



**Anglia Ruskin
University**

**Faculty of Science and
Engineering**

SEMANTIC DATA TECHNOLOGIES

MOD004979

Project : Engineering Materials Ontology

Hasan SHEN

2187317

Table of Contents

1. Introduction.....	4
2. Concept & Aim.....	4
3. Design	5
3.1 Semantic Data Technology	5
3.2 Model Implementation.....	5
3.3 Individuals and Data Pre-Processing	12
4. Implementation	15
4.1 Technology Stack.....	15
4.2 Dataset Connection	15
4.3 User Interface.....	17
5. Evaluation	21
5.1 Query to Select all the Materials.....	22
5.2 Query to Select all the Material Properties Related to All Materials.....	22
5.3 Query to FILTER The Material Property Yield Strength	23
5.4 Query to Find The Equivalent Material of a Material.....	24
5.5 Query to Select the Values Bigger Than The Average	24
5.6 Query to Get The Chemical Composition of A Material.....	25
6. Literature Review.....	26
7. Conclusion	28
8. References.....	29

Table of Figures

Figure 1 Class Hierarchy	Figure 2 Ontology Metrics	6
Figure 3 Application Areas Class		7
Figure 4 Element & Element Couple Classes		8
Figure 5 Eng Material Class		8
Figure 6 Material Property Class & The rest of the classes under Thing		9
Figure 7 Data Properties & Object Properties		10
Figure 8 Data Property - has name		10
Figure 9 Object property – belongs to		11
Figure 10 Object Properties is Equivalent Material Of & has unit		11
Figure 11 Onto Graph		12
Figure 12 Web Scraping		12
Figure 13 Excel sheet data manipulation		13
Figure 14 Transformation Rule Example		13
Figure 15 shows usage of Element Couple	Figure 16 Shows the usage of Carbon Element	14
Figure 17 shows usage of Eng Material individual		14
Figure 18 App Folders		15
Figure 19 Axios Get function usage		16
Figure 20 Nested IF-statement for Filter Statement		17
Figure 21 Shows the one of the Auto Complete sections in the UI		17
Figure 22 shows how the Tables generated the results		18
Figure 23 General Search Page		18
Figure 24 Complete Material page		19
Figure 25 Material Properties Query		20
Figure 26 Application Areas Query		20
Figure 27 Producer Information Query		21
Figure 28 Chemical Composition Query		21
Figure 29 Query to select all the material names		22
Figure 30 Query to select all the material properties related to all the materials in the ontology		22
Figure 31 Query to Filter the Yield Strength Value of all materials		23
Figure 32 Query to find the equivalent material		24
Figure 33 Query to Select the values bigger than the average		24
Figure 34 Query in Figure 33 continued		25
Figure 35 Query to get the chemical composition of a material		25
Figure 36 The results and the rest of the query shown in Figure 35		26

1. Introduction

In this report, Engineering Materials Ontology will be discussed. Engineering Materials Ontology is an ontology based data structure where we use triples to identify and relate our data instead of tabular formatted data.

The structure of this report is as follows:

Firstly, concept and aim of the semantic web application will be discussed in details. Then, the design procedure of the ontology will be explained with the domain details, and classes, data properties, object properties, individuals and the usages of the data and object properties will be discussed. Following that, we will cover how the ontology is implemented to a web application, answering questions such as how the data connections are made, how the queries generated and instructions about the user interface. And after that, we have evaluation section where we will cover 6 of the example queries used in the Fuseki local server which will help us to discover the ontologies limitations and flexibilities. Finally, we will discuss the similar ontologies from the articles in the same domain and make comparisons between them and the ontology which is covered in this report.

As a final touch, we will cover the conclusion part where we criticize our proposed ontology and discuss in which ways it could improve and finally, we will make a conclusion and cover what we have learnt to that point.

2. Concept & Aim

Following the recent development of technology, engineering materials are increasing admittedly in quantity and variety. This creates a problem to material selection processes which needs to be handled in extreme care. In the process of material selection, engineers takes into account a huge variety of parameters such as thermal properties, mechanical properties, electrical properties and some other parameters which are not addressed in this ontology.

With the growing technology, material analysing and measuring their properties evolved into a stage that now in a material datasheet one can find enormous amount of data in variety and quantity. This is where Engineering Materials Ontology can be used instead of the traditional databases, so that handling the data and reaching the result can be more convenient than the current approaches.

