba9?r=recipes/burritorecipes&c=odqrvog5/Burrito%20recipes> (Accessed: 29 November 2024).
* Maedche, A. and Staab, S., 2001. Ontology learning for the semantic web. *IEEE Intelligent systems*, *16*(2), pp.72-79.
* Gómez-Pérez, A. and Corcho, O., 2002. Ontology languages for the semantic web. *IEEE Intelligent systems*, *17*(1), pp.54-60.
* Kaggle (2020) *Food.com - Recipes and Reviews*. Available at: <https://www.kaggle.com/datasets/irkaal/foodcom-recipes-and-reviews?select=recipes.csv> (Accessed at: 27 November 2024)

# Appendix

This Appendix includes the Group Work part of Kamilakhon’s (2363748) report.

## Concept and Requirement Analysis (10%)

### Aim of the Project

To implement the given knowledge of Semantic Data basics principles is the aim of project. Moreover, the key goal is to create the ontology for recipes database which will be used to organize information about recipes. The ontology will simplify search and increase the efficiency of the given output.

### Gap in the Knowledge (Rationale)

It is complicated to determine the exact number of recipes in the world, the data related to this topic can be described as a Big Data. As it stated by Cambridge dictionary, Big Data is a huge amount of existing information created and used by people, that is difficult to manage in traditional ways. Moreover, a food topic is never going to lose its relevance because it is a key part of daily human needs.

This trend can be seen in the Google statistics (Graph 1). The given plot illustrates the percentage of interest in search queries related to the word "recipes" around the world over the past year. Based on the graph, the index of interest remains over 70% throughout the year 2024, indicating the significance of this field (Google, 2024).

The project would help millions of people to find the necessary information by making outputs much more accurate and relevant, covering all specificities of each request.

A blue line graph on a white background

Description automatically generatedGraph 1: Google trend of interest in search "recipes" queries

### List of Questions

The table 1 presents a list of possible popular questions related to this topic, which will be used in “Evaluation and Use” section.

|  |  |
| --- | --- |
| No. | Questions |
| 1 | What should I make for dinner? |
| 2 | What is the rating of "Buttermilk Pie with Gingersnap Crumb Crust"? |
| 3 | What can be cooked on Christmas? |
| 4 | How many grams of chicken are needed in the Carnation Lean Fettuccine Chicken Alfredo recipe? |
| 5 | Are there any Mexican or Italian recipes? |
| 6 | What is the cuisine type of sushi? |
| 7 | What are the ingredients needed for the Crock Pot Chicken recipe? |
| 8 | What is the difficulty level of making Calzone? |
| 9 | What are the nutritional values of Cabbage Soup? |
| 10 | Which recipes have blueberry as an ingredient? |
| 11 | How long does it take to cook Biryani? |
| 12 | What can be prepared in 20 minutes? |
| 13 | Which recipes are gluten-free or sugar-free? |
| 14 | List of recipes for beginners |
| 15 | What can be cooked for 4 people? |
| 16 | Which desserts contain less than 500 calories? |
| 17 | How much protein is in Biryani? |
| 18 | Recipes created by Dancer |
| 19 | Who created the “Butterflied Lamb with Garlic Butter” recipe? |
| 20 | How much fiber is contained in Carina’s Tofu-Vegetable Kebabs? |
| 21 | What can be prepared in under 45 minutes? |
| 22 | How to cook Bread Pudding? |
| 23 | Ingredients for Betty Crocker’s Southwestern Guacamole Dip |
| 24 | Recipes for 5 or more people |

Table 1: List of Questions

## Design (Semantic Model for Recipes Dataset) (25%)

The primary goal of Semantic Web Technologies is to enhance the efficiency of interactions between humans and machines when working with various types of web information. By applying ontology principles, information can be transformed into a structured database, enabling machines to display it effectively and automatically understand the underlying concepts. Ontology helps to identify relationships and meanings between data by creating a structured hierarchy including classes, axioms, properties and etc (Jain and Singh, 2013).

Protégé is the most popular and free software to design ontology that includes some representation languages such as OWL (Stanford Center, 2019). Protégé was used as a main tool to develop Recipe Dataset ontology.

### Ontology Summary

A screenshot of a computer

Description automatically generated

Figure 1: Recipes Dataset Ontology Overview

As illustrated in Figure 1, the Recipes Dataset Ontology consists of 64 classes, 20 of which are main, along with 5998 axioms, 973 individuals, and 34 properties. The ontology represents a hierarchical structure of domain’s entities, its levels, and relationships or attributes. Moreover, it also includes individuals that allow testing the constraints, axioms and relationships in the ontology. In order to build this model, several examples of websites such as Taste or Allrecipes were considered as well.

