

# Artificial Intelligence Fundamentals

## Master in Artificial Intelligence

### Practicum 2: Ontologies

Academic year 2025/26

#### P2.1: Editing ontologies using WebProtégé

During this practice you will learn to use a tool to edit an ontology:

- Food ontology - Ontology of Fast Food Facts (OFFF):
  - o Publication: <https://bmcmmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-021-01636-1>
  - o GitHub: <https://github.com/UTHealth-Ontology/OFFF/>
- Ontology tool: WebProtégé: <https://webprotege.stanford.edu/>

First, you will study the publication about the OFFF ontology in order to understand the structure of this ontology. Second, you will download the food ontology (OFFF) from the GitHub project as **offf.owl** file. Third, you create an account at **Web Protégé** (<https://webprotege.stanford.edu/>), use the tool to import the downloaded ontology (offf.owl) and edit the ontology by adding few components.

#### Food ontology (OFFF)

First you should understand the structure of the food ontology in order to be able to modify it.

#### OFFF publication

Let's have a look at the publication: "The ontology of fast food facts: conceptualization of nutritional fast food data for consumers and semantic web applications", *BMC Medical Informatics and Decision Making* volume 21, Article number: 275 (2021) [[web](#)] [[pdf](#)].

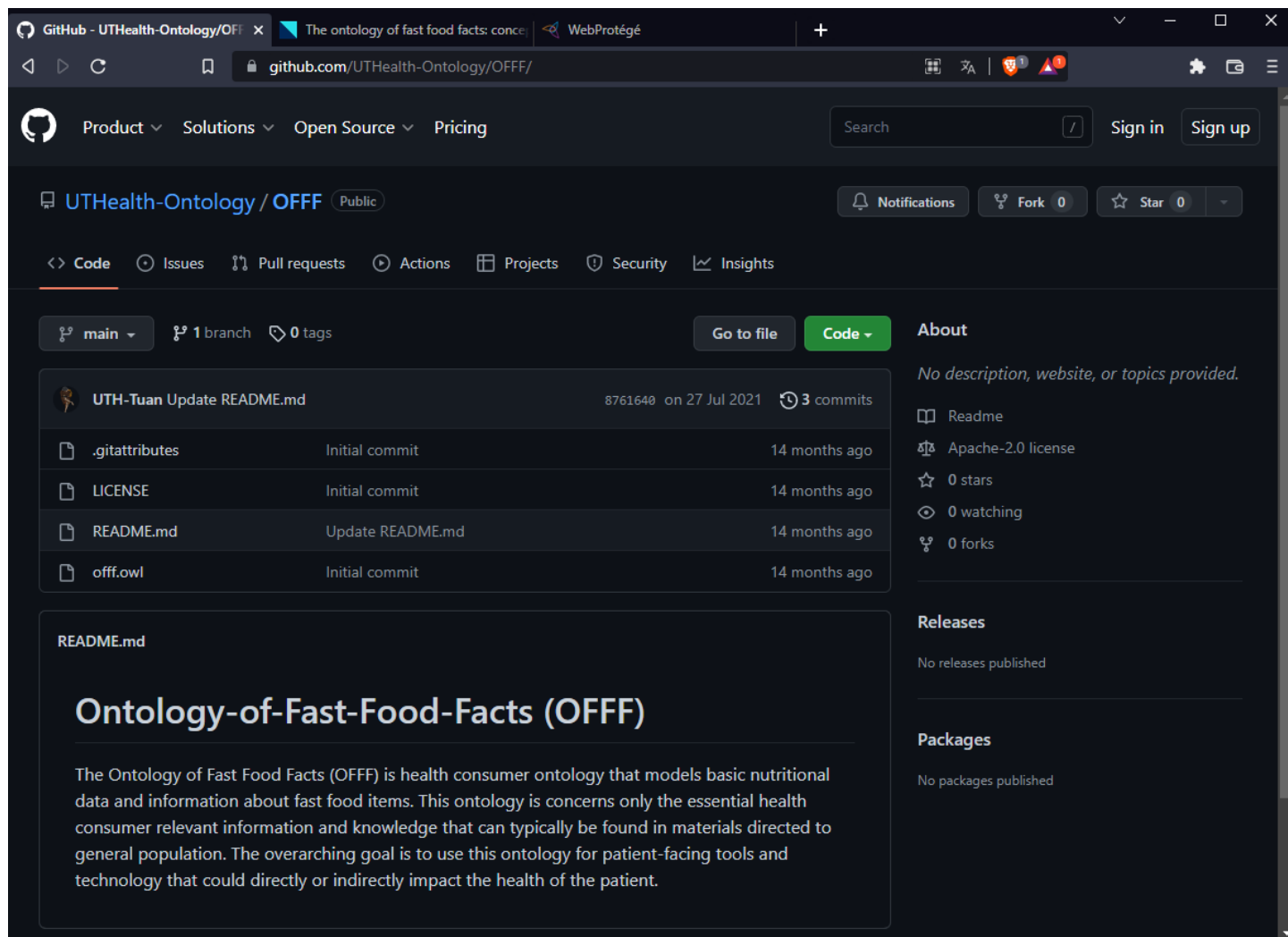
The screenshot shows a web browser displaying the article page on the BMC Medical Informatics and Decision Making website. The article title is "The ontology of fast food facts: conceptualization of nutritional fast food data for consumers and semantic web applications" by Muhammad Amith, Chidinma Onye, Tracey Ledoux, Grace Xiong & Cui Tao. The article is published in BMC Medical Informatics and Decision Making, volume 21, article number 275 (2021). The page includes a "Download PDF" button and a table of contents with sections: Abstract, Background, Methods, Results, Discussion, Conclusion, and Availability of data and materials. The article is categorized as "Research" and "Open Access", published on 09 November 2021. It has 1689 accesses, 1 citation, and 9 altmetric mentions.

Let's read the Background, Methods and Results from the publication (see <https://bmcmmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-021-01636-1>).

We are continuing to read about the “Nutritional components”, “Allergens”, “Ingredients” and “Fast food concept”. Let's comment Figures 1 and 2 from paper in order to understand the relationships between specific classes.

### Download the OFF ontology

You can download in your computer the food ontology OFFF using the [GitHub project](#) or directly the link for the [offf.owl file](#).



The screenshot displays the GitHub repository page for **UTHealth-Ontology / OFFF**. The repository is public and has 0 stars, 0 forks, and 0 releases. The **Code** tab is selected, showing a file list with the following files and their commit history:

File	Commit	Time
.gitattributes	Initial commit	14 months ago
LICENSE	Initial commit	14 months ago
README.md	Update README.md	14 months ago
offf.owl	Initial commit	14 months ago

The **README.md** file is selected, showing the following content:

## Ontology-of-Fast-Food-Facts (OFFF)

The Ontology of Fast Food Facts (OFFF) is health consumer ontology that models basic nutritional data and information about fast food items. This ontology is concerns only the essential health consumer relevant information and knowledge that can typically be found in materials directed to general population. The overarching goal is to use this ontology for patient-facing tools and technology that could directly or indirectly impact the health of the patient.

After you understand the structure of the OFFF ontology and you have it on your computer as an OWL file, you can open the WebProtégé ontology editor.

Protégé (<https://webprotege.stanford.edu/>) is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies. It is offering:

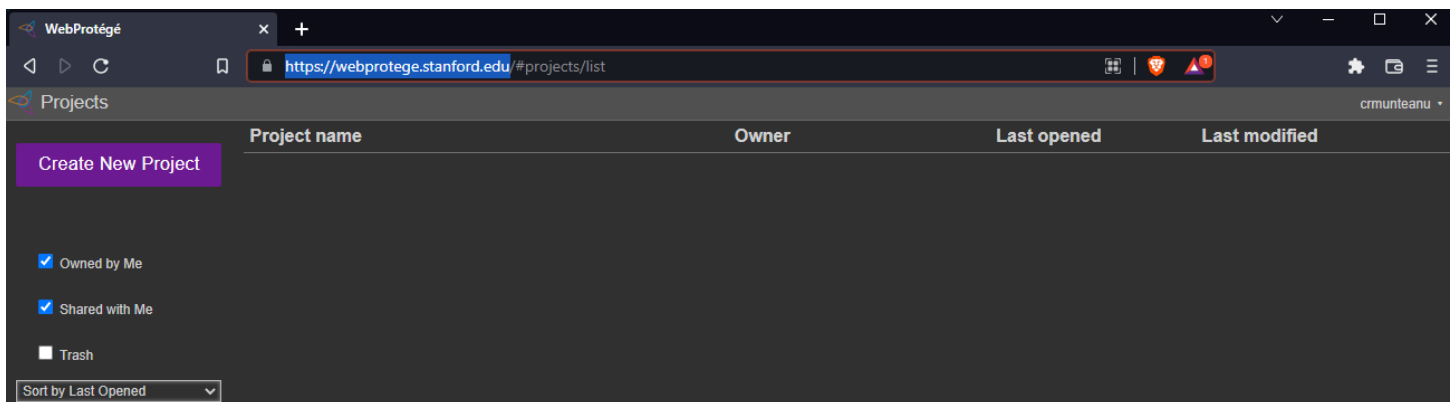
- ✓ W3C standards compliant
- ✓ Customizable user interface
- ✓ Visualization support
- ✓ Ontology refactoring support
- ✓ Direct interface to reasoners
- ✓ Highly pluggable architecture
- ✓ Cross compatible with WebProtégé

WebProtégé is a free, open-source collaborative ontology development environment for the Web. It is developed by the Protege team in the Biomedical Informatics Research Group (BMIR) at Stanford University, California, USA.

Please check more information at <https://protegewiki.stanford.edu/wiki/WebProtegeUsersGuide>.