The Engineering Materials Ontology is a semantic web application in the domain of engineering materials and designed for to filter among material properties and suppliers for mostly engineers to use this application for selection of materials appropriate for their design requirements in their projects. Hence *The Engineering Materials Ontology* can be used in the future as a framework for an AI based material selection algorithm. Since the material selection objective is beyond the requirements of the assignment *The Engineering Materials Ontology* only focuses on materials domain but designed keeping in mind that it can be re-usable in a bigger ontology for classifying materials.

The Engineering Materials Ontology is designed to be used by engineers, and to be maintained and updated by a society which includes engineers, material providers and Domain experts who are mainly material scientists. This ontology can help finding a type of material, filtering a property value, finding the application areas of the materials, finding the suppliers and locations for the suppliers, and finally the ontology can help finding the equivalent material of a material if there is any. The main purpose of the ontology is to select a material for any engineering project.

The knowledge based approach for this purpose is used to make a framework that can be expanded in the future so that the application can handle the needs of the rapidly changing technology and also to make the ontology portable such as it can be included in a bigger ontology which covers a specific application in details.

3. Design

3.1 Semantic Data Technology

Before heading to the the subject of this paper. We must clarify the one of main aims of this paper which is to discover the benefits and capabilities of Semantic Data Technologies. Semantic data technology aims to create a linked data so that it is readable by the machine. In basics, semantic data technologies differ from the conventional databases in a way that all the data is linked to each other as subject predicate and object. Every relation and the two end points of the relations is also a data. When a search is made, in the semantic web the query will follow a certain or a few possible paths to the endpoint which is result. And this result is much more meaningful than the normal database search results since every data has a Unique Resource Identifier and all of these URI's are under the well categorized class hierarchy. There is not much chance to get wrong results, since everything is vocabularied, so that there is only the meaningful and precise results at the endpoints.

To further clarify the subject predicate object we can give an example from our *Engineering Materials Ontology*.

DILLIDUR 500 “produced by” Producer

In the above example subject is “DILLIDUR 500” which is an individual in our ontology under the *Tool Steel* class and *produced by* is our predicate, in other words linking property and in specifically it is object property. Producer is our object in here which is the range of the predicate, the data which is linked to the DILLIDUR 500 individual.

SPARQL language comes in to action here and it helps us to make searches in this web of linked data using the Apache Jena Fuseki server and connecting to our ontology by using the namespace which is <urn:absolute:Engineering_Materials_OWL_V1.2#> in our ontology.

3.2 Model Implementation

The domain is based on engineering materials which is a field that is growing constantly with respect to technological developments. For instance, in the recent half of the decade additive manufacturing technologies popped up although the technology was already in the field, the popularity of it recently made it to advance faster than before. This situation creates a big gap in the handling of materials data, although there are good material databases online such as “*TotalMateria.com*”, “*matmatch.com*” and few others specialized in some material types.

In this paper, proposed ontology is based on *Eng Material* class which is the superclass of categorized materials shown in *figure 1*. And every other class can be related to *Eng Material* with object properties and other information can be related to it by data properties which we will cover in the following sections.

This ontology will mainly be used for searching a material to be used in a specific engineering project with some specific property values. Also users will be able to find data on the producers, elements and in which application areas the material is involved.

Figure 2 shows the overview of the *Engineering Materials Ontology*. In the ontology, we have a total of 11973 axioms, 9779 of them are logical axioms and 2194 of them are declaration axioms. Total class count is 339 with 15 object properties and 12 data properties. And in total 1830 individuals uploaded to the ontology.

