

# **AN ONTOLOGY-BASED RESTAURANT INFORMATION SYSTEM**

A REPORT SUBMITTED TO THE UNIVERSITY OF MANCHESTER  
FOR THE DEGREE OF BACHELOR OF SCIENCE  
IN THE FACULTY OF SCIENCE AND ENGINEERING

2023

Thomas Hayter

Department of Computer Science

# Contents

<b>Abstract</b>	<b>7</b>
<b>Declaration</b>	<b>8</b>
<b>Copyright</b>	<b>9</b>
<b>Acknowledgements</b>	<b>10</b>
<b>1 Introduction</b>	<b>11</b>
1.1 Overview . . . . .	11
1.2 This Report . . . . .	12
1.3 Project Plan . . . . .	12
<b>2 Background</b>	<b>14</b>
2.1 Diets . . . . .	14
2.2 Ontologies . . . . .	15
2.2.1 Object and Data Properties . . . . .	16
2.2.2 Quantifiers . . . . .	17
2.2.3 Defined Classes . . . . .	18
2.2.4 Instances . . . . .	18
2.2.5 Enumerations . . . . .	18
2.2.6 Open World Assumption . . . . .	18
2.3 Technologies . . . . .	19
2.3.1 Protégé . . . . .	19
2.3.2 The OWL API . . . . .	19
2.3.3 Java Swing . . . . .	20
<b>3 Design &amp; Methodology</b>	<b>21</b>
3.1 Requirements . . . . .	21

3.1.1	Functional Requirements . . . . .	21
3.1.2	Non-Functional Requirements . . . . .	22
3.1.3	Dietary Requirements . . . . .	22
3.2	System Design . . . . .	23
3.2.1	Ontology Design . . . . .	23
3.2.2	Restaurant UI Design . . . . .	25
3.2.3	Customer UI Design . . . . .	27
3.2.4	Menu . . . . .	29
3.3	Evaluation Plan . . . . .	30
3.3.1	Ethics . . . . .	31
<b>4</b>	<b>Implementation</b>	<b>32</b>
4.1	Study 1 . . . . .	32
4.1.1	Customer UI Study . . . . .	32
4.1.2	Restaurant UI Study . . . . .	33
4.1.3	Interpreting the Results . . . . .	33
4.2	Deliverables . . . . .	34
4.2.1	Ontology . . . . .	34
4.2.2	Restaurant User Interface . . . . .	35
4.2.3	Customer User Interface . . . . .	37
4.2.4	OntologyManager . . . . .	39
4.3	Design Choices . . . . .	40
<b>5</b>	<b>Results</b>	<b>42</b>
5.1	Study 2 . . . . .	42
5.1.1	Customer UI Study . . . . .	42
5.1.2	Restaurant UI Study . . . . .	43
5.1.3	Interpreting the Results . . . . .	43
<b>6</b>	<b>Summary &amp; Outlook</b>	<b>44</b>
6.1	Did I Meet My Objectives? . . . . .	44
6.2	Improvements . . . . .	45
6.3	Conclusion . . . . .	46
6.4	Future Work . . . . .	47
<b>Bibliography</b>		<b>49</b>

<b>A User Study 1 Tasks and Questions</b>	<b>52</b>
<b>B User Study 1 Results</b>	<b>54</b>
<b>C User Study 2 Tasks and Questions</b>	<b>57</b>
<b>D User Study 2 Results</b>	<b>59</b>
<b>E Final Ontology structure</b>	<b>64</b>

**Word Count: 11034**

# List of Tables

4.1	Agile approach taken to implement solutions after Study 1 . . . . .	34
A.1	Study 1 - Tasks for restaurant owners . . . . .	52
A.2	Study 1 - Tasks for customers . . . . .	53
A.3	Study 1 - Questions for all participants . . . . .	53
B.1	Study 1 - Task results for restaurant owner . . . . .	54
B.2	Study 1 - Task results for customer . . . . .	55
B.3	Study 1 - Question responses from restaurant owners . . . . .	55
B.4	Study 1 - Question responses from customers . . . . .	56
C.1	Study 2 - Tasks for restaurant owners . . . . .	57
C.2	Study 2 - Tasks for customers . . . . .	58
C.3	Study 2 - Questions for all participants . . . . .	58
D.1	Study 2 - Task results for restaurant owners . . . . .	60
D.2	Study 2 - Task results for customers . . . . .	61
D.3	Study 2 - Question responses from restaurant owners . . . . .	62
D.4	Study 2 - Question responses from customers . . . . .	63

# List of Figures

2.1	Example ontology structure . . . . .	16
3.1	Overall system design . . . . .	23
3.2	Initial ontology design . . . . .	24
3.3	Example ontology design for recipes[11] . . . . .	24
3.4	Final ontology design . . . . .	25
3.5	Overall Restaurant UI architecture . . . . .	26
3.6	Restaurant UI - Add Ingredient Page Design . . . . .	27
3.7	Restaurant UI - Search Dish Design . . . . .	28
3.8	Overall Customer UI architecture . . . . .	28
3.9	Customer UI - Extras Page Design . . . . .	29
4.1	Restaurant UI - Final Architecture . . . . .	35
4.2	Restaurant UI - Add Ingredient Page . . . . .	36
4.3	Restaurant UI - Error Message . . . . .	36
4.4	Restaurant UI - Edit Dish Page . . . . .	37
4.5	Customer UI - Search Page . . . . .	38
4.6	Customer UI - Extras Page . . . . .	39
4.7	Comparison of two list selection models . . . . .	41
E.1	Final ontology structure . . . . .	64

# **Abstract**

AN ONTOLOGY-BASED RESTAURANT INFORMATION SYSTEM

Thomas Hayter

A report submitted to The University of Manchester  
for the degree of Bachelor of Science, 2023

The aim of this report is to explore ontologies and their ability to store information. This report explores ontologies in the context of the menu of a restaurant, and how it is possible to store dietary information, along with a list of the ingredients in each dish. This also explores whether ontologies are viable options for storing knowledge in real world systems. It details the development, testing and evaluation of an ontology-based system for a restaurant that has two user interfaces. One interface is for a customer to query the menu, and the other is for the owner of a restaurant to manage the menu within the system. The system is evaluated with a user study after development is complete. The findings of this report find that ontologies are perfectly capable of storing information in a real world application, and have the capability to store extensive information about the menu for a restaurant.

# **Declaration**

No portion of the work referred to in this report has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

# Copyright

- i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.
- ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made **only** in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.
- iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.
- iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see <http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=24420>), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see <http://www.library.manchester.ac.uk/about/regulations/>) and in The University’s policy on presentation of Theses

# Acknowledgements

First of all, I would like to thank my supervisor, Professor Ulrike Sattler, for her guidance, reassurance, and advice throughout this project. Her support is greatly appreciated. I would also like to thank all of the participants of my user studies, who all provided excellent, insightful and extensive feedback. Your observations have transformed this project.

# Chapter 1

## Introduction

### 1.1 Overview

My project, “*An ontology-based restaurant information system*”, explores how ontologies can be used to store information about a restaurant’s menu. In a world where over half of the UK population are eating out at restaurants and fast food chains multiple times a month [30], and a growing emphasis is being placed on dietary requirements, it is vital that food vendors are able to provide such information to customers. While some customers have specific diets such as vegetarianism and veganism, others have specific allergies or are simply looking to reduce their calorific intake. Religious diets also play a huge part in what we can and cannot eat. Therefore, the restaurant staff want to be able to provide sufficient information to customers about what dishes are available to them based on their requirements. There is also the aspect of modifying dishes. For example, “*Is a plant burger with extra bacon still suitable for vegetarians?*” and “*How many calories are in a cheese burger without the cheese?*”.

In this project I explore to what extent it is possible to store the necessary information from a menu in an ontology. For example, whether it is possible to store calorific information about ingredients, calculate dishes suitable for certain diets and calculate new information about a dish with certain ingredients removed or extra ingredients added. I also evaluate whether ontologies can be used in a real world application such as this. To do this, I make sure that the approach that I take to store the information is simple, effective and fast to reason. Then, I evaluate whether an ontology-based system can be used by those who are not familiar with ontologies and how they work, and I conclude whether it would be suitable to implement in restaurants.

The final product consists of two user interfaces, the ontology itself, and the code

to link the two together. The first user interface (UI) is targeted at the customer of the restaurant. They need to view and filter the menu based on their dietary requirements. Furthermore, they could also modify dishes by adding or removing ingredients. Then, they should be able to see the updated information for the modified dish. The second user interface is aimed at the owner of the restaurant. They need to be able to create new dishes and edit the menu for the customers.

The project is evaluated by a user study, where I ask participants to take on the role of either a customer or a restaurant owner. They carry out tasks based on their role, and then I ask them questions afterwards. Their success at completing the tasks, combined with the answers to the questions will form the evaluation of the system.

## 1.2 This Report

This report is broken down into six sections. In the remaining part of the introduction chapter I will describe the planning of the project. The second chapter focuses on the background to the project, including information about diets, technologies that I have used and how ontologies work.

The third chapter discusses all of my design choices for the project and the design of the evaluation. It details the methodology used to create the final system, and also lists my requirements and success criteria for the project. The fourth chapter is where I describe the final product and how I developed it. The fifth chapter lists the results of the user studies that I conducted. The final chapter evaluates the project as a whole, discusses whether I met my objectives, what I could have done better and explores where this project could be developed further.

## 1.3 Project Plan

The project ran from September 2022 until April 2023. I broke the project down into multiple stages: research, development and testing. I spent the first few months researching how ontologies work and following tutorials to understand how different technologies works. This included getting some hands on experience by using the technologies myself to make sure I truly understood how they worked. Then I designed the system that I wanted to create, and developed a minimal viable product that fits this. After that, I padded the system out with all the required features. The next step was to run my first user study, to receive some early feedback about the system. In the final

weeks of development, I finished developing the system and made changes based on the feedback from the first user study. Once it was finished, I conducted a second user study to evaluate the success of the system.

# **Chapter 2**

## **Background**

### **2.1 Diets**

After suffering from the COVID-19 pandemic, restaurants are keen to get as much business as possible. To do this, they need to cater to all customers and all dietary requirements. To achieve that, they need to allow customers to easily deduce which dishes are suitable for them.

Almost 30% of adults in the UK have some sort of vegetarian-based diet [13], and the percentage is greater in the younger generations than the older ones. With an increasing concern about the environmental, ethical and health impacts of the food that we eat - people are looking at reducing and removing meat from their diets (Flexitarian or Vegetarian), or removing all animal products completely (Vegan).

On top of this, 8% of children and 6% of adults have a food-based allergy [24]. The most common food allergies are cow's milk, eggs, peanuts, soybeans, wheat, fish, shellfish and tree nuts [29]. Allergic reactions can vary from mild to severe, and therefore, it is important that a restaurant is able to offer allergen information about their dishes. Furthermore, while Coeliac disease is not an allergy but a autoimmune disease, those who suffer from the disease want to avoid foods that contain gluten, which is a similar problem to allergies.

With 63% of people in the UK [22] and 84% of the population worldwide [9] stating that they have some religion, it is vital to consider religious dietary requirements. I have used this guide [10] to understand what dietary requirements different religions have. At least 45% of the global population will have a dietary requirement purely based on their religious belief, which makes it essential that they are considered by

restaurants. While some religions simply follow or encourage vegetarianism or veganism (such as Buddhism), others are more complex. Hinduism and Sikhism will specifically not eat beef or beef products. Furthermore, Muslims, Jews, and Rastafarians will not eat pork or pork products, with the former two not eating shellfish either. Therefore, specific measures should be taken to ensure that these customers can determine which dishes are suitable for them.