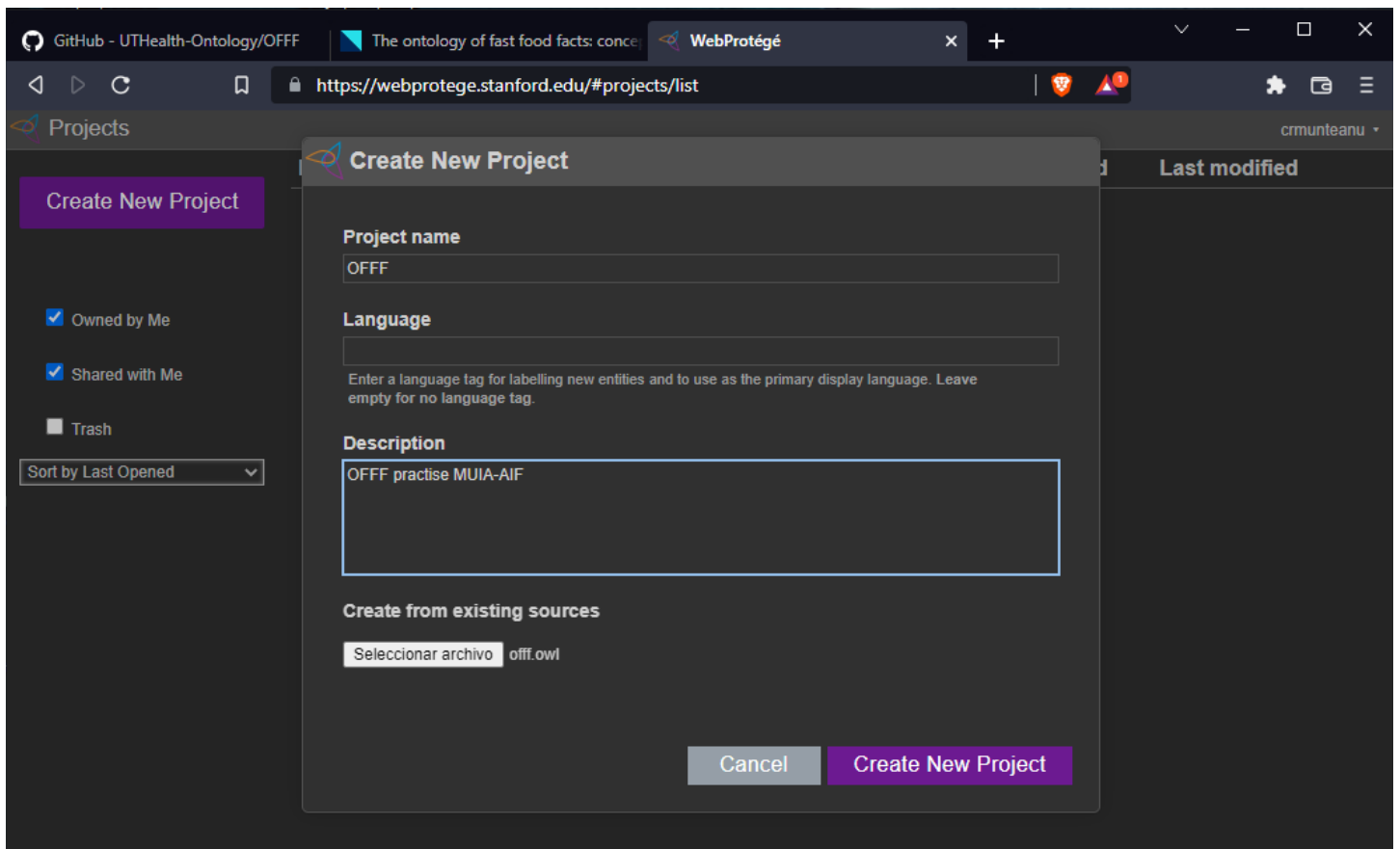
Create a project with WebProtégé by importing an external ontology

After you will create a new account at WebProtégé, login to your projects / ontologies. If it is a new account, you should have a screen with no project (no ontology).

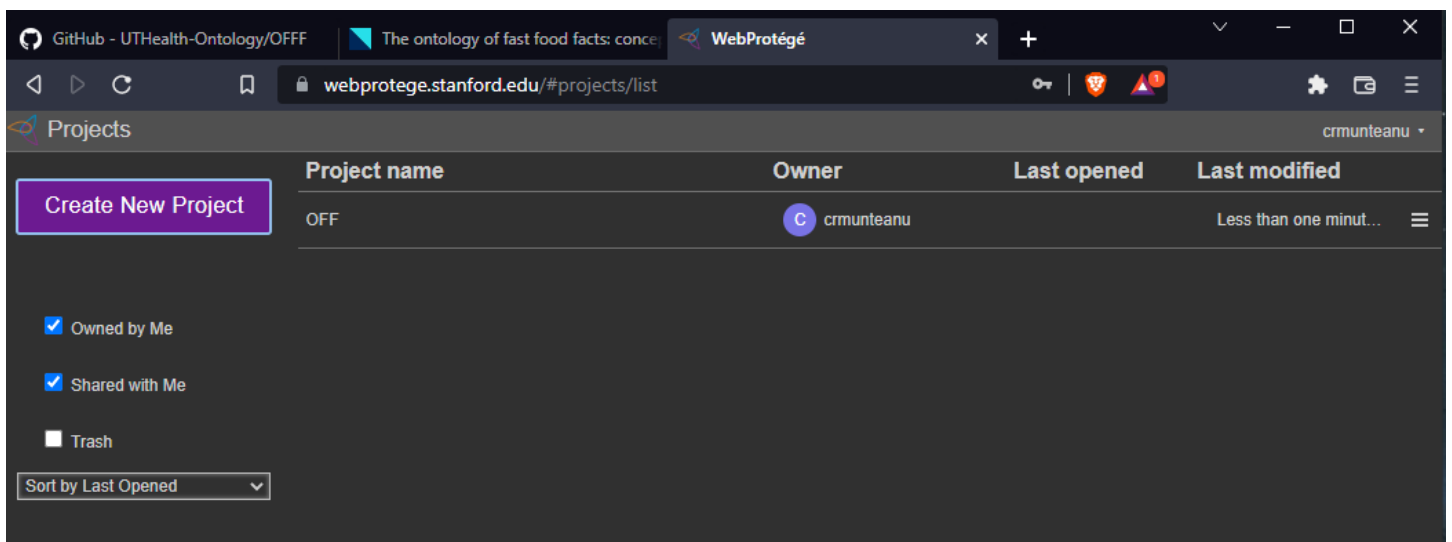


**NOTE: before to use off.owl file, remove the line “Import(<http://www.w3.org/2004/02/skos/core>)”.**

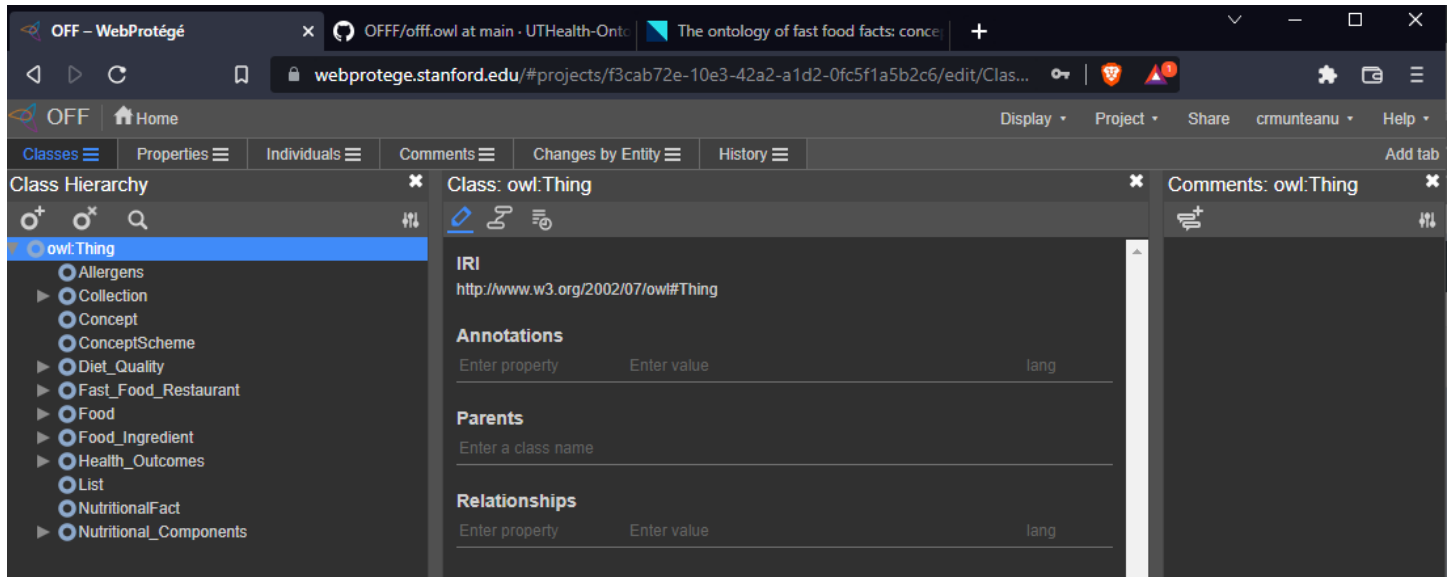
First, you need to import the downloaded OFFF ontology from your computer by creating a new project (ontology) by pressing “Create New Project”. You will introduce a name for your project (such as “OFFF”), a description (it can be “OFFF practise MUIA-AIF”) and select the file offf.owl from your computer in the section “Create from existing sources”. This way, you will import OFFF ontology into your new project with WebProtégé.