### Classes

A diagram of food items

Description automatically generatedA chart of nutrition

Description automatically generated

Figure 2: Class hierarchy

The class hierarchy of the Recipes ontology is composed of 20 classes and 44 subclasses. In addition, some of subclasses such as Frying and Ingredients include also own subclasses.

Figure 2 shows that main classes are *beverages, cooking\_methods, course, cuisine, diet\_type, flavour, food, instructions, location, meal\_type, measurements\_units, nutrition, occasion, person, recipe, review, skill\_level, tags, texture*, and *time*.

The class representation corresponds to its title. For example, *occasion* class includes a list of worldwide holidays such as Christmas or Navruz. (Having that class would be eiser to find appropriate recipes for specific events.)

The *location* class represents the list of *countries* used to identify the origins of recipes or food. However, the *cuisine* class has a similar concept, it was decided to create it as a separate class because the *cuisine* **is not** a *location* or a *country*, making it impossible to include it as a subclass.

A diagram of a nutrition

Description automatically generated

Figure 3: The *Nutrition* class OntoGraf

Another class is *nutrition* that consists of several subclasses as *calories, fat* or *saturated\_fat, carbohydrate, cholesterol, fiber, protein, sodium* and *sugar.* Each subclass is representation of nutrition value of recipes.

*Dessert, main\_dish, side\_dish, starter* classes are considered to be subclasses of the *course* class. Similarly, *breakfast, dinner, brunch, snack* and *lunch* are subclasses of the *meal\_type* class, representing categories related to the classes’ topic.

Furthermore, recipes can be rated and have comments or feedbacks which would be inside of the *review* class. People who comment and rate recipes are added into the *person* class as well as authors of recipes. Therefore, there are two subclasses of *person* class: *creator* and *reviewer*.

A diagram of food ingredients

Description automatically generated

Figure 4: The *Food* class OntoGraf

The biggest class in the hierarchy is *food* class which describes various ingredients. The *ingredients* subclass involves different groups of food products such as *dairy, meat, fruits, vegetables, seafood, spices, and others* that can be identified as sub-subclasses. In addition, some of them includes their own subclasses.

### Properties

There are two types of properties: object properties and data properties. Object properties represent relationships between classes that are linked to each other. In ontology, they generate triples that contain domains and ranges. Figure 5 shows a list of 20 object properties used in the project.

A list of food items

Description automatically generated

Figure 5: Object properties

For instance, the object property (*has\_unit*) includes some classes: *dairy, food, fruits, ingredients, meat, nutrition, seafood, spices, time, vegetables* as domains with *measurements\_units* as the range of this triple.

A screenshot of a computer

Description automatically generated

Figure 6: Description of the (*has\_unit*) object property

Most of the object properties have *recipe* and *beverages* classes in an intersection role, such as (*has\_flavor), (has\_diet\_type), (has\_difficulty\_level), (has\_ingredient), (has\_instructions), (has\_meal\_type), (has\_occasion), (has\_review), (has\_tag).*

There are (created­\_by) and (has\_ingredient) properties that have inverse versions shown below.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Figure 7: Description of the (created­\_by */ has\_ingredient*) object properties

In other words, classes can be both a subject and an object at the same time. This is clearly shown in the Figure 8.

A diagram of a course

Description automatically generated

Figure 8: Relationships between the *recipe* class and others

Data properties represent relationships between entities and literal values or data. There are 14 data properties, including sub-properties in the ontology. The data properties were titled according to their meanings (Figure9).

The *person* class is a domain for *(first\_name)* and *(last\_name)* data properties. It would keep personal information about authors and reviewers such as surnames and names. The *review* class has (*has\_rating/has\_comment*) as data properties. It will contain the content of reviews and given rating score.

A screenshot of a computer

Description automatically generated

Figure 9: Data properties

Data properties as object properties can have several domains. The data property (*name*) has 9 classes as intersections.

A screenshot of a computer

Description automatically generated

Figure 10: Description of (*name*) data property

The data property (*serving\_size*) represents a portion size of a selected dish by identifying the number of servings. The domains of the property are *beverages* and *recipe* classes while the range is *xsd:integer.*

A screenshot of a computer

Description automatically generated

Figure 10: Description of (*serving\_size*) data property

### Individuals

Another important component of the ontology are individuals, which represent real data. Approximately 1,000 individuals were imported into this ontology. Some Excel files containing necessary information were sourced from the Internet, while others were drafted manually.