In addition to specific ingredients, there are other religious beliefs concerning food. Muslims are only permitted to eat Halal (permissible) foods, or foods that are not Haram (forbidden). Halal meats must come from an animal that has been slaughtered in a specific way according to Islamic Law (Shari'ah) [4]. The animal must be alive and healthy at the time of slaughter and, after invoking the name of Allah on the animal, at least three of the Trachea, Oesophagus and both Jugular Veins must be severed. In Judaism, followers may only eat Kosher foods [15] ; foods that conform to *kashrut* (Jewish dietary law). This specifies that they must not eat dairy and meat products together, and the two should not be prepared together either to avoid cross-contamination. Meat should also be sourced from an animal that has been slaughtered in a specific way.

Both Halal and Kosher foods do not solely revolve around what ingredients are in a dish, but also how that dish has been prepared. It should also be noted that not all religions are strict. Some religious followers are stricter with diets than others, so there is not one single catch-all dietary requirement for a religion.

## 2.2 Ontologies

While researching ontologies, I constantly referred to the “*Pizza Tutorial*” [6]. It provides a comprehensive understanding of Web Ontology Language (OWL) ontologies and how they work, while also providing examples and a tutorial to follow along with. In this section, I will represent classes in bold such as **ClassA** and properties in italic such as *propertyA*. Class expressions will be represented in quotation marks and italics, such as “*ClassA propertyA ClassB*”

An ontology is a knowledge-based way of storing information. They are stored in a *.owl* file. It works in a object oriented kind-of way, where a thing is a class. Ontologies make use of the concept of inheritance, so classes can have subclasses where the subclass “*is a*” superclass. For example, the class **House** could be a subclass of **Building**, and could have **Mansion**, **Bungalow** and **Cottage** as its subclasses. This

works because a house “*is a*” building and a cottage “*is a*” house. An implementation of this example in Protégé can be seen in Figure 2.1. Everything that holds true for a class also holds true for all of that class’ subclasses. In an ontology, every class is a subclass of **owl:Thing**. It can be useful to name a subclass with a suffix to make it clear what it is a subclass of. In the previous example, all subclasses of **House** could have the suffix “*House*” (e.g. **CottageHouse** and **BungalowHouse**) to differentiate them from anything else that could have the same name. For example, having one class called **Cottage** could lead to ambiguity between the building and the type of cheese - if both happened to be stored in the same ontology.

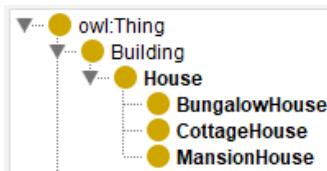


Figure 2.1: Example ontology structure

Ontologies contain facts, properties and axioms, and uses those axioms and properties to infer more facts. There is a tool called a reasoner which infers these facts. While many are available, I have used Hermit [16] as it is the most up to date.

Ontologies are not a common method of storing information. One of the largest ontologies, Yago [27], is predominantly harnessed for natural language tasks such as question answering, entity recognition and semantic textual analysis [20]. The common opinion on using ontologies in real world applications is that they become slow when more expressive reasoning is required [26, 28] due to the need for completeness. If the ontology is kept simple, and the reasoning is fast, I see no reason why OWL ontologies can not be used for real world applications.

### 2.2.1 Object and Data Properties

There are two kinds of properties: object properties and data properties. An object property relates two classes. The property can have a domain and a range class, which includes the class and all of its subclasses. Following on from the previous example, the object property *hasRoom* could have the domain **House** and the range **Room**. Specifically, you could represent a cottage having a kitchen with: “*CottageHouse has-Room KitchenRoom*”. A data property links a class to a literal value, such as an integer, boolean value or string. They can also have a domain class and a range data type (integer, string etc.). For example, the data property *hasNumberOfWindows* could be

used to link a **House** class to an integer value of how many windows the house has. You could represent a bungalow having 3 windows as: “*BungalowHouse hasNumberOfWindows 3*”.

Both these properties can have many characteristics. These properties are also stored in an object oriented fashion, so if the *hasKitchen* property was a subproperty of *hasRoom*, and it is stated that “*CottageHouse hasKitchen ModernKitchen*”, then the reasoner would be able to infer that “*CottageHouse hasRoom ModernKitchen*”. Properties can be transitive: if **X** is related to **Y** via *propertyA*, and **Y** is related to **Z** via *propertyA*, then **X** is related to **Z** via *propertyA*. An example of this would be the property *isAncestorOf*. They can also be functional. This means that there is at most one entity that is related to an entity via that property. Therefore, if two entities are related to an entity via a functional property, then it will be reasoned that they must be the same entity. An example of a functional property would be the data property *hasNumberOfWindows*, to ensure no house has more than one number of windows. Properties can also have inverses. If **X** is related to **Y** via *propertyA* and *propertyB* is the inverse of *propertyA*, the **Y** is related to **X** via *propertyB*. The domain of *propertyA* will be the range of *propertyB* and the range of *propertyA* will be the domain of *propertyB*. A property can be symmetric. If **X** is related to **Y** via *propertyA*, then **Y** is related to **X** via *propertyA*. An example of that would be *hasNeighbour*. Properties can also be asymmetric, which means they can never be symmetric. Additionally, properties can be reflexive, meaning they always relate an individual to itself. The mathematical operation = would be an example of this, as any number is always equal to itself. They can also be irreflexive, which means an individual can never be related to itself. An example could be *hasFather* - as you cannot be your own father.

## 2.2.2 Quantifiers

Two quantifier restrictions are used in this project, existential and universal quantifiers. Existential quantifiers are represented by the keyword “some”, and require at least one relation for a specific property. For example, dishes that “*hasIngredient some VegetableIngredient*” are equivalent to dishes that contain at least one vegetable. Conversely, universal quantifiers require all relations to be of a specific relation, and are represented by the keyword “only”. For example, dishes that “*hasIngredient only VegetableIngredient*” are equivalent to dishes that only contain vegetables.

### 2.2.3 Defined Classes

Defined classes are classes that have constraints defined by axioms. What differentiates a derived class from a normal (primitive) class is that the constraints for a primitive class are necessary, whereas the constraints for a defined class are necessary and sufficient. Necessary axioms can be thought of as “All members of this class must fulfil these conditions”. Necessary and sufficient axioms can be thought of as “Anything that meets these conditions is a member of this class”. An example of a derived class is a **VegetarianDish** could be defined as a dish that contains no **MeatIngredient**. In an ontology, you would say that a **VegetarianDish** is a subclass of **Dish** and a subclass of “*not(containsIngredient some MeatIngredient)*”.

### 2.2.4 Instances

Instances are just that, an instance of a class. You can have many instances of a class, each with different object properties and data properties. This way, you could create two instances of a bungalow, but with different numbers of bedrooms and different addresses. Instances are useful when you want to have many occurrences of a class that are slightly different.

### 2.2.5 Enumerations

Enumerations are also a concept from object oriented programming that is utilised in ontologies. When a property can only take one of a few possible values, it can be convenient to create an enumeration - which explicitly defines those few possible values. An example would be the seasons, which can only be winter, spring, summer or autumn.

There are other aspects of ontologies that are not used in this project, such as annotations and DL and SPARL querys.

### 2.2.6 Open World Assumption

Ontologies have an open world assumption, which means that we can not assume that something is false, purely because we do not know that it is true. We need to explicitly specify that it is false. Otherwise, it is assumed as knowledge that is yet to be added to the ontology. It is not sufficient to say that a dish contains some ingredients, you also

need to state that it contains only these ingredients. Take the example of a **MargheritaPizza**, which contains the ingredients **PizzaBaseIngredient**, **TomatoIngredient**, and **CheeseIngredient**. Simply stating that a **MargheritaPizza** “*containsIngredient some PizzaBaseIngredient*”, “*containsIngredient some TomatoIngredient*”, and “*containsIngredient some CheeseIngredient*” would mean that any pizza with these 3 ingredients is classed as a **MargheritaPizza**. Because of this, a **MargheritaPizza** instance could still contain a **MeatIngredient** as long as it had the three required ingredients, and therefore we cannot class it as a **VegetarianPizza**. It is important to specify that a **MargheritaPizza** “*containsIngredient only (PizzaBaseIngredient or TomatoIngredient or CheeseIngredient)*” so that we specify only those 3 ingredients can be in a **MargheritaPizza**.

## 2.3 Technologies

I initially used Protégé to familiarise myself with working with ontologies. The user interfaces and the code to connect them to the ontology are written in Java as it has the useful and comprehensive OWL Application Programming Interface (API) [5], which can be used to interact with OWL Ontologies. I used Java Swing to create the user interfaces themselves. The project is built and packaged using Maven [21] , which produces a runnable *.jar* file that should be placed in the same folder as the *.owl* file.

### 2.3.1 Protégé

Protégé [8] is a free, open-source software interface for interacting with ontologies. The workspace is customisable and makes it easy to navigate ontologies and the relationships within them. It also supports the querying of ontologies.

I used Protégé throughout my development to monitor my menu ontology. When my user interfaces made changes to the ontology, I used Protégé to ensure the expected changes had been made. I also employed Protégé to help me debug problems, or manually fix the ontology if the system was not interacting with the ontology as expected.

### 2.3.2 The OWL API

The OWL API is a Java API that allows you to create and manage OWL ontologies. It can be installed to a project using Maven. I followed the following tutorial [12] to begin to get to grips with the OWL API, along with the documentation that is available

here [18]. I chose to use this API over various Python APIs because I am familiar with Java and find Java much easier to use than Python when working with new, less documented APIs. On top of this, the API has been developed by The University of Manchester, so it would be easier to find someone to answer any questions I had while using it.

### **2.3.3 Java Swing**

Java Swing [19] is a Graphical User Interface (GUI) widget library for Java. It contains many lightweight components that can be utilised and customised easily. It provides a familiar and intuitive aesthetic to components that make users instinctively know what to do with them. The library is platform independent, which means GUIs developed using it are portable to many devices.

I chose Java Swing because I was familiar with the various components and believed they would provide an intuitive interface for customers. Because my project does not focus on the aesthetics of the system, I was not going to be changing how the components looked. Therefore I wanted a library of components that was already set up to be usable.

In Java Swing, a JFrame is the window of the application, and it contains JPanels (a view). JPanels have a layout, and you can add many types of components to a JPanel. These components are collectively referred to as JComponents. Examples of JComponents include JTextFields, which allow the user to enter text, JButtons which display a button to carry out an action, JLists to display arrays of data and JCheckboxes to display a tick-box. A JSlider allows a user to choose a particular value from a set range, and a JOptionPane displays a message in a pop-up window - which is useful for giving informative feedback. You can also add a JPanel to itself to create more complex designs and layouts.

# **Chapter 3**

## **Design & Methodology**

This chapter begins by outlining the requirements of the system that I built for this project, before exploring the original designs for the system. It concludes by looking at how I planned to evaluate the system.

### **3.1 Requirements**

To critique the system, I have come up with a list of functional and non-functional requirements that the system should comply to. These are broken down into those which are compulsory, and those which would be good to include if time permits.

#### **3.1.1 Functional Requirements**

Functional requirements for the system that are necessary to implement:

- The system needs to be able to store all dishes from the menu of a restaurant, along with the ingredients that make up that dish.
- The system should be able to produce a list of these dishes to the customer.
- The owner of a restaurant should be able to add to and manage the ingredients and dishes in the system.
- A customer needs to be able to query the system for dishes they desire.
- Therefore, the system should filter the dishes by various dietary requirements, such as diets, allergies and filter by calories.

- The system should have the option to hide calorie information to those who do not want to see it.

Possible additions to the system:

- The system could filter dishes by how a meal has been prepared and cooked.
- The system could calculate new information for a dish based on whether a customer would like to add or remove certain ingredients.
- The system could allow for a restaurant owner to register new allergens and diets to filter by.