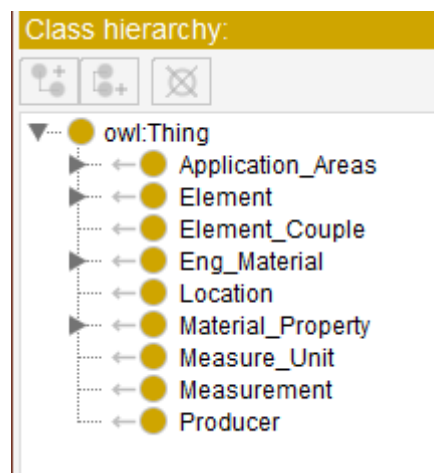


Figure 1 Class Hierarchy

Ontology metrics:	
Metrics	
Axiom	11973
Logical axiom count	9779
Declaration axioms count	2194
Class count	339
Object property count	15
Data property count	12
Individual count	1830
Annotation Property count	0
Class axioms	
SubClassOf	330
EquivalentClasses	0
DisjointClasses	55
GCI count	0
Hidden GCI Count	0
Object property axioms	
SubObjectPropertyOf	14
EquivalentObjectProperties	0
InverseObjectProperties	2
DisjointObjectProperties	0
FunctionalObjectProperty	2
InverseFunctionalObjectProperty	0
TransitiveObjectProperty	1
SymmetricObjectProperty	1
AsymmetricObjectProperty	2
ReflexiveObjectProperty	0
IrreflexiveObjectProperty	1
ObjectPropertyDomain	14
ObjectPropertyRange	14
SubPropertyChainOf	0
Data property axioms	
SubDataPropertyOf	10
EquivalentDataProperties	0
DisjointDataProperties	0
FunctionalDataProperty	0
DataPropertyDomain	18
DataPropertyRange	11
Individual axioms	
ClassAssertion	1724
ObjectPropertyAssertion	4546
DataPropertyAssertion	3034
NegativeObjectPropertyAssertion	0
NegativeDataPropertyAssertion	0
SameIndividual	0
DifferentIndividuals	0
Annotation axioms	
AnnotationAssertion	0
AnnotationPropertyDomain	0
AnnotationPropertyRangeOf	0

Figure 2 Ontology Metrics

As we progress in the class hierarchy, starting from the top to bottom, we have *Application Areas* class which has a total of 164 subclasses which mainly categorizes where the engineering materials are *involved* in as shown in *figure 3*. Every subclass has an individual which is named as *Class name IRI*. The purpose of this is to creating a web of linked data between *Application Areas* and *Eng Material* so that we can link them instead of giving this information directly to the *Eng Material* class as a data property which in the end will help us create more relations and resulting a better ontology design.