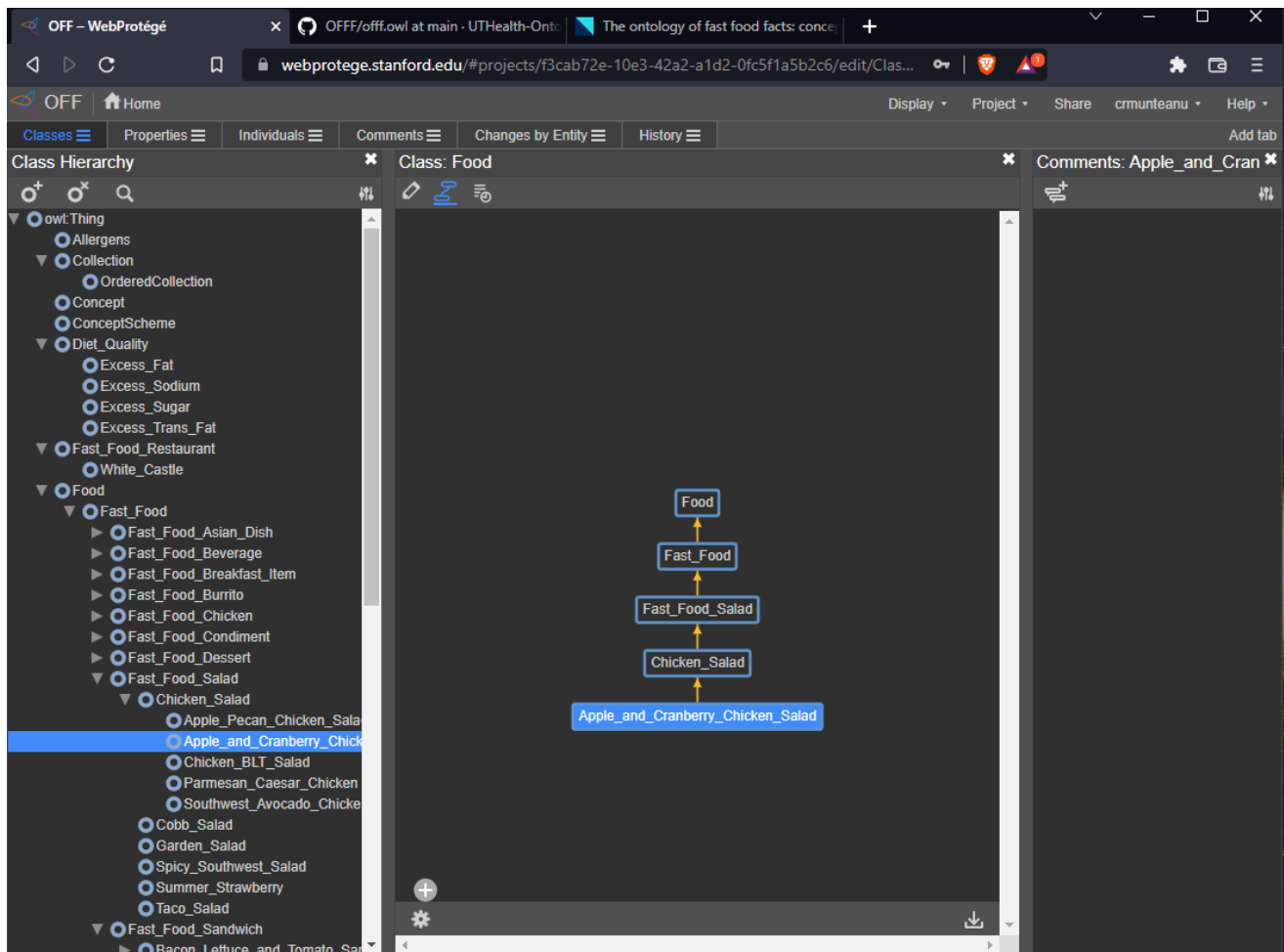
Now you should be able to find your new project into the projects list:



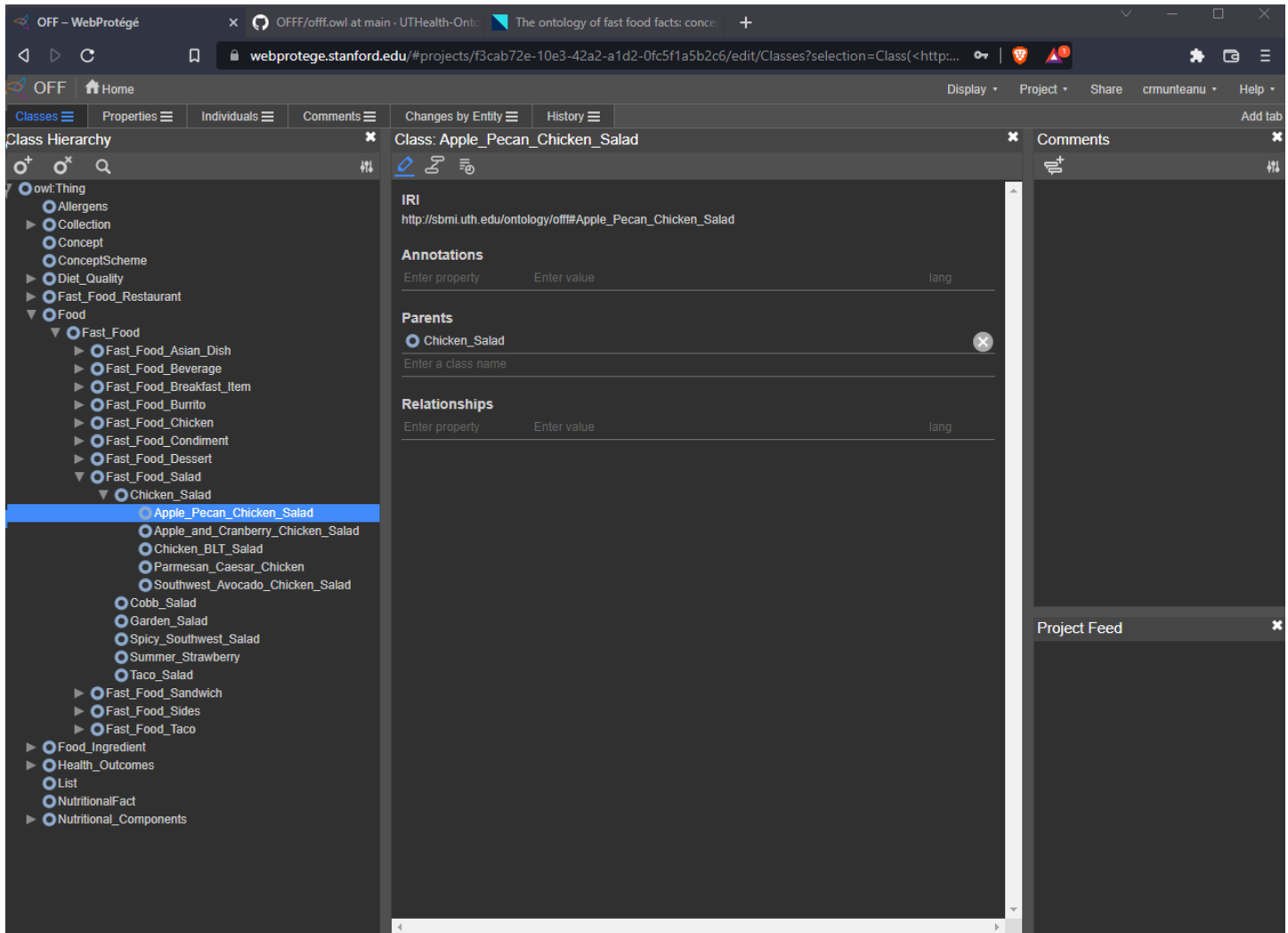
Open the imported ontology by clicking on OFFF or OFF project. Now you can see the structure of OFFF using different tabs. In the tab **Classes**, you have the *Class Hierarchy* (class-subclass).



By clicking on different classes, the subclasses are presented. Example: “Apple\_and\_Cranberry\_Chicken\_Salad” is a member of “Chicken\_Salad” that is member of “Fast\_Food\_Salad”, a member of “Fast\_Food” class that is member of the “Food”. You can see the **Entity Graph** for these classes:

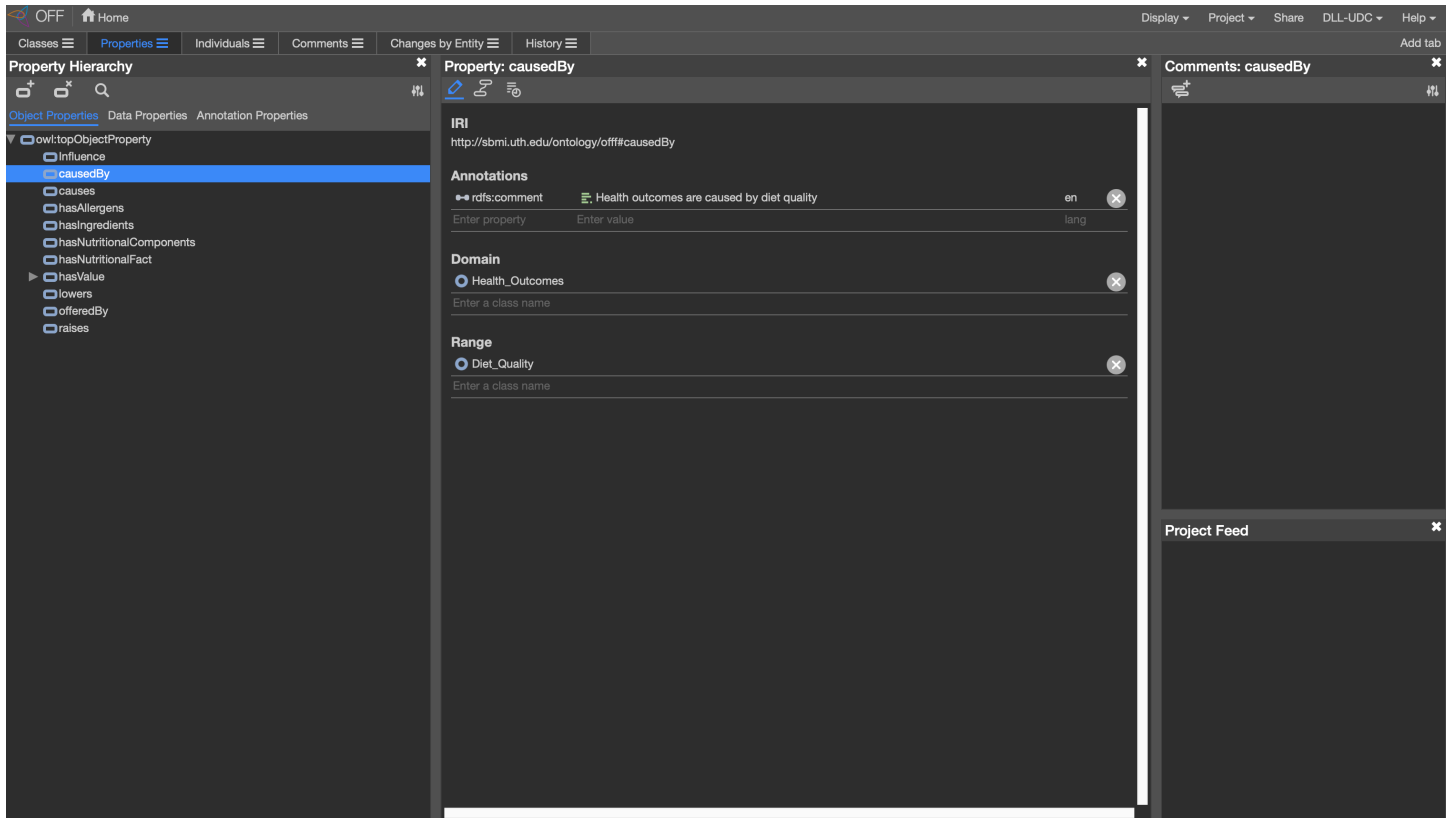


If you select “Apple\_Pecan\_Chicken\_Salad”, in the middle of the screen are presented the IRI and Parents for this class (**Details** of an item).

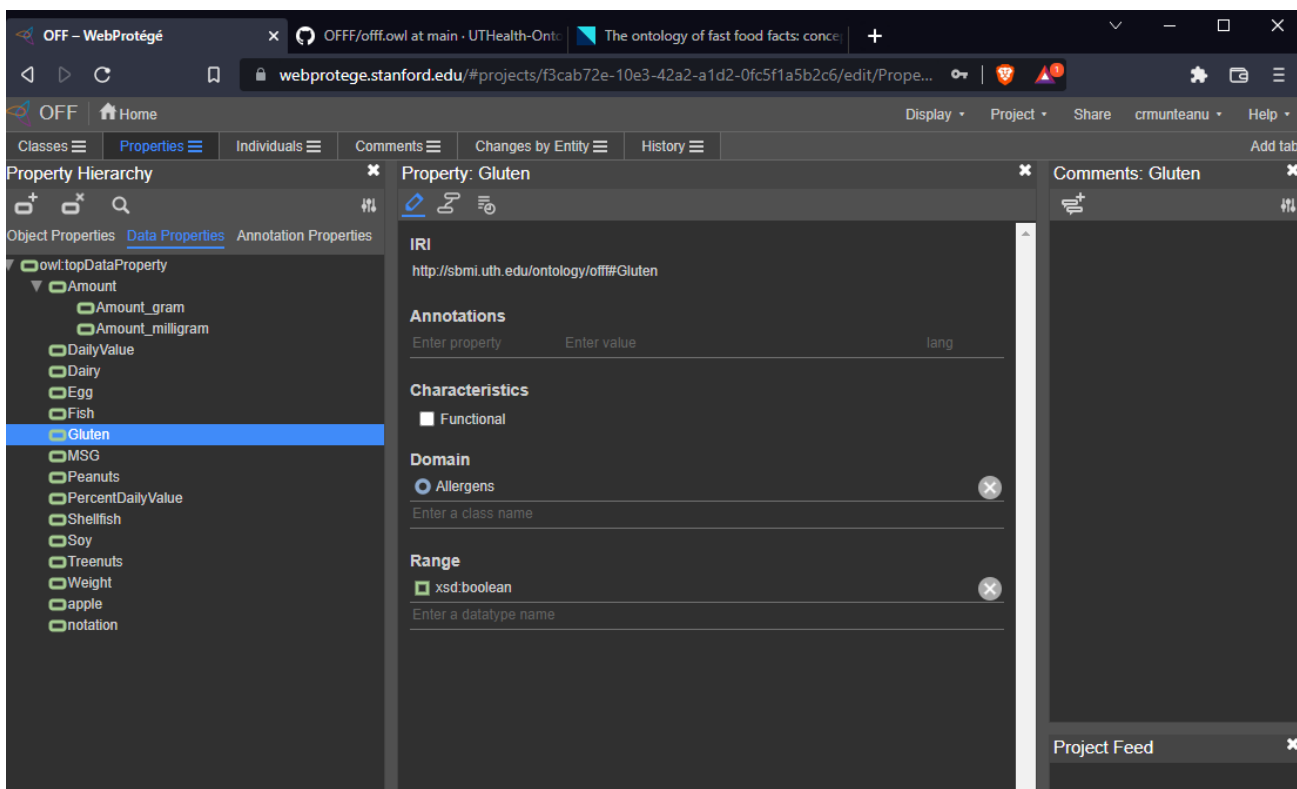


Remember that once classes have been created, they can be populated with individuals – as data on an underlying framework. The tab **Properties** is presenting Object, Data and Annotation Properties. Object properties connect pairs of classes. Object properties connect two individuals (a **subject** and **object**) with a **predicate**, while with data properties the **predicate** connects a single **subject** with some form of **attribute data**. Data properties have defined datatypes including string, integer, date, datetime, or boolean. Both object and data properties are often but not always defined to have a domain class, Allergens, specifying the class membership of the individuals serving as subjects for each object or data property statement. Object properties also may have a range class that specifies the class membership of the individuals serving as objects of the property predicate.

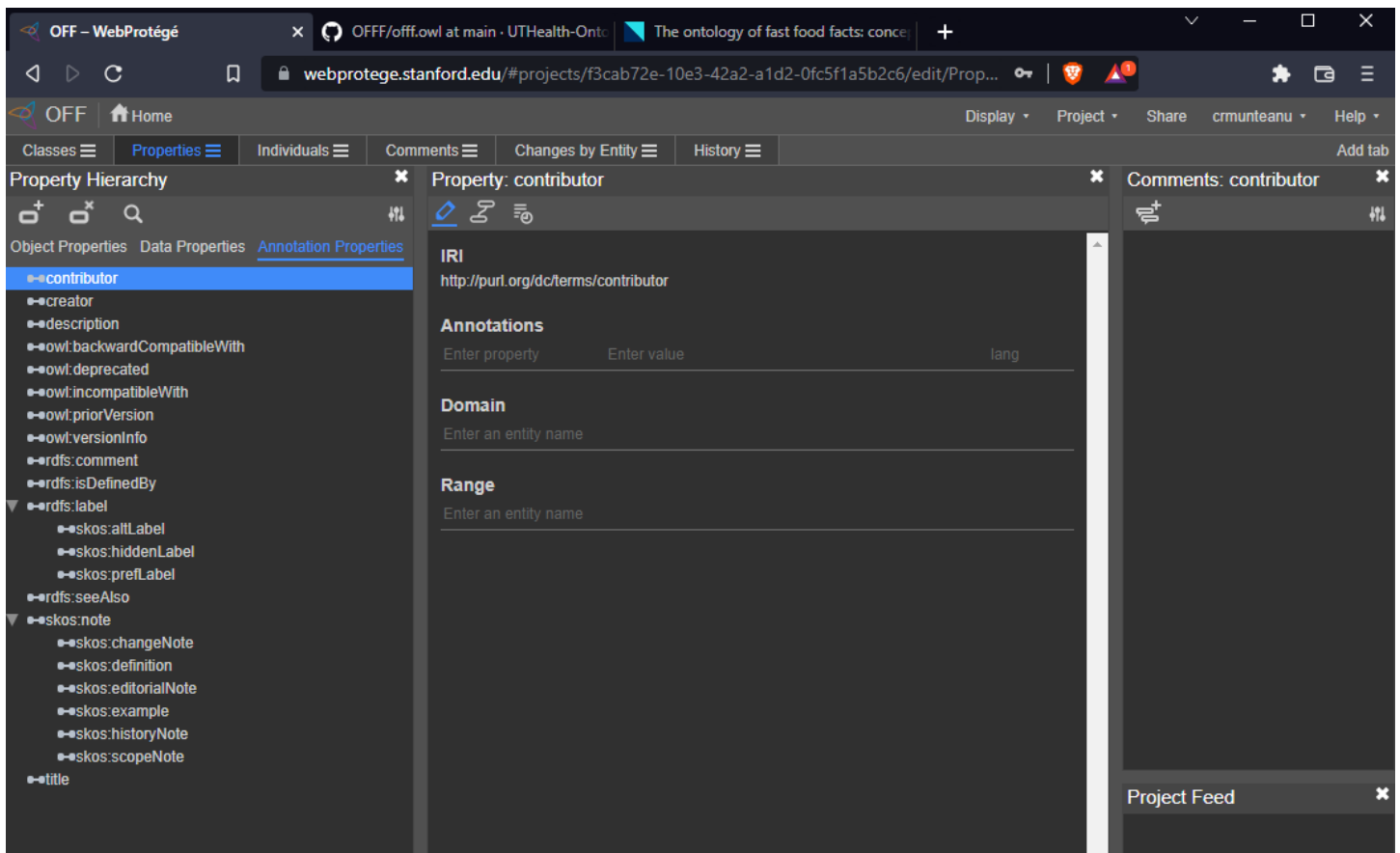
An example of an **Object Property** is “causedBy” that means “Health outcomes are caused by diet quality” and it relates a Health Outcomes to one of Diet\_Quality.