### 3.1.2 Non-Functional Requirements

The non-functional requirements of the project that the system must comply by:

- The system must be intuitive, easy to use, and provide appropriate feedback when changes are made to the ontology.
- The interface must conform to *the eight golden rules of user interface* [23].
- A restaurant owner should require minimal training to use the system.
- A customer should need no training to use the system.
- The system should allow customers to search for dishes within an appropriate response time. Performance is important as the restaurant and the customer will want the customer to make a decision quickly on what dish they would like to order.

### 3.1.3 Dietary Requirements

These are the dietary requirements that I want a user to be able to filter by:

- Vegetarian
- Vegan
- Religious diets e.g. Halal / Kosher preparation, Hindu and Sikh.

The system should also be able to filter by the most common allergies [3]:

- Peanut
- Wheat
- Cow's milk
- Eggs
- Fish
- Shellfish
- Tree nuts
- Soybeans

## 3.2 System Design

I have used paper prototypes [25] for my designs because they are incredibly quick to produce and can be modified easily if additions need to be made. This also makes it quick to compare different designs side-by-side.

The design for the system was to have the ontology in one file, and the two user interfaces separately. These would be linked together by an “OntologyManager” class that handles all of the work with the OWL API. This design would remove any ontology functionality from the UI classes to keep them solely focused on dealing with user input and output. A rough design for this can be seen in Figure 3.1.

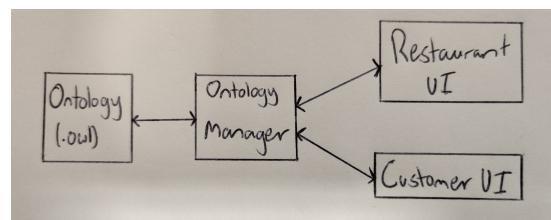


Figure 3.1: Overall system design

### 3.2.1 Ontology Design

My first ontology design was laid out as can be seen in Figure 3.2 and was based on the design in Figure 3.3. It consisted of the following classes: **Dish**, **Component**,

**Ingredient** and **Nutrient** with an object property *hasX* to link them in that order. A **Dish** contains **Components**, which contains **Ingredients** which contains **Nutrients**. Then there are **Customers**, who *canEat* a **Dish**. **Ingredient** has subclasses to categorise different food types: **CarbohydrateIngredient**, **DairyIngredient**, **FatIngredient**, **MeatIngredient**, **VegetableIngredient** and **OtherIngredient**. Calorie information is stored on an ingredient level, with a data property *hasCalories*. **VegetarianDish** will be a subclass of **Dish** that will also be a defined class. It will be defined by “*Dish and not(hasPart some MeatIngredient)*”. Similarly, **VeganDish** will be a defined class defined by “*Dish and not((hasPart some MeatIngredient) or (hasPart some DairyIngredient))*”.

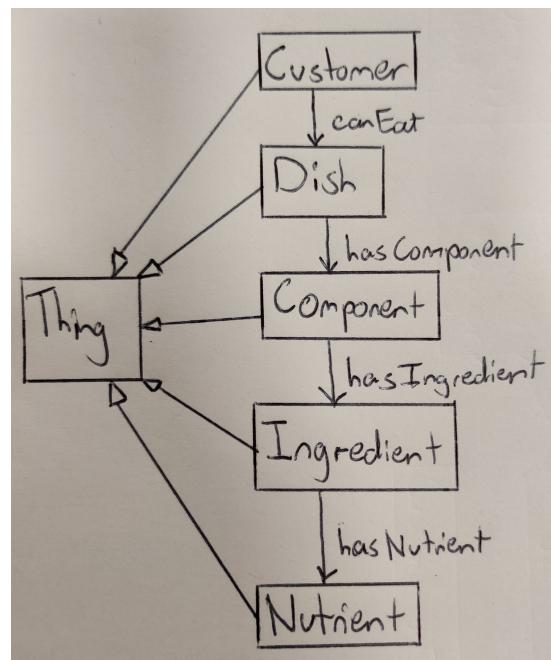


Figure 3.2: Initial ontology design

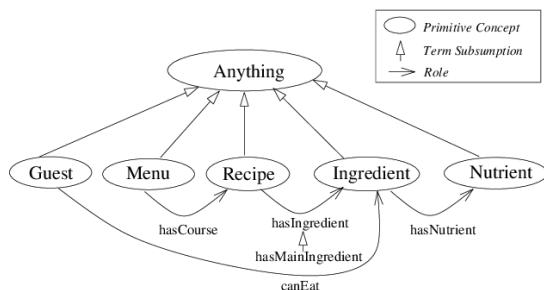


Figure 3.3: Example ontology design for recipes[11]

After reading the evaluation of the paper that inspired that diagram [11], I realised how easy it was to create redundant data. For example, the **Customer** class is not required at all - you only need to filter and search for dishes. No information about a specific customer needs to be stored. Considering I wanted to keep the ontology as streamlined as possible, I removed the **Customer** class. On top of this, I realised a **PreparationMethod** class would need to be created to store whether a dish was suitable for halal or kosher diets based on how it was prepared. Therefore, I refined the original design to the one in Figure 3.4.

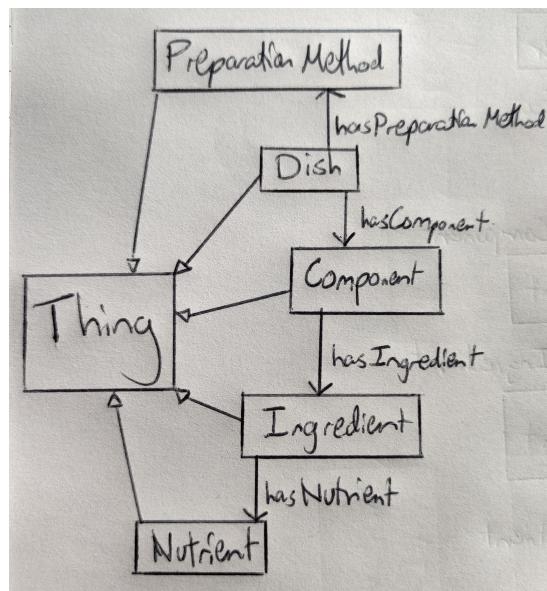


Figure 3.4: Final ontology design

I did not plan to use enumerations or instances in my ontology. I chose not to use instances because each ingredient would be the same regardless of which dish it was in. The beef patty in two different burgers would be exactly the same, as would the burger bun and the lettuce etc. Therefore, the information for each ingredient instance would be the same, so I could define all the information on a class level rather than an instance level. This is the same for components and dishes - both are always the same in every instance.

### 3.2.2 Restaurant UI Design

The user interfaces were designed to be simple but effective. They needed to contain all the functionality required of the system but no more. Where the concept of ontologies can be difficult to grasp, the user should not be put at any higher risk of being

confused. On top of this, I aimed to follow *The Eight Golden Rules of Interface Design* [23]. Specifically, I wanted to make sure that the UI was consistent, gave informative feedback, allowed the easy reversal of actions, reduced the short-term memory load and let the user feel in control of the system. These five points were more applicable to this system, so I chose to make them a part of the success criteria.

To conform to the *Golden Rules*, I set out to make sure to keep the appearance and terminology consistent throughout the system. Moreover, I designed back buttons on all pages to make navigation easy, and planned to display information boxes when changes are made to the ontology, such as when an entity is added or removed. Furthermore, I intended to make sure that the system behaves as expected and all actions and buttons are labelled correctly. Finally, I aimed to design the system in a way that meant that all necessary information was displayed on the screen, so that a user would not have to remember too much information.

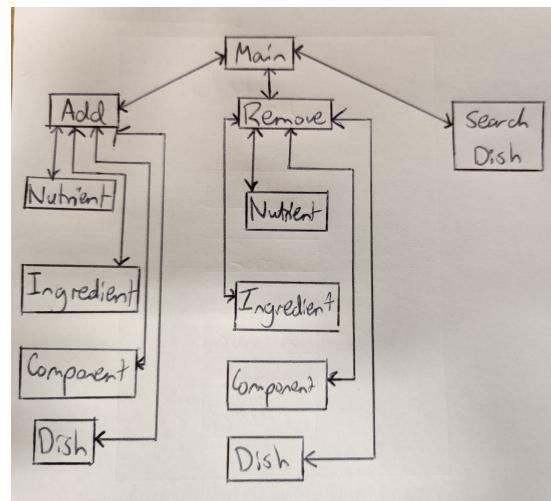


Figure 3.5: Overall Restaurant UI architecture

The first design of the architecture of the system can be seen in Figure 3.5. Each square represents a page in the system, and the arrows represent the ability to navigate from one page to another.

Upon entry to the system, the restaurant owner is shown the Main Menu Page. It contains buttons that navigate the user to the Add Menu Page, the Remove Menu Page and the Search Dish Page. The user can also close the system from here. The Add and Remove Menu Pages simply contain 5 buttons: Allergen, Ingredient, Component, Dish and Back. This allows the user to select what they would like to add or remove from the ontology.

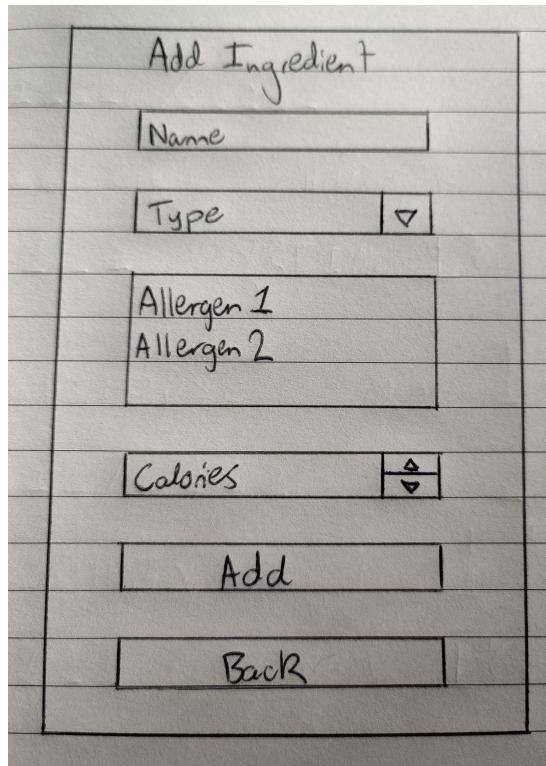


Figure 3.6: Restaurant UI - Add Ingredient Page Design

Each Add Page design contains a set of components to gather the necessary information about whatever is being added to the ontology. For example, the Add Ingredient Page design contains a text field - to enter the name of the ingredient, a drop-down box to select the type of ingredient that it is (Meat, Vegetable etc.) and a list to select all the allergies contained. There is also an input for the number of calories in the ingredient. The design for the Add Ingredient Page can be seen in Figure 3.6.

Conversely, the design for a Remove Page contains a drop down list of all the ingredients or components or dishes etc. for the user to pick one to delete.

The design of the Search Dish Page (see Figure 3.7) is to be split into two halves. The left hand side contains a list of results. The right hand side contains the filter menu, split into diets, calories and allergens. Once the user inputs their preferences, the “Search” button refreshes the list of results.