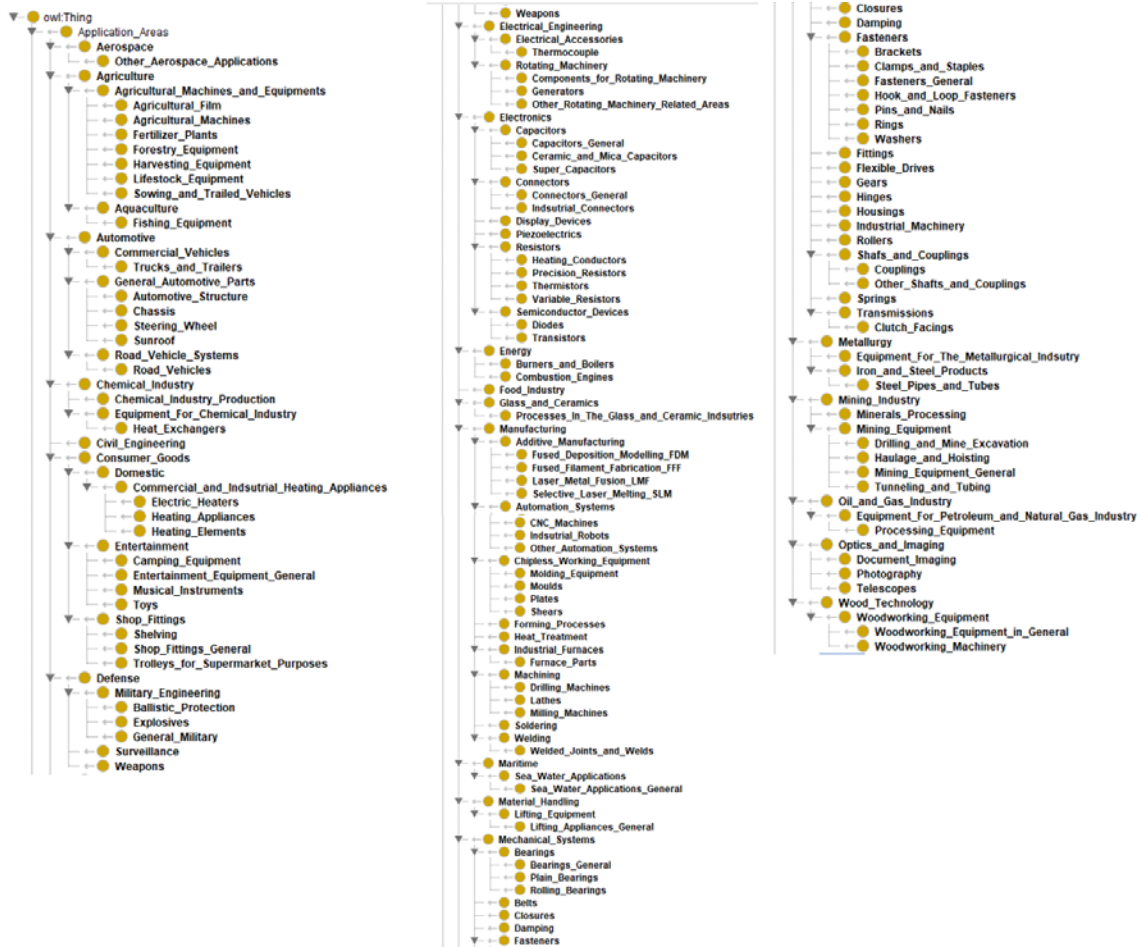


Figure 3 Application Areas Class

Following class hierarchy, we have *Element* class which has 11 subclasses categorizing periodic table elements. Individuals of subclasses of *Element* will be used in the *Element Couple* class as *Element Couple* -> has Element -> *Element*. *Element Couple* class does not have a name as data property, and the reason for this we will never be reaching to the *Element Couple* by a direct SPARQL query. We will be reaching through the material individual, then *Chemical Composition* class, and then finally *Element Couple*. One other reason why we need an *Element Couple* class is that we need to attach more than one *Element* individuals to the *Chemical Composition* class which is under *Material Property* class, and creating disjoint data properties related to every different *Element* individual is not possible and convenient, and we cannot create any links in between different *Element* individuals under the *Chemical Composition* class. *Element Couple* and *Element* class is shown in

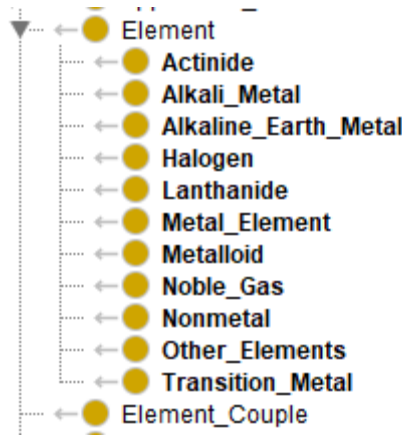


Figure 4 Element & Element Couple Classes

Eng Material class is the main domain of focus in this ontology and categorizes the material types. The class has 92 subclasses under it and every individual of material is created under their own material type class. This enables us to FILTER according to their category under the hierarchy. The class and its subclasses can be seen in Figure 5.

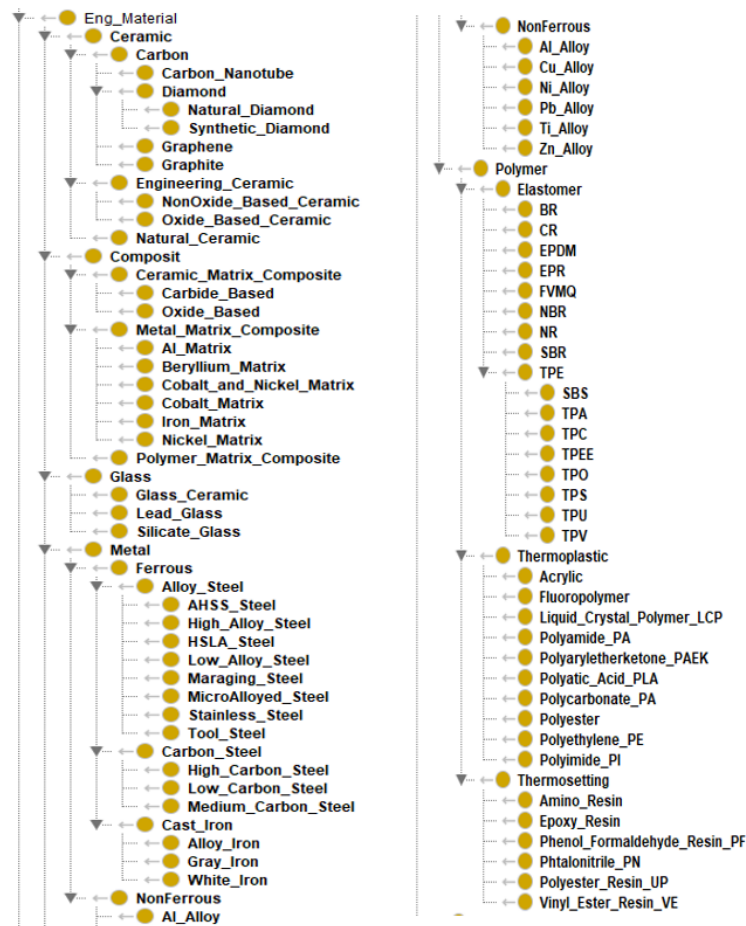


Figure 5 Eng Material Class

Under it we have *Location* class and it is basically only for the *Producer* class. Following that, we have *Material Property* class which is also the second main focus of domain in this ontology since

we make most of our SPARQL queries according to materials properties. As we mentioned above, we apply the same structure as the *Chemical Composition* class for our material properties. While checking the materials data's uploaded to this ontology, realized that some material properties has more than one value which we can call it *Measurement* which is a class under *Thing*. So we link the material properties with the measurements except the *Chemical Composition* since it requires another definition different from the rest of the material properties. *Material Property* class hierarchy can be seen in Figure 6.