A screenshot of a website

Description automatically generated

Figure 11: The source of dataset

For example, Reviews Excel file includes information about ID for recipes/reviews/reviewers, comments’ content, rating scores, reviewers’ names and surnames and etc.

A screenshot of a computer

Description automatically generated

Figure 12: Reviews Excel file

Before importing the data from Excel sheets, it is important to clean the data. In this case, reviewers’ personal data were divided into two columns: Reviewer\_first\_name and Reviewer\_last\_name. There were used two methods for adding individuals into the ontology. The first method involved importing data using Cellfie by writing transformation rules. Figure 13 illustrates rules examples that were applied to import data from the Reviews Excel table.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Figure 13: Rules for Reviews Excel file

Additionally, the created individuals are categorized into the appropriate class, and their properties are established when outlining the facts.

A screenshot of a computer

Description automatically generated Figure 14: Individuals of *review* class

The second method of adding individuals was used in the process of linking required ingredients to recipes. Due to the lack of specific items in the *ingredient* subclass, they were added manually and connected to related recipes. (Figure 15)

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Figure 15: Screenshots of adding individuals manually

In Figure 16, there are individuals and relations of *recipe* class. Individuals of recipe class consist of data properties like (*name), (serving\_size), (total\_time), (cooking\_time), (preparation\_time)* and *(description).* They also include object properties which describe nutrition values, cuisine and occasion belongings, used ingredients, authors, reviews and etc.

A diagram of a diagram

Description automatically generated with medium confidence

Figure 16: The OntoGraf of *recipe* class

In Figure 17 can be seen all individuals belonging to food class and its relations to other subclasses. It means that individuals are sorted by types of food products, for example: fruits, vegetables, type of meat and etc.

A screenshot of a computer screen

Description automatically generated

Figure 17: The OntoGraf of *food* class

The individual of *review* class has data properties as (*release\_date), (has\_comment) and (has\_raiting).* It also displays the author of review linked by (*reviewed\_by*) object property. (Figure 18)

A screenshot of a computer

Description automatically generated

Figure 17: The Individual of *review* class

There are more graphs in the Appendix section.

## Requirements Mapping (05%)

Mapping is a great way to understand the logical chain of how effectively the created ontology will be used to process user requests. There is a table (Table 2) below that reveals the necessary elements for getting response from all the questions created in first task 'Concept and Requirement analysis'.

The key words were marked to increase understanding.