### 3.2.3 Customer UI Design

The Customer UI design needed to be simple; the customer needs to be able to make a decision on what they want to order quickly, and therefore this system needs to be

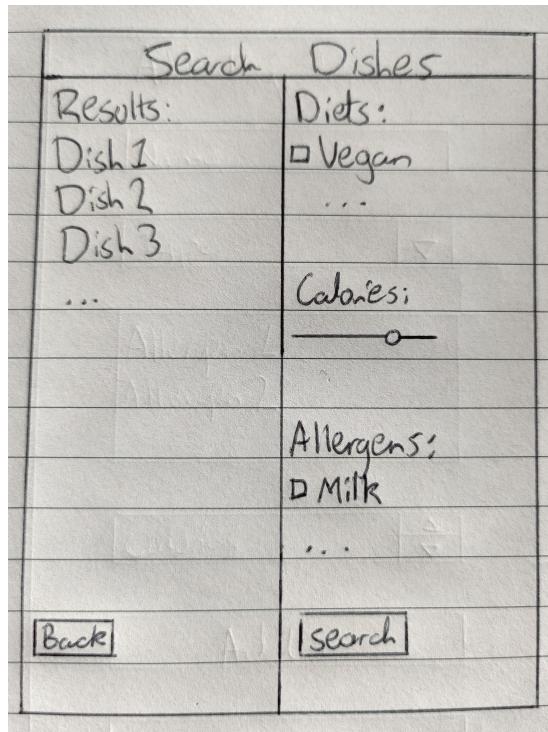


Figure 3.7: Restaurant UI - Search Dish Design

straight forward. The architecture of the Customer UI can be seen in Figure 3.8. It only consists of five pages, the Main Page, the Settings Page, the Search Page, the View Dish Page and the Extras Page.

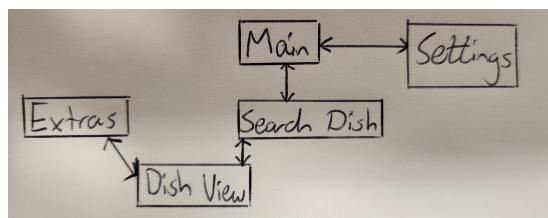


Figure 3.8: Overall Customer UI architecture

The design starts at the Main Page, where the user is directed upon opening the application. From here, they can navigate to the Settings Page, or the Search Page.

The Settings Page design consists of one checkbox option, whether the user wants to see calorie information or not.

The Search Page design (see Figure 3.7) will look the same as it does in the Restaurant UI, where the user can filter for dishes based on requirements. However, each result will also be linked to a Dish View Page for that dish.

The Dish View Page design contains all the information about a single dish. It

presents a list of ingredients in the dish, as well as the allergen, dietary and calorific data for that specific dish. It also contains a button to navigate to the Extras Page for that dish.

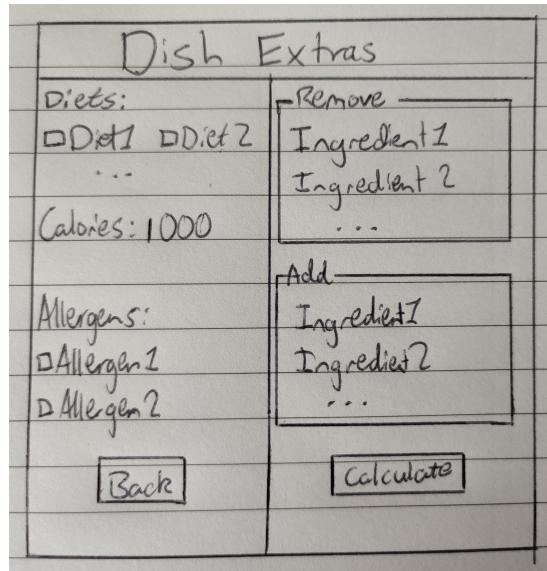


Figure 3.9: Customer UI - Extras Page Design

The Extras Page for a dish is where a customer can modify the dish, by adding or removing ingredients, and calculate new dietary information. The design for this page (see Figure 3.9) is split into two sides. The left hand side contains the dietary data. The right hand side contains two lists, the top one for removing ingredients that are currently in the dish. The bottom list is for selecting which ingredients should be added to the dish. The calculate button updates the data on the left hand side.

### 3.2.4 Menu

I chose to use the Honest Burgers [2] menu to fill my ontology with information for multiple reasons. First, the online menu contains a full list of ingredients for each dish. Second, it also provides lots of information about what allergens are contained in each dish, and whether they are suitable for vegans, vegetarians and whether the dish is Halal. Finally, they have also published the calorie information for their dishes [1], which would help me calculate calories for their ingredients.

### 3.3 Evaluation Plan

To test and evaluate the ontology system, I planned to conduct two user studies, where I would ask participants to carry out certain tasks using the system, such as adding or removing a dish from the ontology, and searching for dishes with certain parameters. These tasks cover each aspect of the functional requirements of the system to completely evaluate how successfully the system meets the criteria. I would then ask questions to assess how successful the system was at meeting all requirements. An example question is asking the user how fast they found the system to respond to their actions. Responses for these kind of questions were recorded using a Likert scale.

I planned the evaluation to be carried out in two studies. The first study (Study 1) would be carried out during development, so I could gauge what improvements still needed to be made to the system. This was important as I would be able to identify whether I was on the right track with the project, and gain feedback on what does and does not work. Then, I would carry out a second user study (Study 2) after development on the system has finished to see how well the system fits the requirements set out. Ideally, I would carry out a series of continuous feedback sessions - where users can repeatedly give feedback on the system for me to improve. However, given the time constraints of the project, I did not expect this to be possible.

Each user study would be split into two smaller studies, one of the user interface for the customer, and one of the user interface for the restaurant owner. The participants would be split into two groups and requested to act the role of either a customer or restaurant owner. Each study would be tailored towards what those two roles should be able to achieve from their separate interfaces. For example, the participants who play the role of a customer would be asked to conduct searches for many different dietary requirements. On the other hand, the participants playing the restaurant owner would be asked to add new dishes to the ontology and remove others. I particularly wanted to test whether, given a dish and a list of ingredients it contains, a participant was able to create the dish in the ontology by making their own components with no guidance. See Appendix A for the full list of tasks and questions in Study 1, and Appendix C for Study 2.

The participants taking the customer study would not be given any instruction on how to use the system, as real customers would not and should not require training to use the system. However, the owner of the restaurant would be told how the system works before the Study 1, but not before the Study 2. This is because in the real world, they would only receive training when initially learning the system, and should not

need to relearn every time they want to use the system. Since a restaurant does not usually change its menu that often, they would not be using the system frequently.

It is important that the participants of the study are comfortable and give an honest review of the system. To do this, I aim to avoid putting them under any pressures. I would not place them under any time constraints to complete the tasks, and make sure that when I propose the study to them that the time estimation is accurate. It is also important that the participant does not feel any pressure to falsely speak well of the system when it is in fact failing. To do this, before the study I would make it clear to the participants that they are not being judged based on whether they were able to complete the tasks, and that the results are solely a reflection on how well the system is built and not their abilities. I would also make sure they knew that I wanted to hear about all problems as I want to know how to improve the system.

I chose to have 3 participants per role for my study, as I found in Study 1 (see Section 4.1) that this number was the most appropriate. The big issues that I expected were brought up by all participants, and everything else was small or personal preferences that were low priority issues to fix.

### 3.3.1 Ethics

I aimed to carry out the studies out in an ethical way by following the university ethical guidance [17]. All participants are over 18 and able to fully consent to taking part. No participant is coerced into completing the study, and they are free to withdraw at any point. No personal information is stored about them, or any information that can be used to identify them. The only information that is collected is whether they were able to complete all the tasks, and the feedback they give, which is both qualitative and quantitative.

On top of this, no harm is inflicted on the participants, and I make it clear how I am use the information I collect.

# **Chapter 4**

## **Implementation**

In this chapter I will begin by discussing the results of Study 1, which occurred during development. I will then explain each aspect of the final system that I developed, and how they each work.

### **4.1 Study 1**

I carried out Study 1 with 6 participants, 3 who acted as a customer and 3 who acted as a restaurant owner. The aim of this study was to find out whether the user interface I was developing was on track to meet the requirements I set out in Section 3.1 and find out where it needed to be improved. I asked them to carry out tasks specific to those roles. Then I asked them questions based on their experience. The full list of tasks and questions can be found in Appendix A, while the results can be found in Appendix B. This study was carried out during development of the system, so I did not expect the results to be perfect as there were still some bugs in the system, and not all of the features had been implemented.

#### **4.1.1 Customer UI Study**

The results of the customer study showed that everybody was able to complete each task except two: the task that required the participant to add and remove ingredients to and from a dish, and the task that required them to find the allergens in a modified dish. The former happened because the participants struggled to select multiple items in the current selection model. The latter happened because the functionality had not been developed into the system yet. One participant stated that needing to double click

on a dish in the results list was not an intuitive way to open the dish page, and another that the system should include information concerning gluten free alternatives.

The Customer UI was rated, on average, 4.6/5 for speed, 4/5 for intuitiveness and 5/5 for ease of use.

### 4.1.2 Restaurant UI Study

The study on the restaurant UI revealed that participants particularly struggled with tasks that revolved around them selecting multiple items in a list. Only 1 participant realised that the Control key had to be held while selecting multiple items, but only after failing to do so previously. The study also unveiled a bug concerning updating the pages when the ontology was updated. They each said they initially struggled with the relationship between dishes, components, and ingredients - but got used to it eventually. One participant mentioned that it would be useful to be able to edit Ingredients, Components and Dishes, without needing to delete everything and recreate it. They also mentioned that it would be useful to have all the existing ingredients and components listed somewhere in the application. However, the participants liked the clear layout of the system.

The Restaurant UI was rated, on average, 4.6/5 for speed, 4/5 for intuitiveness and 4/5 for ease of use.

### 4.1.3 Interpreting the Results

With the results of the study, I adopted a mildly agile workflow to move forward. I listed all of the problems that were identified during the study, and then a list of solutions that would solve them. I scored each solution from 1-5 with an urgency score - based on how important it was that this change was made. I also gave each solution a time score - how long I estimated it would take to implement the solution. I then made sure to tackle the high urgency issues first, especially those with a small time estimate. This way, I could make sure to get as many of the more impactful problems solved within the time I had left. An extract of this method can be seen in Table 4.1.

Solution	Urgency	Time
Remove need to hold Ctrl to select multiple items	5	2
Make edit pages	4	5
Underline focused links in list of dishes	3	2
...	...	...

Table 4.1: Agile approach taken to implement solutions after Study 1

## 4.2 Deliverables

### 4.2.1 Ontology

The final ontology (stored in ‘*Menu.owl*’) contains the Honest Burgers [2] menu. The overall structure of the final ontology without any menu data can be seen in Appendix E. Each dish on the menu is a subclass of **NamedDish** in the ontology.

All the ingredients in a dish are also in the ontology. Each ingredient stores the number of calories that ingredient contains (on a class level), and the allergens it contains. There were times when I had to decide what level an ingredient was, for example the **PestoIngredient** which contains multiple ingredients in itself, but is listed as an ingredient in the dish. I thought it was not useful to a customer to list every single thing that makes up pesto, but just to make sure the allergens are stored. Otherwise the list of ingredients would be much larger.

What ingredients to put in a component is also a subjective task, and for this menu there is no intuitive set of components to make. I decided in this case to store the patty, burger bun, and fillings for each dish as the components.

The object properties in the ontology are *hasPart*, which has the sub-properties *hasNutrient*, *hasIngredient*, and *hasComponent*. These are used to link a **Nutrient** to an **Ingredient**, an **Ingredient** to a **Component** and a **Component** to a **Dish** respectively. These components are transitive, because if a **Ingredient** is part of a **Component**, then it is also a part of any **Dish** that **Component** is a part of. The object property *preparedUsingMethod* links a **PreparationMethod** such as Halal or Kosher to a **Dish**, to store which dishes have been prepared in that way.

The data properties in the ontology are *hasCalories*, which links an **Ingredient** to an integer value of how many calories it contains. The data property *hasGluten-FreeOption* links a **Dish** to a boolean value of whether there is a gluten free alternative of the dish available. This was an addition made after Study 1 (see Section 4.1).