Figure 6 Material Property Class & The rest of the classes under Thing

Moving in the ontology design, we created our classes and also explained a bit about the data properties, object properties and the relations between them. Now let's look at the details of data properties and object properties.

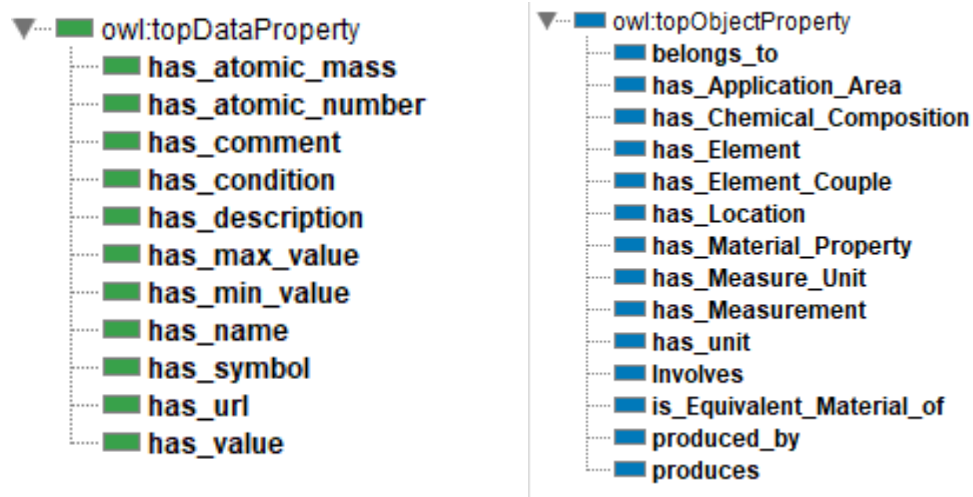


Figure 7 Data Properties & Object Properties

Starting from data properties, we have set all of our data property ranges to xsd:String since this is more convenient for uploading individuals. All the data properties and object properties can be seen from Figure 7. Max cardinality 1 added to the following data properties: *has atomic mass*, *has atomic number*, *has symbol*, *has max value*, *has min value*, *has value*, *has name*. An example can be seen from the Figure 8.

Figure 8 Data Property - has name

For the object properties, there are 2 inverse object properties which are *produces*, *produced by* and *has Application Area*, *Involves*; there are 2 functional object properties which are *has unit* and *has Element*; there are 2 Asymmetric object properties which are *produces* and *produced by*; there is 1 Transitive object property which is ; there is 1 Symmetric and Irreflexive object property which is *is equivalent material of*. Some of the examples can be seen from the Figure 9 and Figure 10.

Inverse object properties are assigned as such since they are relating to same individuals in an oppositely way. Functional object properties are the properties which can only have one relation to the individual that is related which in our case *measurement* class can only have *unit* linked with *has unit* object property.

Characterist	Description: belongs_to
<input type="checkbox"/> Functional <input type="checkbox"/> Inverse functional <input checked="" type="checkbox"/> Transitive <input type="checkbox"/> Symmetric <input type="checkbox"/> Asymmetric <input type="checkbox"/> Reflexive <input type="checkbox"/> Irreflexive	Equivalent To + SubProperty Of + owl:topObjectProperty Inverse Of + Domains (intersection) + Chemical_Composition or Element or Element_Couple or Material_Property or Measure_Unit or Measurement Ranges (intersection) + Chemical_Composition or Element_Couple or Eng_Material or Material_Property or Measurement Disjoint With +

Figure 9 Object property – belongs to

Characterist	Description: is_Equivalent_Material_of	Characterist	Description: has_unit
<input type="checkbox"/> Functional <input type="checkbox"/> Inverse functional <input type="checkbox"/> Transitive <input checked="" type="checkbox"/> Symmetric <input type="checkbox"/> Asymmetric <input type="checkbox"/> Reflexive <input checked="" type="checkbox"/> Irreflexive	Equivalent To + SubProperty Of + owl:topObjectProperty Inverse Of + is_Equivalent_Material_of Domains (intersection) + Eng_Material Ranges (intersection) + Eng_Material Disjoint With + SuperProperty Of (Chain) +	<input checked="" type="checkbox"/> Functional <input type="checkbox"/> Inverse functional <input type="checkbox"/> Transitive <input type="checkbox"/> Symmetric <input type="checkbox"/> Asymmetric <input type="checkbox"/> Reflexive <input type="checkbox"/> Irreflexive	Equivalent To + SubProperty Of + owl:topObjectProperty Inverse Of + Domains (intersection) + Element_Couple or Measurement Ranges (intersection) + Measure_Unit has_unit max 1 Measure_Unit Disjoint With + SuperProperty Of (Chain) +

Figure 10 Object Properties is Equivalent Material Of & has unit

From the below Figure 11 Onto Graph we can observe the structure of the ontology.