An example of **Data Property** is “Gluten” that is a property for any instance of “Allergens” as Boolean values (true or false).

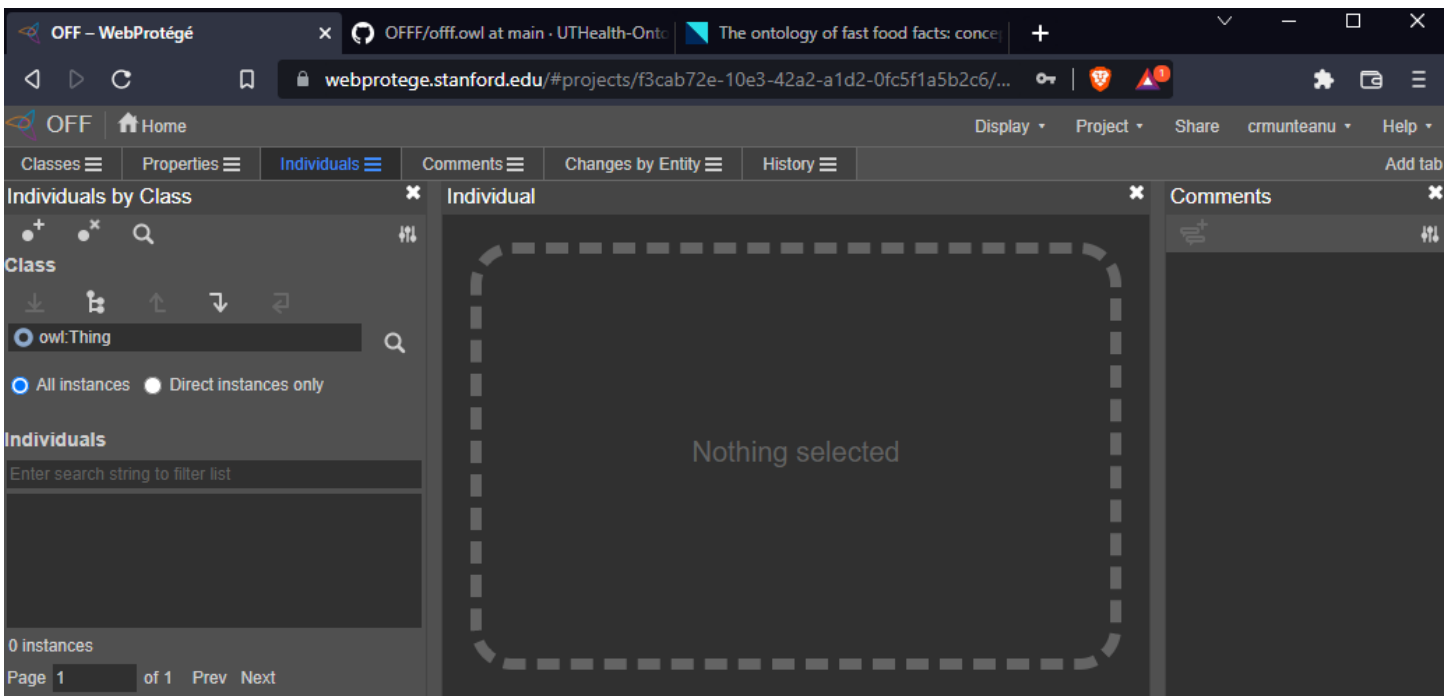


The ontology and any component of it can have annotations for a better understanding of the concepts and to relate the concepts with other ontologies. [Annotation properties](#) are used to place annotations on individuals, class names, property names, and ontology names. Ontology properties relate ontologies to other ontologies, in particular being used for importing information from other ontologies. Individual identifiers are used to refer to resources, and data literals are used to refer to data values.



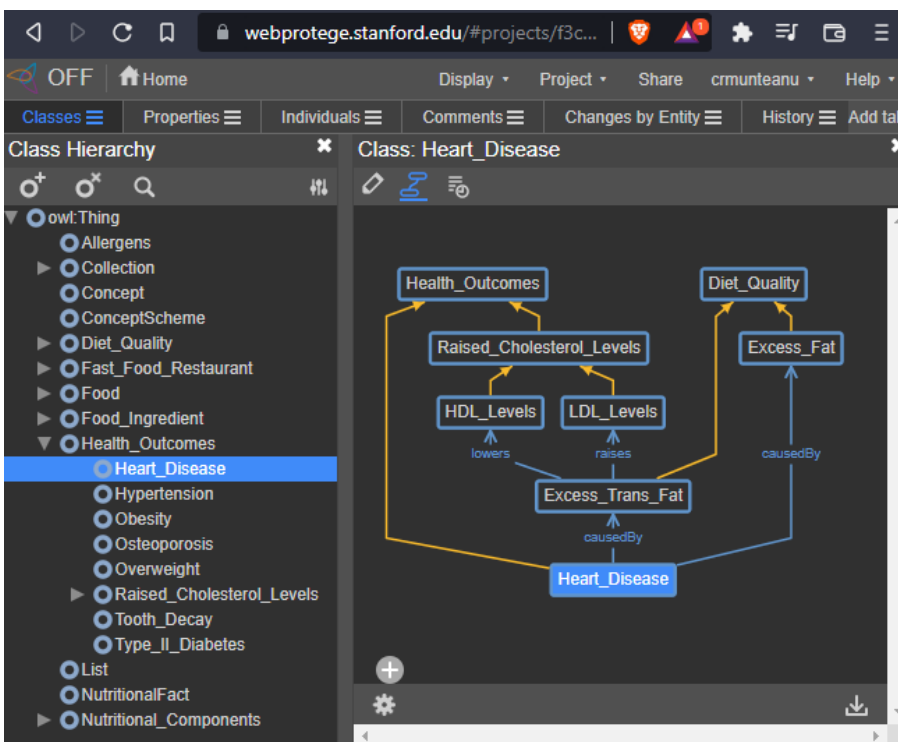
If you chose the tab [Individuals](#), you can observe that where is no data added, no individuals. We imported only the ontology definition/structure.



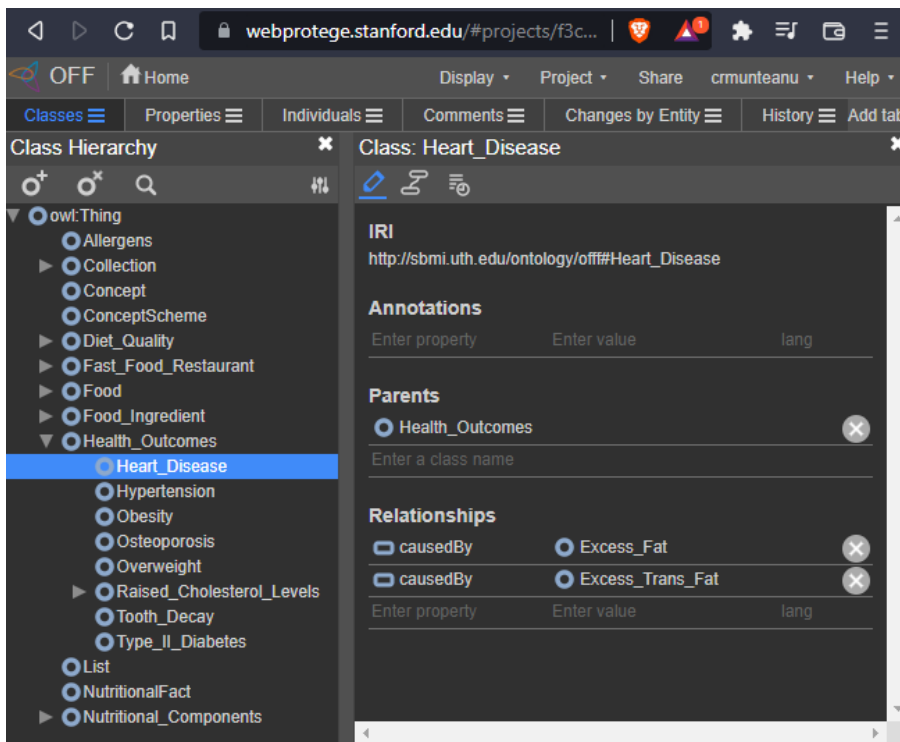


In tab **Comments** threads can be added about the ontology. In tab **Changes by Entity** and **History** are registered all the modifications of this ontology/project.

Let's see all classes and relationships for the class **Heart\_Disease**. If you double click on a class, you can jump to the specific class in the **Class Hierarchy** (localization of the class in ontology).