Looking at defined classes, **VegetarianComponent** is defined by the class expression “*Component and not(hasIngredient some MeatIngredient)*”, while **VeganComponent** is defined by the class expression “*Component and not((hasIngredient some MeatIngredient) or (hasIngredient some DairyIngredient))*”. **VegetarianDish** is then defined as “*Dish and (hasComponent only VegetarianComponent)*”, and **VeganDish** is defined as “*Dish and (hasComponent only VeganComponent)*”. **HalalDish** is defined by the class expression “*Dish and (preparedUsingMethod some HalalPreparationMethod)*”, and similarly **KosherDish** is defined by “*Dish and (preparedUsingMethod some KosherPreparationMethod)*”.

#### 4.2.2 Restaurant User Interface

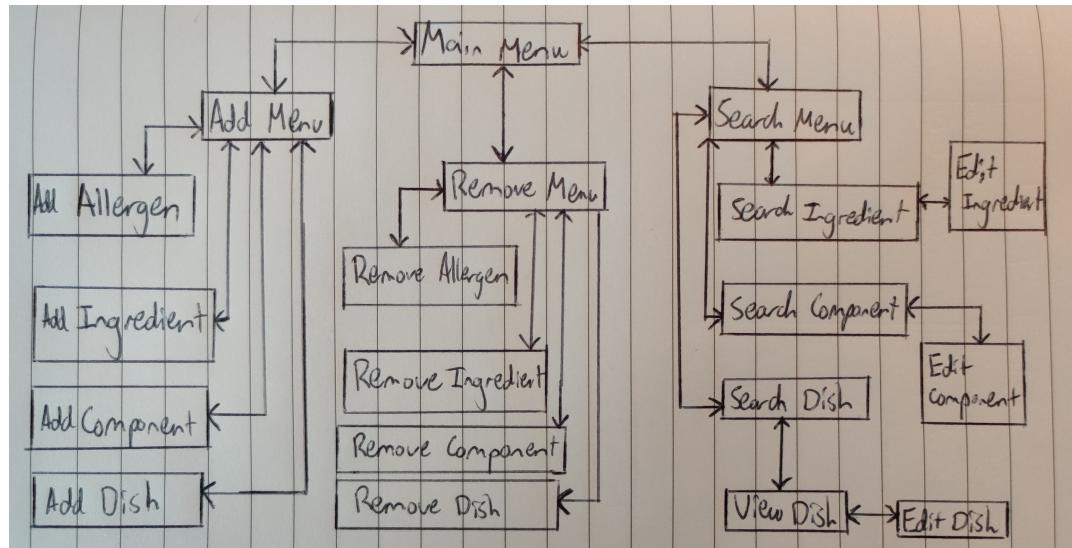


Figure 4.1: Restaurant UI - Final Architecture

The final architecture for the Restaurant UI can be seen in Figure 4.1. The user interface for the restaurant owner opens onto the Main Menu Page. The user is presented with 4 buttons to navigate the system: to add to the ontology, remove from the ontology, search/edit the ontology and quit the application. The quit button will save the ontology and close the application.

The Add section contains options to add either an allergen, ingredient, component or dish to the ontology. The page for adding an ingredient can be seen in Figure 4.2. Each of the add pages contains the necessary JComponents that allow the user to input the information required to create that entity. After attempting to add to the ontology, a message box will be shown in response, depending on whether the attempted addition

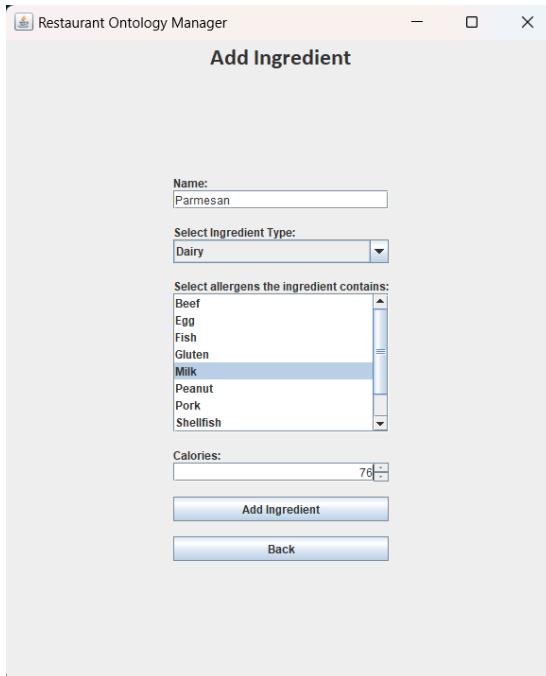


Figure 4.2: Restaurant UI - Add Ingredient Page

to the ontology is valid or not. If the user tries to add an entity without a name, a component that contains no ingredients, or a dish that contains no components, they will be shown an error message (see Figure 4.3), and the entity will not be added to the ontology. If all is valid, a message to inform the user that the entity has been added to the ontology will be displayed on the screen.

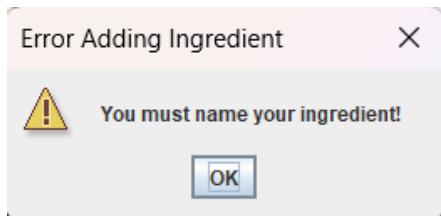


Figure 4.3: Restaurant UI - Error Message

The remove section is very similar to the add section, except they show lists for a user to select an element to remove. Similarly, a message feedback box is shown to inform the user of the changes to the ontology.

There is also a section for searching and editing ingredients, components and dishes. This section was also inspired by the feedback from Study 1 (see Section 4.1). In the search page for an ingredient, component or dish, a user can scroll through a list of all existing entities and select one to edit.

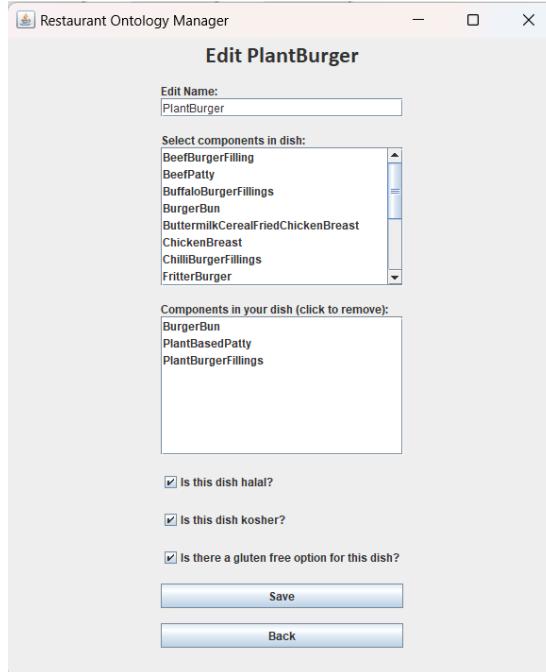


Figure 4.4: Restaurant UI - Edit Dish Page

Upon selection, the user is presented with a page that allows them to edit their selection. The edit pages consist of a similar form of JComponents to the add pages. However, they are loaded containing all of the information about that entity. Figure 4.4 shows the Edit Dish Page, which allows them to select new components for their dish, remove existing ones and change some properties about the dish.

### 4.2.3 Customer User Interface

The architecture of the Customer UI turned out as originally planned (see Figure 3.8). The user interface for the customer also opens onto a main menu page where the user has 3 buttons to navigate through the application. The user can search the menu, open the settings page or quit the application.

The settings page contains one option, the option to hide calorific information. This option will disable the ability to filter by calories, and also hide how many calories are in a dish and a dish with the ingredients modified.

The search page (see Figure 4.5) contains the results of a search on the left hand side. Clicking on a dish will load the information for that dish. On the right hand side of the page is the list of filters. The first filter returns dishes suitable for various diets (Vegetarian, Halal). The second returns dishes with a gluten free alternative available.

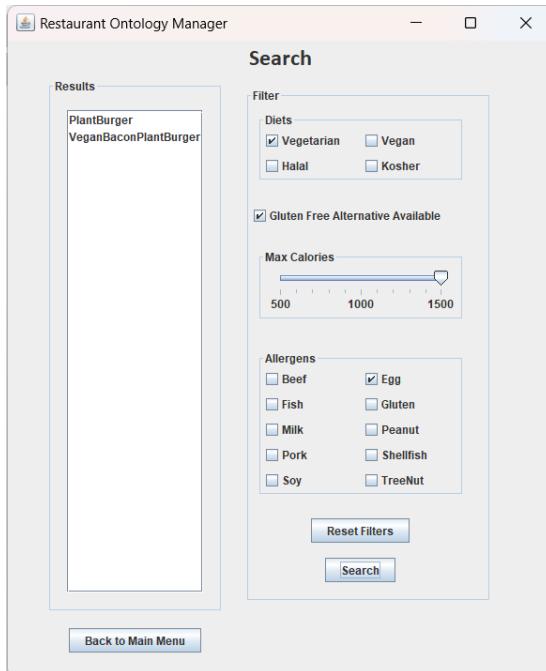


Figure 4.5: Customer UI - Search Page

The third filter returns dishes with less than the maximum amount of calories specified. The fourth filter removes dishes that contain specified allergens.

Clicking on a dish in the search page will take you to page specific to that dish. This page displays the full list of ingredients in the dish on the left, along with the dietary, calorific and allergen information for that dish on the right. It also contains a link to add or remove ingredients from the dish and see updated information.

The extras page (see Figure 4.6) also contains the list of ingredients in the top left. However, this time clicking on one of the ingredients will remove it from the dish. There is a list of all ingredients in the bottom left of the page, and clicking on one of them will add it to the dish. On the right side of the page is a “Reset Dish” and “Calculate” button. Clicking “Calculate” will update the allergen and calorie information, also on the right hand side of the page, to reflect the modifications the user has made to the dish. Clicking the “Reset Dish” button will reset the list in the top left to the original list of ingredients in that dish, and reset the dietary information on the right. Because preparation is done on a dish level and not on an ingredient level, it is impossible to know if a dish will be Halal or Kosher purely based on the ingredients - so this information could not be included on this page.

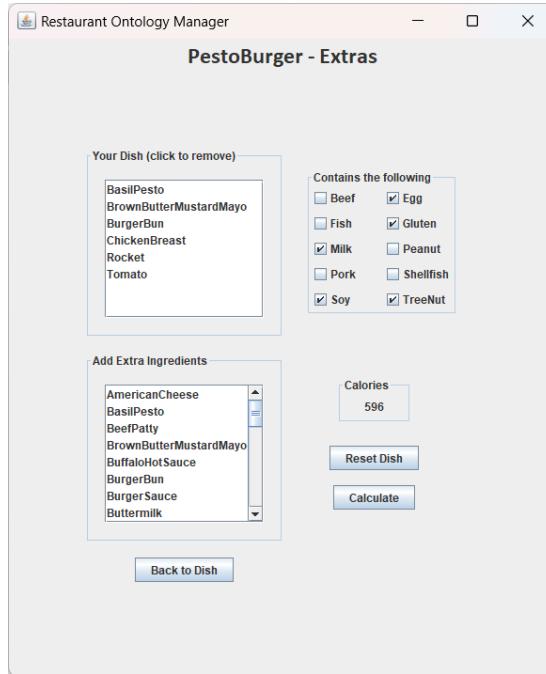


Figure 4.6: Customer UI - Extras Page

#### 4.2.4 OntologyManager

The Ontology Manager class contains many methods for interacting with the ontology via the OWL API. There are methods for adding allergens, ingredients, components, and dishes to the ontology by creating the various necessary subclass axioms, and removing them by removing each axiom that contains that class. There are also methods for updating ingredients, components, and dishes by replacing the subclass axioms with the new updated axioms. They also use the OWLEntityRenamer [18] to map the old class name to the new one.