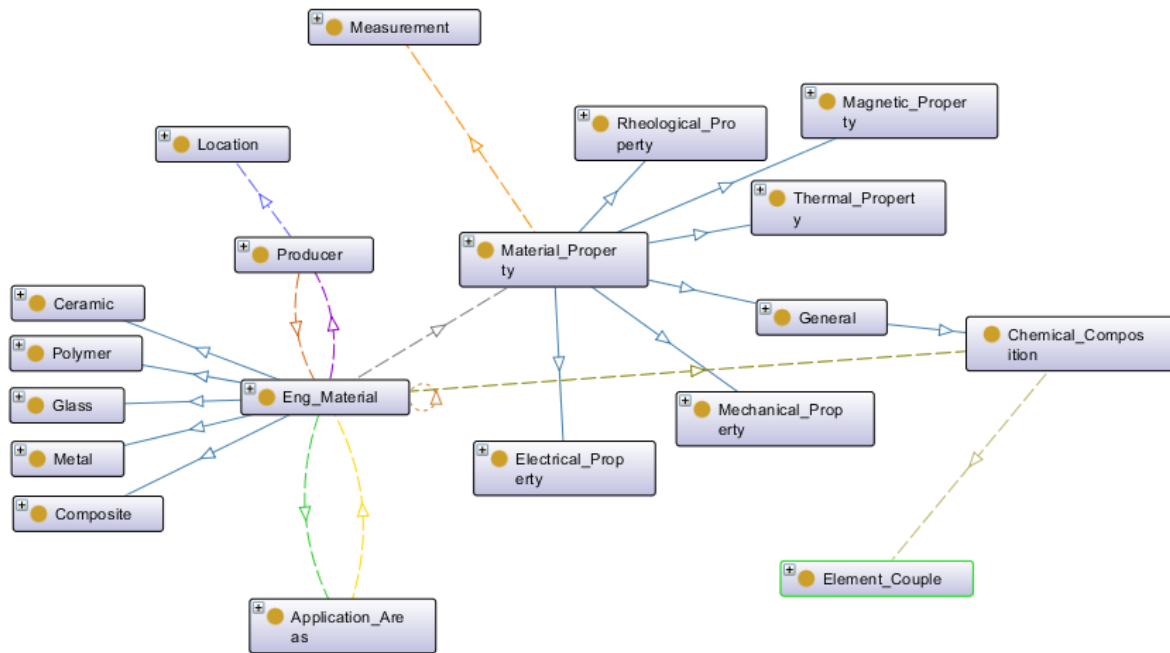


Figure 11 Onto Graph

3.3 Individuals and Data Pre-Processing

Now we have created our ontology and it has come to the point where we upload our individuals. For individuals, <matmatch.com> website is used mostly and at first web scraping applied with the python's Selenium library, but it was very time consuming since the data is so variant such as all the materials do not have the all material properties and there is too much variance in the data. Instead of web scraping, excel sheets taken from the website is used and with python programming data manipulation and extraction is applied on the excel data sheets and appended to a new excel file in order to make the uploading convenient. *Figure 12* shows the Selenium library usage in the project. *Figure 13* shows the excel sheet manipulation and appending to another excel sheet for the convenience of uploading. And *Figure 14* shows an example of the Transformation Rules that used while uploading individuals.

```

from selenium import webdriver
import pandas as pd
import numpy as np
import openpyxl
import math

driver = webdriver.Chrome("chromedriver.exe")

driver.get("https://matmatch.com/materials/dill022-dillidur-500")
driver.maximize_window()

name =
driver.find_element_by_xpath("/html/body/div[1]/div/div[4]/div/div[1]/div[1]/div[2]/h1")
Description_html =
driver.find_element_by_xpath("/html/body/div[1]/div/div[4]/div/div[1]/div[1]/div[2]/div[1]/div/div[1]/p[1]")
Producer_html =
driver.find_element_by_xpath("/html/body/div[1]/div/div[4]/div/div[1]/div[2]/div[1]/div/div/div[1]/div[1]/div[2]/div/div/p")

Material_name = name.text
Description = Description_html.text
Producer = Producer_html.text
producer_IRI = Producer

properties =
driver.find_elements_by_xpath("*/html/body/div[1]/div/div[4]/div/div[3]/div[2]/div/table/tbody/tr/td[1]")

property_name_Array = []
i = 0
measurements = []
for property in properties:
    property_name = str(property.text)
    if not property_name:
        print("property not found")
    property_name_Array.append(property_name_Array[i-1])
    else:
        property_name_Array.append(property_name)

#print(property_name_Array)

property_name_Array_URI = []
rank = 0
for i in range(len(property_name_Array)):
    a = ""
    b = ""
    for j in range(len(property_name_Array[i].split(" " ))):
        b = property_name_Array[i].replace(" ", "")
        if b == (len(property_name_Array[i].split(" " )) - 1):
            a += b.split(" ")[j] + "_dillidur500"
        else:
            a += b.split(" ")[j] + "_"

    property_name_Array_URI.append(a)
    measurements.append("measurements_"+str(i))

print(property_name_Array_URI)
print(measurements)