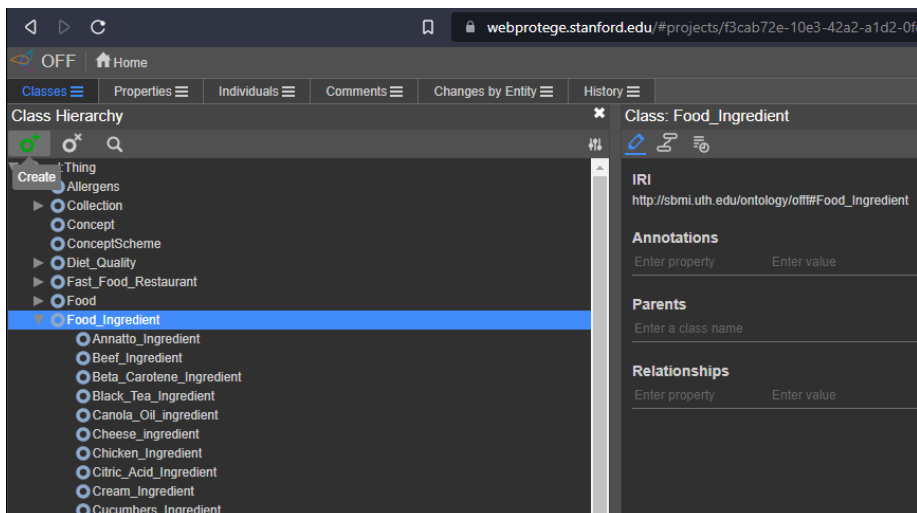


If you check the description of *Heart\_Disease* you can see the relationships *causedBy* and that it has the parent as *Health\_Outcomes*:

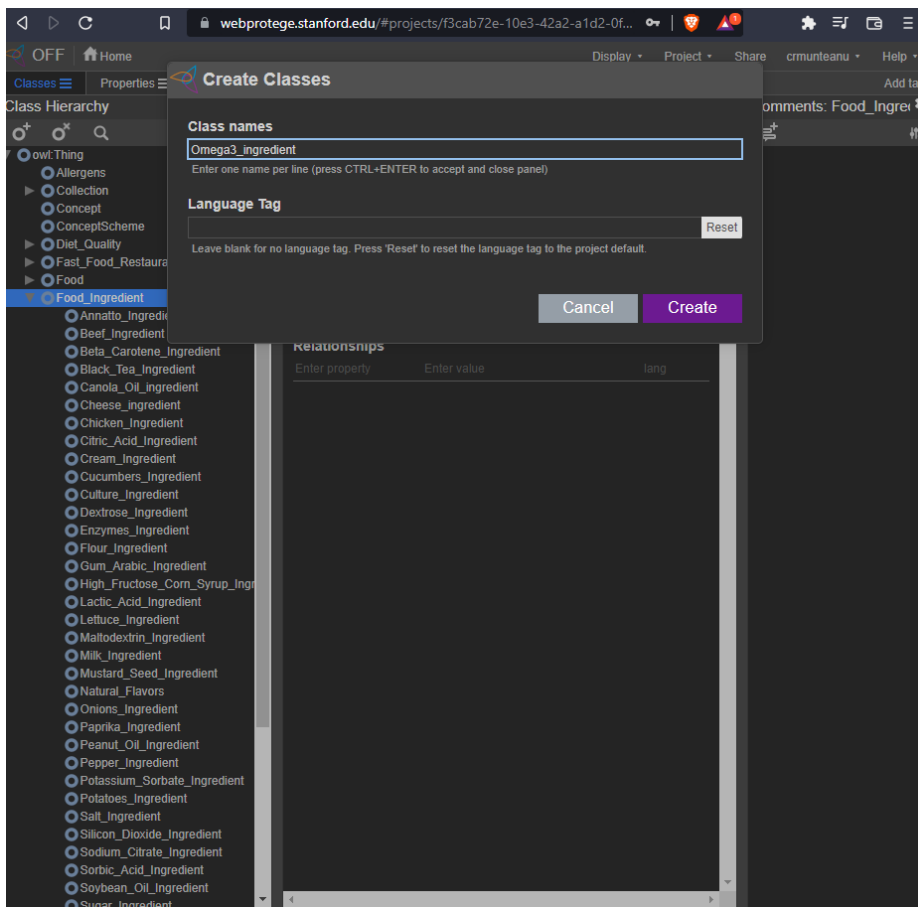


How to add a new class of OFFF

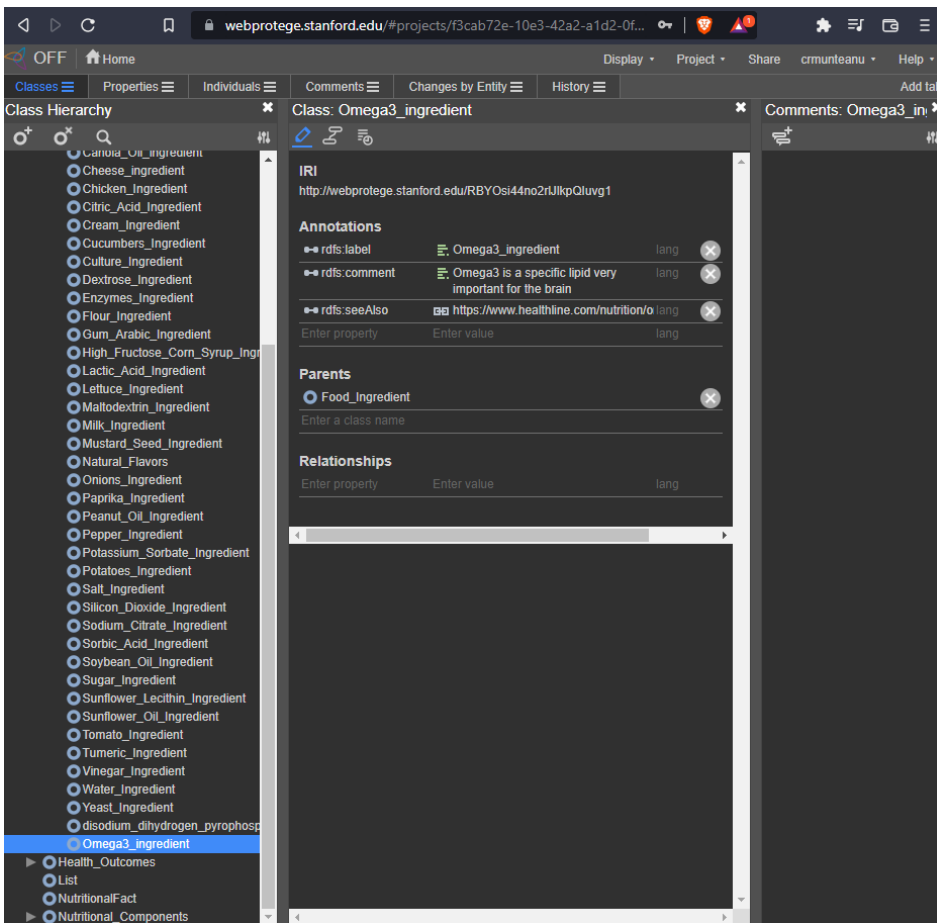
Let's add a new *Food\_Ingredient* as a subclass: *Omega3\_ingredient*. Select the class *Food\_Ingredient* and press **Create**:



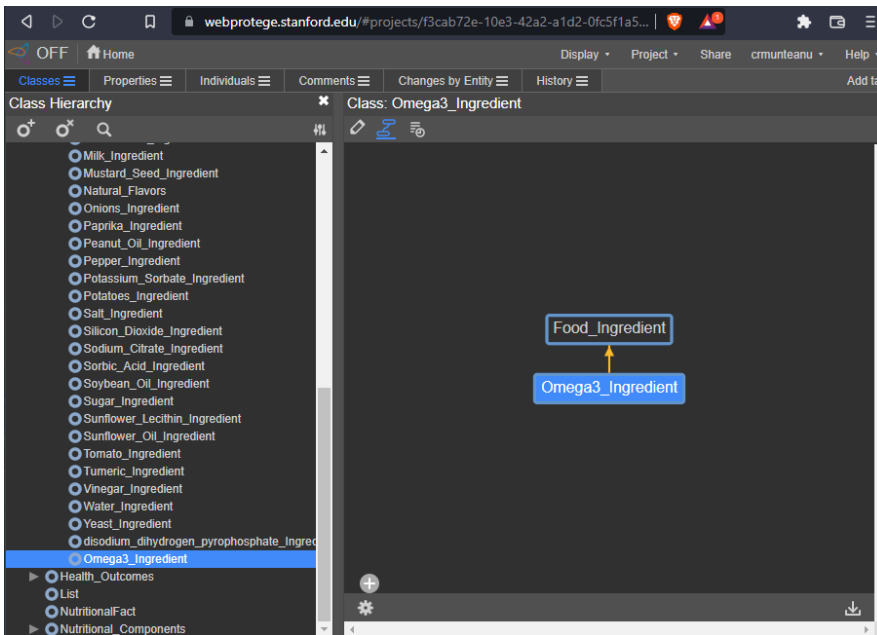
You will describe the class name such as *Omega3\_ingredient*. Thus, this new class will be a subclass of *Food\_Ingredient*:



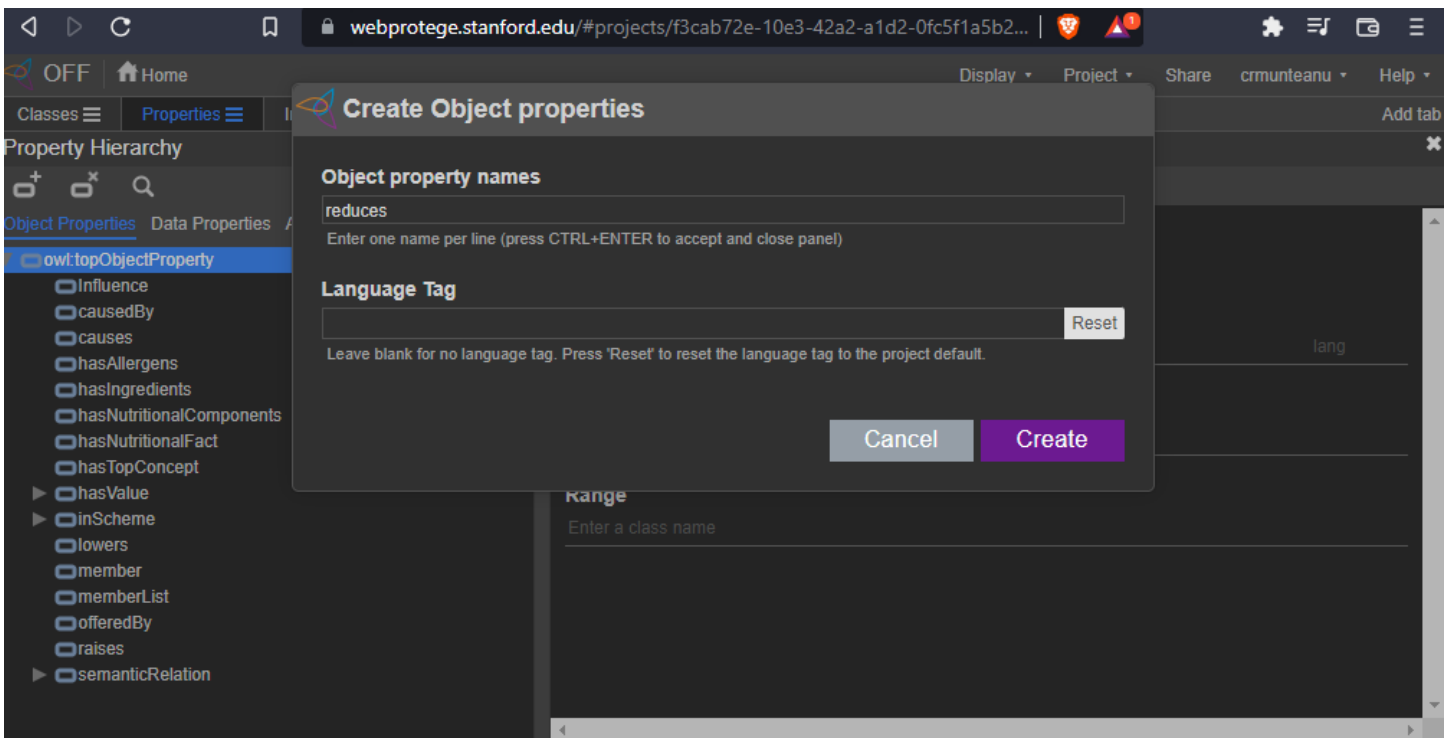
When you add a **new class**, the tool will add a unique ID such as **IRI** for the new class. You can add an **Annotations property** such as a *comment* (ex: “Omega3 is a specific lipid very important for the brain”). You can add *seeAlso* property as a web link: <https://www.healthline.com/nutrition/omega-3-fish-oil-for-brain-health>:



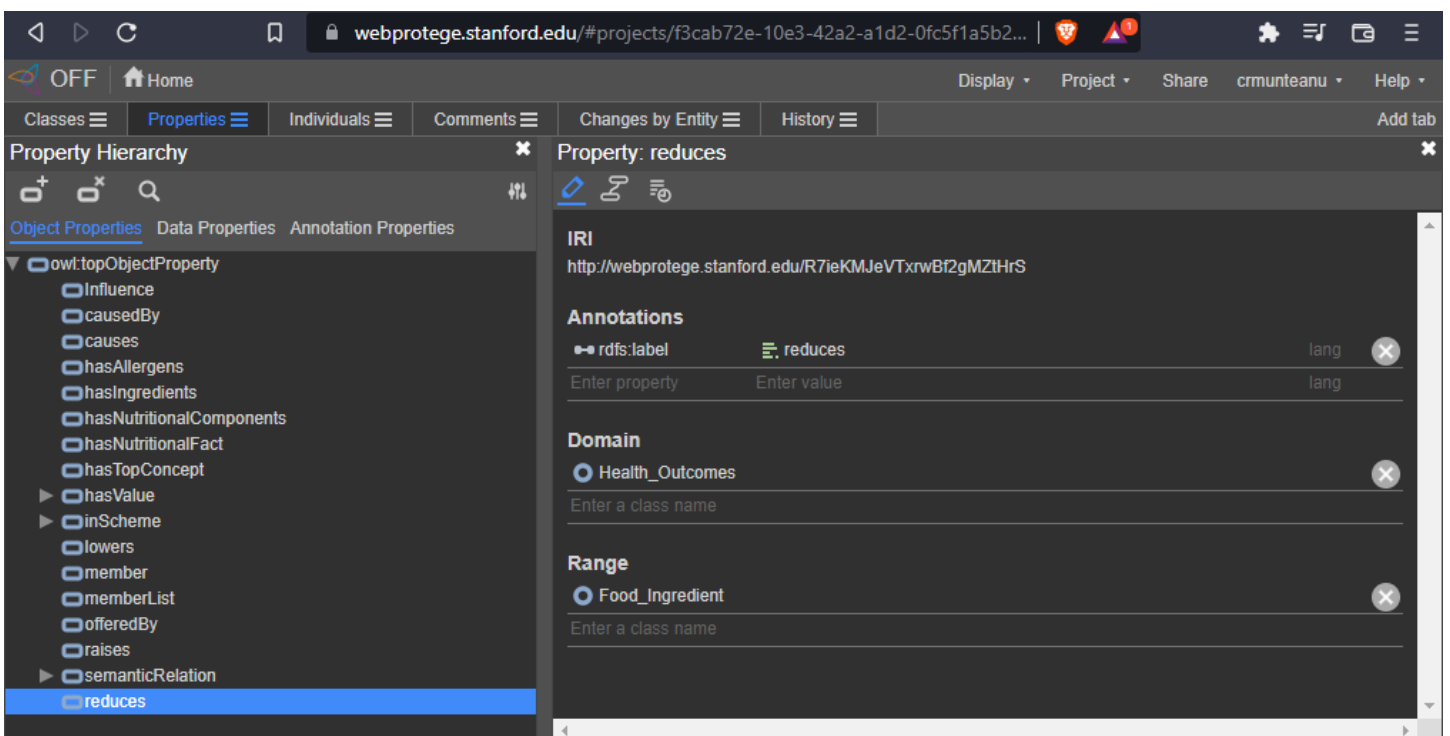
Remember that you can create several classes in one step by using ENTER in the Create step (you define a list of classes in one step). We can check the Entity Graph for this new class:



Let's add *reduces* as a **new Object Property** in order to define that *Omega3\_ingredient reduces Heart\_Disease*. Go to **Properties, Object Properties** and click on Create (+). Set the name of the new property as *reduces*:



You should use the **Range** as *Food\_Ingredient* and the **Domain** as *Health\_Outcomes* because we shall use *Omega3\_ingredient* to link with a *Health\_Outcomes* such as *Heart\_Disease*:



Thus, we have a new class as *Omega3\_ingredient* and a new object/class property as *reduces*. Let's add a new relationship for *Omega3\_ingredient* with the *Heart\_Disease*. Select *Omega3\_ingredient* in Class Hierarchy and add a new relationship as *reduces* with *Heart\_Disease* in **Relationships**:

The screenshot shows the Protege web interface. On the left, the 'Class Hierarchy' panel lists various food ingredients, with 'Omega3\_ingredient' selected. On the right, the 'Class: Omega3\_ingredient' panel displays its IRI, annotations, parents, and relationships.

**Class: Omega3\_ingredient**

**IRI**  
http://webprotege.stanford.edu/RBYOsi44no2rJlKpQlUvg1

**Annotations**

- rdfs:label: Omega3\_ingredient (lang)
- rdfs:comment: Omega3 is a specific lipid very important for the brain (lang)
- rdfs:seeAlso: https://www.healthline.com/nutrition/omega-3-fish-oil-for-brain-health (lang)