The most challenging methods to write were the methods that searched or queried the ontology. For example, to retrieve the calories in an ingredient, the method gets a list of all the subclass axioms for that ingredient, checks for the one that contains the *hasCalories* data property, and then use a regular expression (regex) approach to extract the number of calories from the string version of that axiom. While it is not the cleanest solution, and not one I would recommend, it fulfills the purpose that I need it to since I could not find an existing method in the OWL API that would extract literal values from an OWL class expression. A similar method is applied to other similar methods, such as whether a dish has a gluten free option.

## 4.3 Design Choices

There were many things that did not go to plan, or that had to be changed during development. Study 1 (see Section 4.1) brought up problems I had not considered.

Initially, I developed each user interface entirely in one Java class, with methods that would create and return a JFrame for each of the different pages. I would then change between JFrames by closing the old one and opening the new one. After making a few pages I soon realised this was a bad idea. I ended up making 19 different pages, and this would create a very convoluted class. After doing some research [7], I found that a better way to approach this was to make each page its own class, extending the JPanel class. Then, create a JPanel with the CardLayout layout that can switch between each page. CardLayout makes it so that each JComponent is added to a JPanel with a String that can be used to identify it, and then you can switch to the JPanel you desire by using its identifier. This improvement was worth the time it took to learn about, because it made my code much easier to navigate and therefore saved me time in the long run.

Another design choice that was made after my pilot study was to change the way I showed how things in a list were selected. Previously, users had a list of everything provided, and had to click on an element to select it. If a user wanted to select multiple elements in the list, they would have to hold the Control (Ctrl) key on their keyboard to do so. If Ctrl was not held, any previously selected elements would be deselected. This is similar to how file selection works in File Explorer, and it is the default selection model for a JList. However, as many of lists were long, it was not obvious that the previously selected elements were now deselected, as the user would have to scroll to check. My first solution was to change the selection model so that you only needed to click on each element to select it (no Ctrl required), and the only way to deselect an element was to click on it again. However, with such long lists of ingredients and components it would still not be clear what had been selected, and you would have to scroll through the entire list to check. This solution would therefore not comply with the reduction in short-term memory load principle that I set out to follow. Therefore, I ultimately decided on a different method where there is one list which contains all possible selections (List 1) and another list which contains all the selected elements (List 2). To select an element, the user would click on it in List 1. This will add it to List 2. To deselect an element, the user would click on it in List 2, and it will be removed from List 2. This makes it much more clear what has been selected as you can see all the elements in List 2 without seeing everything that has not been selected. It

also supports the ability to select an element multiple times, which could be desirable in the case where a customer wants to add an ingredient to a dish twice - such as adding double cheese to a burger.

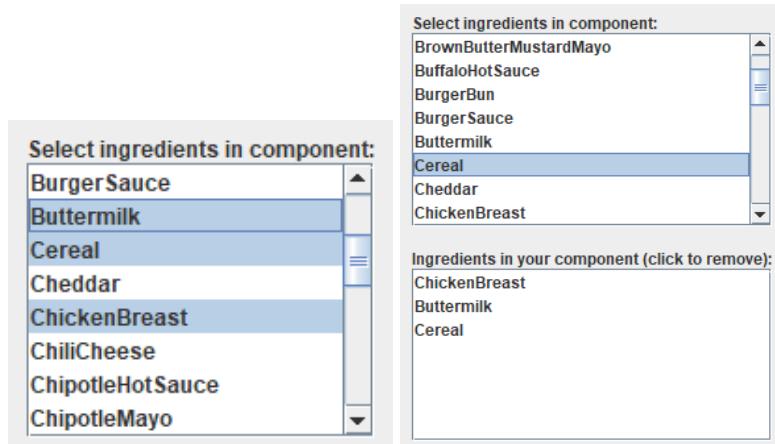


Figure 4.7: Comparison of two list selection models, pre-Study 1 (left) and final (right)

A comparison of these two methods can be seen in Figure 4.7, where (in the model on the right) the list at the top is List 1 and the list at the bottom is List 2. A user can constantly see what is selected, which makes this task much easier and gives the user reassurance they are in control of the application.

One final point that was raised in the studies was that the system should include information regarding whether dishes have a gluten free alternative. This was a simple addition to make, I added a boolean data property for a dish called *hasGlutenFreeOption*. Then a tick-box can be used to enter the information while creating a dish.

# **Chapter 5**

## **Results**

In this chapter I will discuss the results of the second user study that I conducted during my project.

### **5.1 Study 2**

Study 2 was carried out after I finished development on the system. The aim of this study was to evaluate how well the system met the requirements I set out in Section 3.1, and evaluate whether it would be able to be used in the real world. I also carried out Study 2 with 6 participants. The 3 participants who acted as a restaurant owner in Study 1 played the same role in this study. I asked them to carry out the same tasks, along with a couple of new tasks concerning new functionality that had been introduced since Study 1. All the same questions from Study 1 were asked, along with one new one for the restaurant owners to evaluate how well they could use the system without any training one month later. On top of this, I also asked some questions to evaluate how well the system fit *The eight golden rules of interface design* [23]. The full list of tasks and questions can be seen in Appendix C, while the results can be found in Appendix D.

#### **5.1.1 Customer UI Study**

The results of the customer study demonstrated that every participant was able to complete every task without any training or guidance. They reported that the system was very quick and easy to use. The calorie slider received praise because it removes the disappointment of finding a dish you like only to see it has more calories than you

would like. The importance of being able to turn off calories was also reiterated by the participants. They stated that all the information was clearly displayed and easy to find. Two of the three participants said that because they spent so much time in the search page, they almost forgot about the main menu and settings page.

This time, the Customer UI was rated, on average, 5/5 for speed, 4.6/5 for intuitiveness and 5/5 for ease of use. It also complied to all four of the applicable *golden rules of user interface* according to each participant.

### 5.1.2 Restaurant UI Study

The restaurant owner study also showed that every participant was able to complete every task set for them. They noted that the ability to search for ingredients, as well as edit them directly made them feel much more in control of the system, and made it easier to fix mistakes. They also noted that every part of the system was laid out in a clear and logical way, which made it obvious how to do what a restaurant owner would want to do. Conversely, they also noted that it is not clear what a Component should be in the context of this menu. One participant also noted that it would be useful to be able to delete a component from the edit page. They also appreciated the new method for selecting multiple items, which was much more intuitive. Overall, they all thought it had improved since the system in Study 1.

The Restaurant UI was rated, on average, 4.6/5 for speed, 5/5 for intuitiveness and 5/5 for ease of use. 5/5 was also the average score for how easy it was to pick up after not using the system for a month. On top of this, each participant stated that it complied to all five of the applicable *golden rules of user interface*.

### 5.1.3 Interpreting the Results

As this study was conducted after development had finished on my system, the results will form part of the evaluation of the system, which can be seen in Section 6.

# Chapter 6

## Summary & Outlook

### 6.1 Did I Meet My Objectives?

Overall, I believe I met all the necessary objectives for this project. Looking back to the System Requirements (see Section 3.1), I created a system that was able to store all the dishes on a restaurant's menu, along with the ingredients in the dish. The system could list the dishes to a customer, and they could filter the dishes by various dietary requirements. The system also had the option to hide calorie information. The owner of a restaurant could manage the menu by adding, removing and editing ingredients and dishes. I also managed to include the ability to generate new information after a customer modified the ingredients in the dish. I also managed to include filters based on how a dish had been prepared. I gave the restaurant owner the ability to add new allergens to the system, however one place that the system falls short is that they can't add new diets easily.

In terms of the non-functional requirements, participants believed that the system was intuitive, easy to use, and that it provided appropriate feedback. They also stated that they felt in control of the system at all times, and could easily reverse any actions they made. The system was reported as being consistent, and did not require the user to remember lots of information. Therefore, I believe it adheres to *The eight golden rules of interface design*. On top of this, participants were able to use the system with either no training if they acted as a customer, or minimal training if they acted as a restaurant owner. Finally, all participants found that the user interfaces ran fast.

In terms of dietary requirements, the system caters for all required allergens, the vegetarian and vegan diets, and the halal and kosher diets. It also provides information regarding whether gluten free alternatives are available for dishes. However, it could

be improved. Sikh's do not eat halal or kosher foods [10], as it is forbidden for them to eat meat from animals slaughtered according to other religious guidelines. Currently in the system, you cannot choose to remove Halal or Kosher options, only to remove the dishes that are not Halal or Kosher.

## 6.2 Improvements

There are still many improvements that can be made to the system. First of all, the appearance of the interfaces could definitely be improved. The aesthetics were not the focus of this project and I solely focused on the functionality, so there is great room for improvement. The visuals were mentioned during Study 2 by one user of the RestaurantUI, who mentioned that it was difficult to differentiate between all the different menu pages without reading the title as they are all very similar. Improving the visuals and making it easier to differentiate between the different pages would help this, as long as the overall appearance still remained consistent.

Another addition that could be made is the inclusion of a text search at the top of lists. The list of ingredients and components can grow very long, even with the ingredient type filter, and it can be easy to miss things - whereas a text search would make it easy to find specific items. This was also mentioned during Study 2.

On top of this, a lot of the text handling could be improved. At the minute, no class can have a space in the name of it as the system can't handle it. Spaces are stored differently in the ontology file and not handled when reading from it. The solution I would implement now is the following: When receiving an input of text from the user interface, remove any spaces and convert the text to pascal case before passing it to the OWL API. Then, whenever the name of an entity is received from the ontology to be displayed: enter a space before every capital letter except the first letter. This would work because the standard for storing things in an ontology is in pascal case, but the names of entities would be displayed with spaces in the user interfaces.

Another text handling issue is that you can't include the words "Dish", "Component", "Ingredient" etc. within the name of a class. This is because they are used as suffixes in the ontology, and all instances of the suffix are removed when displaying names from the ontology. This would be a simple fix, by making sure only the final occurrence of that text is removed from the name of a class, so that only the actual suffix is removed.

One improvement that was brought to light during the final user study that is specific to the Customer UI regards the settings page. Two of the three users stated that since you spend so much time on the search page, you forget about the main menu page. Therefore, it is easy to forget about the settings page, which can only be navigated to from the main menu page. They suggested being able to navigate to it from the search page instead, which would reduce the cognitive load of the user. This would also mean the Customer UI could be simplified further, and the home page could be removed entirely, making the UI even more lightweight and easier to navigate.

One issue I had throughout this project was what to call the collective group of specific foods that a customer could filter out. While a lot of these foods are allergens, I also included beef in this list to account for those who do not want to eat beef or beef products for reasons other than allergies. I have called them allergens in the user interface, but nutrients in the ontology. In my opinion, the term “Allergen” conveyed best the functionality that is inhibited by this list of foods in that a customer would want to view dishes without these foods. However, I used the term ‘Nutrient’ in the ontology because I believed that term encompasses the list of foods most accurately. It would improve the clarity of the system to have one term consistently throughout.

Another consideration I had at the end was the purpose of the Component. With the particular menu I have chosen, components are not particularly necessary. A burger is a whole meal and it does not consist of definitive components, just ingredients. This caused some confusion at what components to make during the studies when the user was given free reign to make a dish. Other menus lend themselves more to the idea of a Component. If your dish is a chicken and mushroom pie with mashed potato and roasted vegetables - you have three very clear components to the final dish made up of many ingredients themselves. The component architecture lends itself to larger dishes that contain multiple parts, rather than menus where you order lots of small bits. The reason I decided to keep components in is because it gives the system flexibility to be used in different kinds of restaurants. If components are not needed, you could still make the system work by making every dish a Component, and then the Dish in the system would only contain one Component.

## 6.3 Conclusion

Primarily, the results of Study 2 (see Section 5.1) reflect well upon the system. All participants were able to complete all the tasks they were set, which cover the functional

requirements of the system (see Section 3.1.1). They also reported that the system was easy to use, fast to respond and intuitive.