```

Figure 12 Web Scraping

For further details, we can look into individuals and their usages in the ontology. In the *Figure 15* and *Figure 16* we can observe the general structure as Carbo-(material name) *has Element* Carbon IRI. Following that Carbon *Element* *has name* as Carbon and *has symbol* as C and the other details related to it linked with data properties. For the *Element Couple*, it *has Element* as Carbon IRI, *has value* as 0.02, and *has unit* as unit percent and *belongs to* is an object property which relates it with *Chemical Composition*.



Figure 15 shows usage of Element Couple

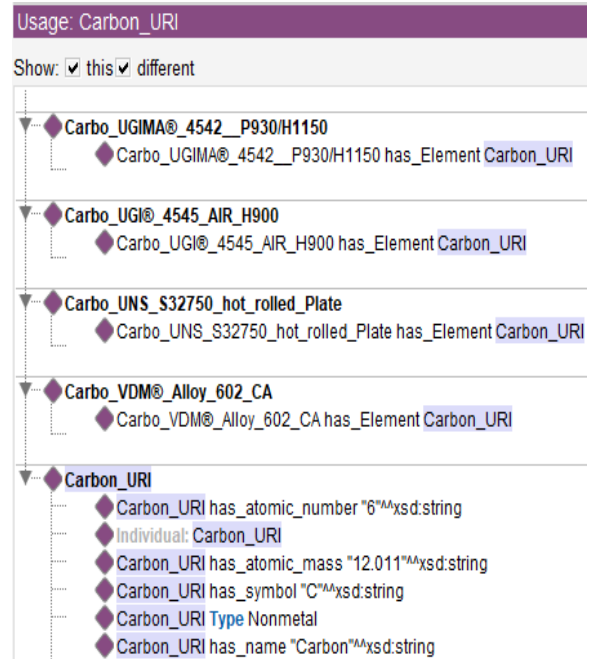


Figure 16 Shows the usage of Carbon Element

When we look at the focus domain in our *Engineering Material Ontology* in *Figure 17*, we can observe the ontology structure.

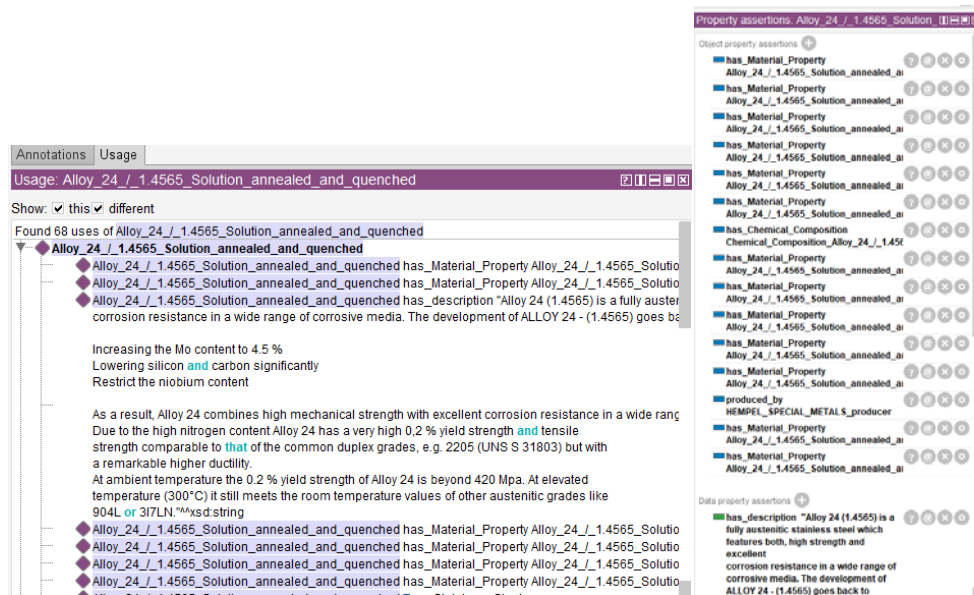


Figure 17 shows usage of Eng Material individual

4. Implementation

In this section, implementation approaches for our ontology to the web application will be discussed. First, technologies are used in the user interface will be covered. Then, dataset connection approaches will be discussed. Finally, the finished user interface and usage will be debated.