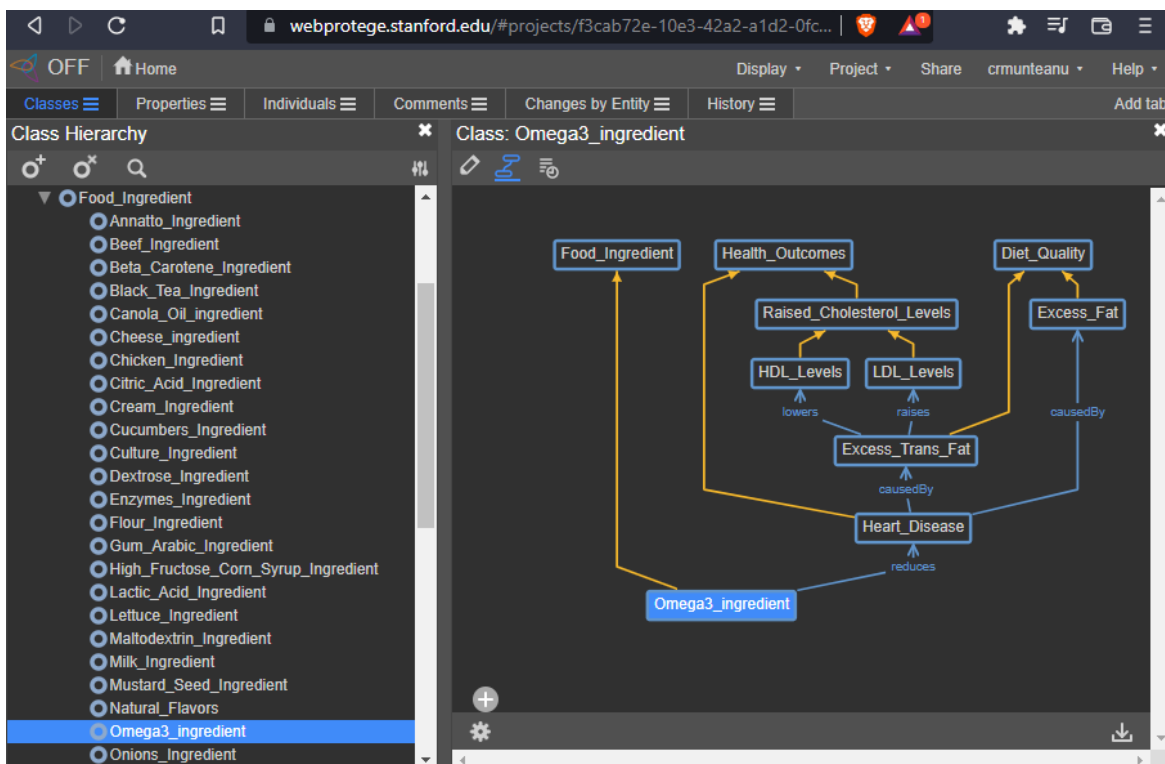
**Parents**

- Food\_Ingredient

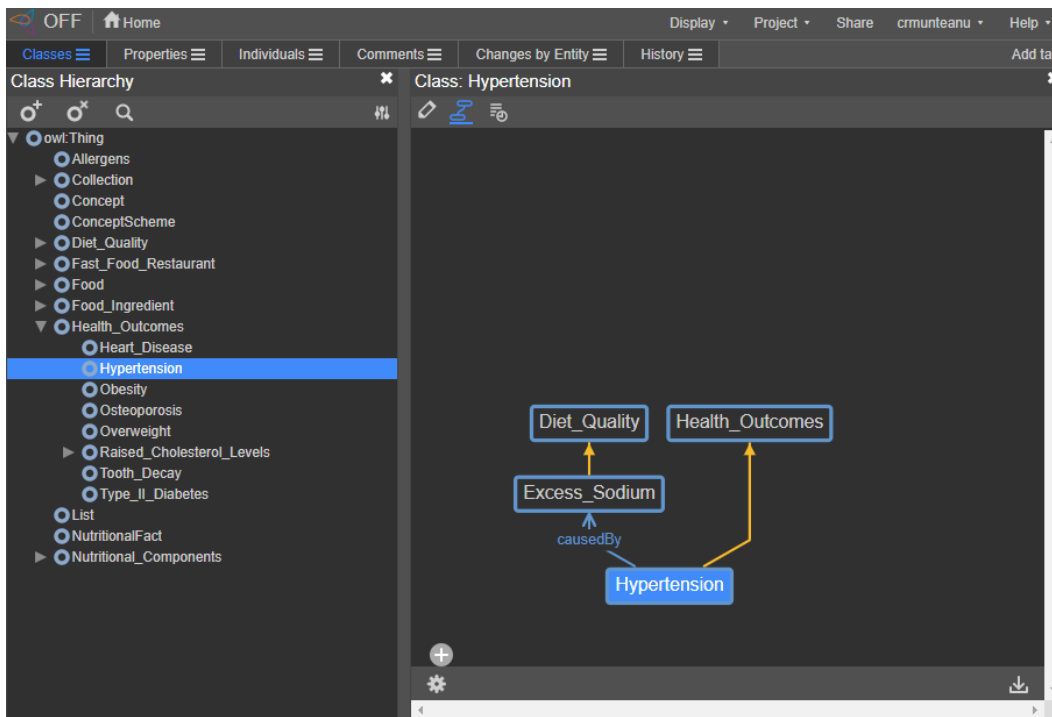
**Relationships**

- reduces: Heart\_Disease

Check the new entity graph for *Omega3\_ingredient*:



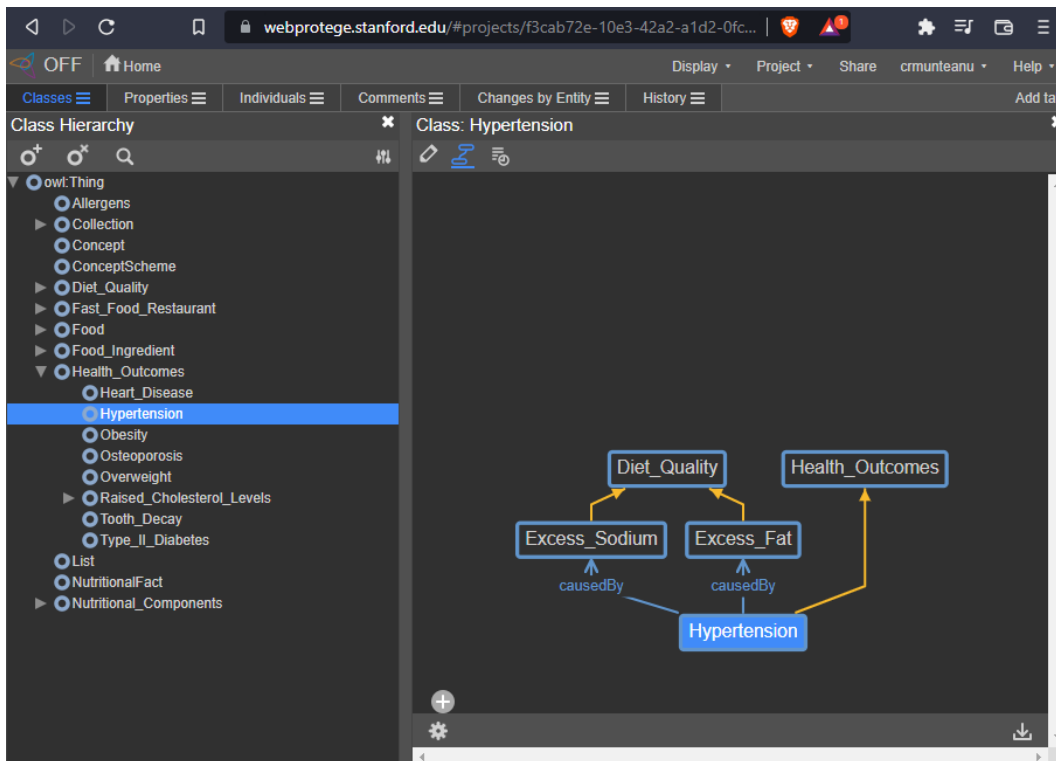
If you want to add a new relationship between classes by using an existing object property, you should try to define that *Hypertension* is *causedBy* *Excess\_Fat*. Before to add the new relationship, this is the Entity Graph:



Just select *Hypertension* from *Health\_Outcomes* and add a new relationship by choosing *causedBy* and *Excess\_Fat*:

This screenshot shows the 'Class: Hypertension' editor with the 'Relationships' section expanded. It shows two instances of the 'causedBy' relationship: one with 'Excess\_Sodium' and another with 'Excess\_Fat'. The 'Excess\_Fat' relationship is highlighted. The 'Annotations' and 'Parents' sections are also visible, showing the IRI and the parent class 'Health\_Outcomes' respectively.

The Entity Graph will show the new relationship:



## Adding data property

Let's add new units for vitamin D such as UI. First, you should add a new **Data Property** by selecting Amount and pressing Create (+). Use the name *Amount\_ui*, the **Domain** as *Nutritional\_Components* and the **Range** as positive integer:

The screenshot shows the Protege interface with the 'Properties' tab selected. The 'Property Hierarchy' panel on the left shows a tree structure with 'Amount\_ui' selected under 'Amount'. The main panel displays the details for the 'Property: Amount\_ui'. The 'IRI' is 'http://webprotege.stanford.edu/RV51K2iHswfzaawdbkDxm8'. The 'Annotations' section shows 'rdfs:label' with the value 'Amount\_ui'. The 'Characteristics' section shows 'Functional' checked. The 'Domain' is 'Nutritional\_Components'. The 'Range' is 'xsd:positiveInteger'.



Now you are able to add this data property to the class *Vitamin\_D* from *Nutritional\_Components*. Change to the tab **Individuals** and Create (+) an instance with the name “40”, set the class name as *Vitamin\_D* in the **Types**, add a data property as *Amount\_ui* and set its value to 4000:

The screenshot shows the Protege web interface at [webprotege.stanford.edu/#projects...](http://webprotege.stanford.edu/#projects...). The interface is divided into several panels:

- Top Panel:** Contains navigation tabs: OFF, Home, Display, Project, Share, crmunteanu, and Help.
- Left Panel:** Contains a sidebar with tabs: Classes, Properties, Individuals (selected), Comments, Changes by Entity, History, and Add tab.
- Individuals by Class Panel:** Shows a list of individuals under the class *owl:Thing*. The individual *My vitamin D* is selected.
- Individual Panel:** Shows the details for the selected individual *My vitamin D*. It includes:
  - IRI:** <http://webprotege.stanford.edu/RBJtZmllbwaLjoHRyqn0PC3>
  - Annotations:** A table with columns for property, value, and language. The first row shows *rdfs:label* with value *My vitamin D* and language *lang*.
  - Types:** A section where the class *Vitamin\_D* is assigned to the individual.
  - Relationships:** A table with columns for property, value, and language. The first row shows *Amount\_ui* with value *4000* and language *lang*.
  - Same As:** A section for defining same-as relationships.

TO DO: Modification of OFFF ontology

Please check the Tasks file.