Therefore, I conclude that ontologies can be used to represent the necessary ingredient and dietary information stored on a menu. You can store information about many different types of dietary requirements, and even if they were not included in my final system - I discovered there is the ability to store them in an ontology. Importantly, it is possible to create a user interface that hides all the complex logic of an ontology. The user interfaces allow those who have no knowledge of ontologies to use a system that stores information in them without needing to know how they work. In this context, the only knowledge required is the hierarchical nature of ingredients, components and dishes.

Compared to a traditional database, I believe that an ontology makes it easier to represent real world objects and relationships, which makes it more applicable in a context like this. Ontologies also allow for information to be calculated, such as whether dishes are suitable for vegetarians and what allergens are present in a dish. In a traditional database system, this information would need to be input by a user and stored in a field that contained a boolean value. A database system would also require the user to enter a list of allergens for each dish, again increasing the workload for the owner of a restaurant. While it could take longer to set up, I believe the ontology-based system would save the owner of a restaurant time in the long run.

I believe this system, with some adjustments, could be used in a restaurant to manage and display dietary information. With some adjustments discussed in Section 6.4, I believe this system could be transformed into a fully functional electronic menu system. I also believe that ontologies can be used in real world applications where the size of the ontology is not going to be large. They have an intuitive nature about them and make it easy to represent real world objects.

## 6.4 Future Work

There are many possibilities when it comes to extending this project. The first is to add a transactional side to it, so that a restaurant owner would be able to add prices and the cost of adding extra ingredients. A customer would then be able to order dishes and pay for them, while the order would be sent through to the kitchen. This extension could see the project become a viable option for restaurants to use in practice.

Another extension of this project would be to make it more customisable from the

perspective of the restaurant owner in terms of diets. Diets are changing over time, and there are new dietary requirements that need to be catered for as time goes on. It would be useful if the owner of the restaurant could add more than just new allergens to the system. For example, Halal and Kosher are currently types of preparation method, but it would be useful if the owner could make more of these. One example would be deep fried foods, which are causing concern because of their high calorific content, high concentration of trans fats and the increase they cause in the risk of developing heart disease, diabetes and obesity [14]. Therefore the restaurant owner might think that it would be useful if customers were able to filter out dishes that have been deep fried at some point. Currently, they would have to contact a software developer who would have to hard code this change into the system. It would be useful if this was not the case.

# Bibliography

- [1] Honest Burgers. Honest burgers core menu nutritional info, Nov 2022. [https://www.honestburgers.co.uk/wp-content/uploads/2022/11/hb\\_nutritionals261022.pdf](https://www.honestburgers.co.uk/wp-content/uploads/2022/11/hb_nutritionals261022.pdf) (Last Accessed 08-03-2023).
- [2] Honest Burgers. Burger menus from honest burgers, Jan 2023. <https://www.honestburgers.co.uk/menus/> (Last Accessed 24-02-2023).
- [3] Wesley Burks, Ricki Helm, Steve Stanley, and Gary A Bannon. Food allergens. *Current opinion in allergy and clinical immunology*, 1(3):243–248, 2001.
- [4] Halal Monitoring Committee. What is halal? the meaning of halal explained in reference to food, Dec 2018. <https://halalhmc.org/resources/definition-of-halal/> (Last Accessed 20-04-2023).
- [5] OWL CS. Owl api, Feb 2023. <https://owlcs.github.io/owlapi/> (Last Accessed 20-04-2023).
- [6] Michael DeBellis. New protégé pizza tutorial, 2021. <https://www.michaeldebellis.com/post/new-protege-pizza-tutorial> (Last Accessed 20-04-2023).
- [7] Oracle Java Documentation. How to use cardlayout, Nov 2011. <https://docs.oracle.com/javase/tutorial/uiswing/layout/card.html> (Last Accessed 20-04-2023).
- [8] Stanford Center for Biomedical Informatics Research. A free, open-source ontology editor and framework for building intelligent systems, 2023. <https://protege.stanford.edu/> (Last Accessed 20-04-2023).
- [9] Conrad Hackett and Brian J Grim. *The Global Religious Landscape*. Pew Research Center, Dec 2012.

- [10] Public Health. Guidance on foods for religious faiths, 2009. <https://www.publichealth.hscni.net/sites/default/files/FaithsPosterA2.pdf> (Last Accessed 20-04-2023).
- [11] Pablo López-García, Eduardo Mena, and Jesús Bermúdez. Some common pitfalls in the design of ontology-driven information systems. *KEOD 2009 - 1st International Conference on Knowledge Engineering and Ontology Development, Proceedings*, pages 468–471, 01 2009.
- [12] Nicolas Matenzoglu and Ignazio Palmisano. An introduction to the owl api - university of manchester, Apr 2016. <http://syllabus.cs.manchester.ac.uk/pgt/2021/COMP62342/introduction-owl-api-msc.pdf> (Last Accessed 20-04-2023).
- [13] Edouard Mathieu and Hannah Ritchie. What share of people say they are vegetarian, vegan, or flexitarian?, May 2022. <https://ourworldindata.org/vegetarian-vegan> (Last Accessed 20-04-2023).
- [14] Kayla McDonnell. Why are fried foods bad for you?, Feb 2023. <https://www.healthline.com/nutrition/why-fried-foods-are-bad> (Last Accessed 20-04-2023).
- [15] Sade Meeks. What is kosher? definition, examples, diet, and more, Jan 2023. <https://www.medicalnewstoday.com/articles/what-is-kosher#what-are-the-rules> (Last Accessed 20-04-2023).
- [16] Boris Motik, Rob Shearer, Birte Glimm, Giorgos Stoilos, and Ian Horrocks. Hermit owl reasoner. <http://www.hermit-reasoner.com/> (Last Accessed 20-04-2023).
- [17] The University of Manchester. Research ethics, Jul 2016. <https://www.staffnet.manchester.ac.uk/rbe/ethics-integrity/ethics/> (Last Accessed 20-04-2023).
- [18] The University of Manchester. Owl api documentation, 2023. [http://owlcs.github.io/owlapi/apidocs\\_4/index.html](http://owlcs.github.io/owlapi/apidocs_4/index.html) (Last Accessed 20-04-2023).
- [19] Oracle. Package javax.swing, Jun 2020. <https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html> (Last Accessed 20-04-2023).

- [20] Thomas Pellissier Tanon, Gerhard Weikum, and Fabian Suchanek. Yago 4: A reason-able knowledge base. In *The Semantic Web: 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31–June 4, 2020, Proceedings 17*, pages 583–596. Springer, Springer International Publishing, 2020.
- [21] Brett Porter, Jason van Zyl, and Olivier Lamy. Welcome to apache maven, Apr 2023. <https://maven.apache.org/> (Last Accessed 20-04-2023).
- [22] Michael Roskams. *Religion, England and Wales: Census 2021*. Office for National Statistics, Nov 2022.
- [23] Ben Shneiderman and Catherine Plaisant. *Designing the user interface: Strategies for effective human-computer interaction*. Pearson Education India, 2010.
- [24] Scott H. Sicherer and Hugh A. Sampson. Food allergy: A review and update on epidemiology, pathogenesis, diagnosis, prevention, and management. *Journal of Allergy and Clinical Immunology*, 141(1):41–58, Nov 2017.
- [25] Carolyn Snyder. *Paper prototyping: The fast and easy way to design and refine user interfaces*. Morgan Kaufmann, 2003.
- [26] Kavitha Srinivas. Owl reasoning in the real world: Searching for godot. *Description Logics*, 477, 2009.
- [27] Fabian M. Suchanek, Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, and Pierre Senellart. *Putting into Practice: Ontologies in Practice*, page 236–239. Cambridge University Press, 2011.
- [28] Timo Weithöner, Thorsten Liebig, Marko Luther, Sebastian Böhm, Friedrich Von Henke, and Olaf Noppens. Real-world reasoning with owl. In *The Semantic Web: Research and Applications: 4th European Semantic Web Conference, ESWC 2007, Innsbruck, Austria, June 3-7, 2007. Proceedings 4*, pages 296–310. Springer, 2007.
- [29] Helen West. The 8 most common food allergies, Apr 2023. <https://www.healthline.com/nutrition/common-food-allergies> (Last Accessed 20-04-2023).
- [30] Nils-Gerrit Wunsch. *How often do you dine out?* Statista, Jun 2022.

## Appendix A

### User Study 1 Tasks and Questions

Task
Add a Dairy Ingredient “Parmesan” to the menu, with 76 calories. Contains allergens: Milk.
Add Component “ChickenParmesan” to system. Contains ingredients: ChickenBreast, Parmesan and Tomato.
Add Dish “ChickenParmesanBurger” to system. Contains components: BurgerBun, Chicken Parmesan.
Search for the “ChickenParmesanBurger” Dish.
Remove the “ChickenParmesanBurger” Dish.
Remove the “ChickenParmesan” Component.
Remove the “Parmesan” Ingredient.
Add new allergen “Mustard”.
Create a new dish: Steak Roll. Ingredients: BurgerBun, Steak, English Mustard (contains “Mustard” allergen), Red Onion and Rocket.
Remove the Steak Roll dish.

Table A.1: Study 1 - Tasks for restaurant owners

<b>Task</b>
Find how many calories are in a BeefBurger.
Find all dishes that contain less than 600 calories.
Find all dishes suitable for vegetarians.
Find all dishes that are Halal.
Find all dishes that contain no egg or milk.
Find dishes that are Halal, suitable for vegetarians, contain no egg and are under 700 calories.
Find how many calories are in a PestoBurger with added SmokedBacon and Cheddar, but with the Tomato removed.
Find which allergens are contained in the TributeBurger.
Search with the calories turned off.
Find which allergens are contained in a BeefBurger with the BeefPatty removed.

Table A.2: Study 1 - Tasks for customers

<b>Question</b>
How fast was the system to respond? (1-5)
How intuitive was the system? (1-5)
How easy was the system to use? (1-5)
Was there anything you wish was explained to you better before use?
Where do you think the system could be improved?
General Feedback.

Table A.3: Study 1 - Questions for all participants. Questions marked “(1-5)” were marked on a Likert scale.

## Appendix B

### User Study 1 Results

Task	Participant 1	Participant 2	Participant 3
Add a Dairy Ingredient “Parmesan” to the menu, with 76 calories. Contains allergens: Milk.	✓	✓	✓
Add Component “ChickenParmesan” to system. Contains ingredients: Chicken-Breast, Parmesan and Tomato.	✗	✗	✓
Add Dish “ChickenParmesanBurger” to system. Contains components: BurgerBun, Chicken Parmesan.	✓	✓	✓
Search for the “ChickenParmesanBurger” Dish.	✓	✓	✓
Remove the “ChickenParmesanBurger” Dish.	✓	✓	✓
Remove the “ChickenParmesan” Component.	✓	✓	✓
Remove the “Parmesan” Ingredient.	✓	✓	✓
Add new allergen “Mustard”.	✓	✓	✓
Create a new dish: Steak Roll. Ingredients: BurgerBun, Steak, English Mustard (contains “Mustard” allergen), Red Onion and Rocket.	✗	✓	✓
Remove the Steak Roll dish.	✓	✓	✓

Table B.1: Study 1 - Task results for restaurant owner

Task	Participant 1	Participant 2	Participant 3
Find how many calories are in a Beef-Burger.	✓	✓	✓
Find all dishes that contain less than 600 calories.	✓	✓	✓
Find all dishes suitable for vegetarians.	✓	✓	✓
Find all dishes that are Halal.	✓	✓	✓
Find all dishes that contain no egg or milk.	✓	✓	✓
Find dishes that are Halal, suitable for vegetarians, contain no egg and are under 700 calories.	✓	✓	✓
Find how many calories are in a Pesto-Burger with added SmokedBacon and Cheddar, but with the Tomato removed.	✗	✗	✗
Find which allergens are contained in the TributeBurger.	✓	✓	✓
Search with the calories turned off.	✓	✓	✓
Find which allergens are contained in a BeefBurger with the BeefPatty removed.	✗	✗	✗

Table B.2: Study 1 - Task results for customer

Question	Participant 1	Participant 2	Participant 3
How fast was the system to respond? (1-5)	5	4	5
How intuitive was the system? (1-5)	4	4	4
How easy was the system to use? (1-5)	4	4	4
Was there anything you wish was explained to you better before use?	Ingredient, Component, and Dish hierarchy.	Hold Ctrl to select multiple. Ingredient, Component, and Dish.	Ingredient, Component, Dish hierarchy.
Where do you think the system could be improved?	Ctrl to select multiple is not intuitive. Edit an ingredient or component easier.	Search bar at top of lists. Double click not intuitive.	No page to see information about an ingredient.
General Feedback.	Liked search by max calories, clear layout and organisation.	Easy to use.	Liked: layout clear, automatic calculation of Vegan etc. dishes.