|  |  |
| --- | --- |
| **Question** | **Mapping** |
| What to cook for dinner? | The connection between 2 classes such as Recipe and Meal\_type, by an object property (**has\_meal\_type),** can be an answer to this question  recipe **has\_meal\_type** meal\_type -> dinner  (object property) |
| What is the rating for  "Buttermilk Pie with Gingersnap Crumb Crust" recipe? | This case can be answered using Recipe and Review classes connected by object property (**has\_review**), and  data property (**has\_rating**) of Review class  recipe **has\_review** review (object property)  review **has\_rating** rating (data property) |
| What can be cooked on Christmas? | The request can be answered by object property (**has\_occasion**) between Recipe and Occasion classes  recipe **has\_occasion** occasion (object property) |
| How many grams of chicken are needed in the Carnation Lean Fettuccine Chicken Alfredo recipe? | It is necessary to use:  an object property (**has\_ingredient**) that connects Recipe and Ingredients classes; object property **(has\_unit**) for Ingredients, Measurements\_units classes; data property (**has\_quantity**) for Food class  recipe **has\_ingredient** food –> ingredients  (object property)  ingredients **has\_unit** measurements\_units  (object property)  food –> ingredients **has\_quantity** quantity  (data property) |
| Are there any Mexican or Italian recipes? | This question can be answered using Recipe and Cuisine classes linked by object property (**belongs\_to\_cuisine**)  recipe **belongs\_to\_cuisine** cuisine (object property) |
| What is the cuisine type of sushi? | The request can be answered using Recipe and Cuisine classes linked by object property (**belongs\_to\_cuisine**)  recipe **belongs\_to\_cuisine** cuisine (object property) |
| What are the key ingredients for the Crock Pot Chicken recipe? | The connection between Recipe and Ingredients classes by an object property (**has\_ingredient**) is an answer to this question.  recipe **has\_ingredient** ingredients (object property) |
| What is the difficulty level of making Calzone? | In order to get response to this question, it is necessary to utilize classes such as Recipe and Skill\_level connected by an object property (**has\_difficulty\_level**)  recipe **has\_difficulty\_level** skill\_level  (object property) |
| What is the nutritional value of a Cabbage Soup? | To get an answer to this question, Recipe, Nutrition and Measurements\_units classes with object properties **(has\_nutrition/ has\_unit)** and a data property **(has\_quantity)** are should be taken.  recipe **has\_nutrition** nutrition (object property)  nutrition **has\_unit** measurements\_units  (object property)  nutrition **has\_quantity** quantity  (data property) |
| Which recipe uses the blueberries as main ingredients? | The request can be answered using Recipe, Beverages, Food(ingredients) classes and inverse function (**is\_ingredient\_of**) of object property (**has\_ingredient**) that links that classes  Food->ingredient **is\_ingredient\_of** recipe/beverages (object property) |
| How long does it take to cook Biryani? | An answer to this query is utilization of Recipe class and data property  (**time**->**total\_time**)  recipe **total\_time** time (data property) |
| What can be prepared in 20 minutes? | To get a response to this question, Recipe class and data property (**time**->**total\_time**) need to be used  recipe **total\_time** time (data property) |
| Which recipes are gluten-free or sugar-free? | The linked Recipe and Diet\_type classes by an object property (**has\_diet\_type**) is the answer to this question.  recipe **has\_diet\_type** diet\_type (object property) |
| List of recipes for beginners | This request can be answered by object property (**has\_difficulty\_level**) between Recipe and Skill\_level classes  recipe **has\_difficulty\_level** skill\_level  (object property) |
| What can be cooked for 4 people? | This question can be answered by usage of Recipe class and data property (**serving\_size**)  recipe **serving\_size** servingsize (data property) |
| Which desserts contain less than 500 calories? | Recipe, Course, Nutrition and Measurements\_units classes with object properties **(has\_course/has\_nutrition/ has\_unit)** and a data property **(has\_quantity)** are should be taken to answer the question  recipe **has\_course** course->dessert  (object property)  recipe **has\_nutrition** nutrition->calories  (object property)  nutrition **has\_unit** measurements\_units  (object property)  nutrition->calories **has\_quantity** quantity  (data property) |
| How much protein is in Biryani? | Recipe, Nutrition and Measurements\_units classes with object properties **(has\_nutrition/has\_unit)** and a data property **(has\_quantity)** are should be used to answer the question  recipe **has\_nutrition** nutrition->protein  (object property)  nutrition **has\_unit** measurements\_units  (object property)  nutrition->protein **has\_quantity** quantity  (data property) |
| Recipes created by Dancer | The request can be answered by object property (**created\_by**) between Recipe and Person (Creator) classes and data properties (**last\_name/ first\_name**)  recipe **created\_by** person (object property)  person->creator **first\_name** name (data property)  person->creator **last\_name** surname (data property) |
| Who is the author of the “Butterflied Lamb with Garlic Butter” recipe? | The connection between 2 classes such as Recipe and Person (Creator) by object property (**created\_by**) and data properties (**last\_name/ first\_name**)  recipe **created\_by** person (object property)  person->creator **first\_name** name (data property)  person->creator **last\_name** surname (data property) |
| How much fiber is contained in Carina's Tofu-Vegetable Kebabs? | Recipe, Nutrition and Measurements\_units classes with object properties **(has\_nutrition/has\_unit)** and a data property **(has\_quantity)** are should be used to answer the question.  recipe **has\_nutrition** nutrition->fiber  (object property)  nutrition **has\_unit** measurements\_units  (object property)  nutrition->fiber **has\_quantity** quantity  (data property) |
| What can be prepared in under 45 minutes? | Recipe class and data property (**time**->**total\_time**) need to be used to get a response to this question  recipe **total\_time** time (data property) |
| How to cook Bread Pudding? | This request can be answered by using Recipe class and data property (**description**)  recipe **description** description (data property) |
| Ingredients for Betty Crocker's Southwestern Guacamole Dip | The request can be answered using Recipe, Ingredients classes and object property (**has\_ingredient**) that links that classes  recipe **has\_ingredient** ingredients (object property) |
| Recipes for 5 or more people | This question can be answered by usage of Recipe class and data property (**serving\_size**)  recipe **serving\_size** servingsize (data property) |

Table 2: Requirement Mapping