Table B.3: Study 1 - Question responses for restaurant owners. Questions marked “(1-5)” were marked on a Likert scale.

<b>Question</b>	<b>Participant 1</b>	<b>Participant 2</b>	<b>Participant 3</b>
How fast was the system to respond? (1-5)	4	5	5
How intuitive was the system? (1-5)	4	4	4
How easy was the system to use? (1-5)	5	5	5
Was there anything you wish was explained to you better before use?	Ctrl to select multiple in list.	Ctrl to select multiple in list.	Ctrl to select multiple in list.
Where do you think the system could be improved?	Stick calorie slider to minor ticks.	Double click not intuitive.	Add gluten free options.
General Feedback.	None given.	Very usable, especially in context.	None given.

Table B.4: Study 1 - Question responses for customers. Questions marked “(1-5)” were marked on a Likert scale.

## Appendix C

### User Study 2 Tasks and Questions

Task
Add a Dairy Ingredient “Parmesan” to the menu, with 76 calories. Contains allergens: Milk.
Add Component “ChickenParmesan” to system. Contains ingredients: ChickenBreast, Parmesan and Tomato.
Add Dish “ChickenParmesanBurger” to system. Contains components: BurgerBun, Chicken Parmesan.
Search for the “ChickenParmesanBurger” Dish.
Remove the “ChickenParmesanBurger” Dish.
Remove the “ChickenParmesan” Component.
Remove the “Parmesan” Ingredient.
Edit the calories in the BeefPatty ingredient from 451 to 551.
Edit the name of the “Cheddar” ingredient to “CheddarCheese”.
Edit BeefBurgerFilling component to also include CheddarCheese.
Edit the BeefBurger dish to state that it is Halal.
Add new allergen “Mustard”.
Create a new dish: Steak Roll. Ingredients: BurgerBun, Steak, English Mustard (contains “Mustard” allergen), Red Onion and Rocket.
Remove the Steak Roll dish.
Attempt to add an ingredient with an empty name.
Attempt to add a component with no ingredients.

Table C.1: Study 2 - Tasks for restaurant owners

Task
Find how many calories are in a BeefBurger.
Find all dishes that contain less than 600 calories.
Find all dishes suitable for vegetarians.
Find all dishes that are Halal.
Find all dishes that have a gluten free alternative.
Find all dishes that contain no egg or milk.
Find dishes that are Halal, suitable for vegetarians, contain no egg and are under 700 calories.
Find how many calories are in a PestoBurger with added SmokedBacon and Cheddar, but with the Tomato removed.
Find which allergens are contained in the TributeBurger.
Search with the calories turned off.
Find which allergens are contained in a BeefBurger with the BeefPatty removed.

Table C.2: Study 2 - Tasks for customers

Question
How fast was the system to respond? (1-5)
How intuitive was the system? (1-5)
How easy was the system to use? (1-5)
How easy was it to pick up after not using the system for a month? (1-5) *
Was there anything you wish was explained to you better before use?
What did you like about the system?
Where do you think the system could be improved?
Has the system improved since the previous study? *
Did the system offer informative feedback to your actions?
Was the system consistent throughout?
Did you feel in control of the system?
Did you feel the need to remember lots of information?
Did you feel that you could easily reverse changes you made to the system?
General Feedback.

Table C.3: Study 2 - Questions for all participants. Questions marked “(1-5)” were marked on a Likert scale. Questions marked with an asterisk were only asked to participants who acted as a restaurant owner.

## **Appendix D**

### **User Study 2 Results**

Task	Participant 1	Participant 2	Participant 3
Add a Dairy Ingredient “Parmesan” to the menu, with 76 calories. Contains allergens: Milk.	✓	✓	✓
Add Component “ChickenParmesan” to system. Contains ingredients: Chicken-Breast, Parmesan and Tomato.	✓	✓	✓
Add Dish “ChickenParmesanBurger” to system. Contains components: BurgerBun, Chicken Parmesan.	✓	✓	✓
Search for the “ChickenParmesanBurger” Dish.	✓	✓	✓
Remove the “ChickenParmesanBurger” Dish.	✓	✓	✓
Remove the “ChickenParmesan” Component.	✓	✓	✓
Remove the “Parmesan” Ingredient.	✓	✓	✓
Edit the calories in the BeefPatty ingredient from 451 to 551.	✓	✓	✓
Edit the name of the “Cheddar” ingredient to “CheddarCheese”.	✓	✓	✓
Edit BeefBurgerFilling component to also include CheddarCheese.	✓	✓	✓
Edit the BeefBurger dish to state that it is Halal.	✓	✓	✓
Add new allergen “Mustard”.	✓	✓	✓
Create a new dish: Steak Roll. Ingredients: BurgerBun, Steak, English Mustard (contains “Mustard” allergen), Red Onion and Rocket.	✓	✓	✓
Remove the Steak Roll dish.	✓	✓	✓
Attempt to add an ingredient with an empty name.	✓	✓	✓
Attempt to add a component with no ingredients.	✓	✓	✓

Table D.1: Study 2 - Task results for restaurant owners

Task	Participant 1	Participant 2	Participant 3
Find how many calories are in a Beef-Burger.	✓	✓	✓
Find all dishes that contain less than 600 calories.	✓	✓	✓
Find all dishes suitable for vegetarians.	✓	✓	✓
Find all dishes that are Halal.	✓	✓	✓
Find all dishes that have a gluten free alternative.	✓	✓	✓
Find all dishes that contain no egg or milk.	✓	✓	✓
Find dishes that are Halal, suitable for vegetarians, contain no egg and are under 700 calories.	✓	✓	✓
Find how many calories are in a Pesto-Burger with added SmokedBacon and Cheddar, but with the Tomato removed.	✓	✓	✓
Find which allergens are contained in the TributeBurger.	✓	✓	✓
Search with the calories turned off.	✓	✓	✓
Find which allergens are contained in a BeefBurger with the BeefPatty removed.	✓	✓	✓

Table D.2: Study 2 - Task results for customers

Question	Participant 1	Participant 2	Participant 3
How fast was the system to respond? (1-5)	5	4	5
How intuitive was the system? (1-5)	5	5	5
How easy was the system to use? (1-5)	5	5	5
How easy was it to pick up after not using the system for a month? (1-5)	5	5	5
Was there anything you wish was explained to you better before use?	No	No	No
What did you like about the system?	Ability to search for existing entities, and edit entities directly.	Logical, intuitive, calculates information for you. Easy to use and obvious how to use interface.	Laid out logically and clearly, edit directly from search was useful. Fast and responsive.
Where do you think the system could be improved?	Components seem redundant, could they be removed? Add prices.	Not clear what components should be. Text search for lists to find item.	Ability to delete from edit page.
Has the system improved since the previous study?	Yes	Yes	Yes
Did the system offer informative feedback to your actions?	Yes	Yes	Yes
Was the system consistent throughout?	Yes	Yes	Yes
Did you feel in control of the system?	Yes	Yes	Yes
Did you feel the need to remember lots of information?	No	No	No
Did you feel that you could easily reverse changes you made to the system?	Yes	Yes	Yes
General Feedback.	Logical process, easy to visualise. All pages look similar, hard to distinguish pages other than title.	Multiple selection is more clear. Beef is not an allergen, could be misleading.	Editing is useful, multiple selection has improved. Clearly laid out, easy to use and intuitive.

Table D.3: Study 2 - Question responses from restaurnat owners. Questions marked “(1-5)” were marked on a Likert scale.

<b>Question</b>	<b>Participant 1</b>	<b>Participant 2</b>	<b>Participant 3</b>
How fast was the system to respond? (1-5)	5	5	5
How intuitive was the system? (1-5)	5	5	4
How easy was the system to use? (1-5)	5	5	5
Was there anything you wish was explained to you better before use?	No.	No.	No.
What did you like about the system?	Very easy to use, extensive dietary options. Important to be able to turn off calories.	Very quick, customers don't want to be waiting. Multiple selection has improved. Contains everything necessary and nothing more.	New gluten free option. Search by calories so you don't find out after choosing dish. Allergens were clearly presented, felt in control as someone who looks for that information.
Where do you think the system could be improved?	Forgot about settings, move them to search page? Sentence to explain whether filter includes or excludes selection.	Calorie slider can't click to move, you have to drag. Could view specific gluten free alternative ingredients etc.	Put settings in search page, forgot they were there until asked.
Did the system offer informative feedback to your actions?	Yes	Yes	Yes
Was the system consistent throughout?	Yes	Yes	Yes
Did you feel in control of the system?	Yes	Yes	Yes
Did you feel the need to remember lots of information?	No.	No.	No.
Did you feel that you could easily reverse changes you made to the system?	Yes	Yes	Yes
General Feedback.	None given.	Very intuitive, covered all bases.	Easy to use and flexible.

Table D.4: Study 2 - Question responses from customers. Questions marked “(1-5)” were marked on a Likert scale.

## Appendix E

### Final Ontology structure

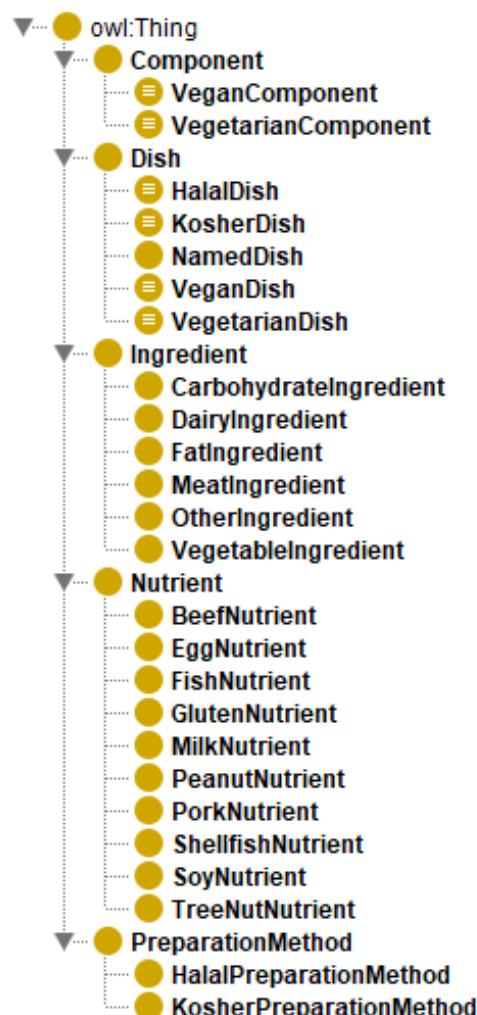


Figure E.1: Final ontology structure