4.1 Technology Stack

Protégé is used to create ontology and SPARQL queries are used to get the filtered data results via apache jena fuseki server. The web application is created with React, JavaScript and material UI. React is a free JavaScript library for building user interfaces(React) and it makes creating a html file easier since while writing the code you don't deal too much with html coding the webpage. Material UI enabled the nice and neat look in the webpage with built in and ready to use html components library and also in their free subscription plan they provide everything except the copyrights, since in this project we are working only for academic purposes, material UI is one of the best options for a person who has not dealt with UI side.

In the project folder we have src file where we have our hosting App.js file where the working code hosted on and project webpages are located under the Components folder.

Axios is used to get request from Apache Jena Fuseki Server. Axios is an HTTP client library which enables us to send request to a given endpoint which in our case is Apache jena Fuseki. In these files *Homepage.js* is empty since the users make their searches in the main page and Complete Material page.

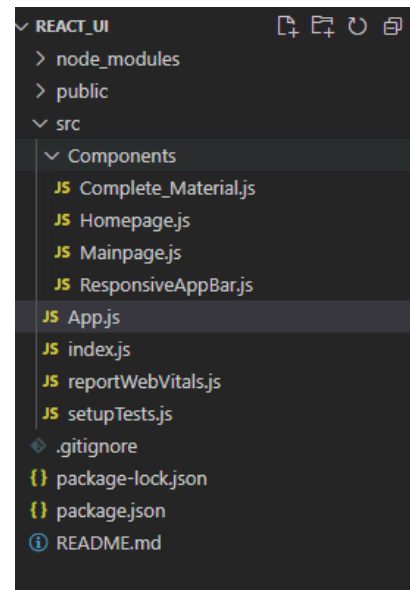


Figure 18 App Folders

4.2 Dataset Connection

In this section how *Engineering Materials Search Application* connects the Apache Jena Fuseki server will be explained. Axios library is used for this purpose and the implementation is shown in Figure 19.

```
hasan@MSI MINGW64 ~/OneDrive/Masaüstü/Semantic Project/react_ui
$ npm install axios
```

First we run the code in the above code snippet in our project file location in git bash command prompt to load the library. Importing the Axios library into our code will make the Axios available for us to use.

After creating user interface we got our inputs from the user and implemented a code to manipulate the user input to get a valid query from the Apache Fuseki Server. “Axios. Get()” is used to send query request to the server. In Figure 19, we can see the query code with the mentioned function.

In the Axios function, the endpoint set as `http://localhost:3030/#/dataset/Engineering_Materials_OWL_V1.2.owl/query` which is Fuseki servers local address. The query is set in the function as `{params: {query: 'SPARQL Query'}}`. And the response comes as an JSON variable and it is then assigned to `responseData` `useState` variable created as shown in the below code snippet.

```
const [responseData, setResponseData] = useState([])
```

The following code snippet shows how the response data is assigned to `responseData` variable.

```
.then(response => {
    setResponseData(response.data.results.bindings)
})
```

```
axios.get('http://localhost:3030/Engineering_Materials_OWL_V1.2.owl/',
    {params: {query: `
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX material: <urn:absolute:Engineering_Materials_OWL_V1.2#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT  ?name (CONCAT(?min_value, ?unitname) as ?value_unit)
(GROUP_CONCAT(DISTINCT ?app_name; SEPARATOR=", ") AS ?Application_Areas)
?producer (GROUP_CONCAT(DISTINCT ?location; SEPARATOR=", ") AS
?Locations) WHERE {
    ?x a ?y.
    ?y rdfs:subClassOf* material:Eng_Material.
    ?x material:has_name ?name .
    ?x material:produced_by ?prod .
    ?prod material:has_name ?producer .
    ?prod material:has_Location ?loc .
    ?loc material:has_name ?location .
    ?x material:has_Application_Area ?app .
    ?app material:has_name ?app_name.
    ?x material:has_Material_Property ?prop .
    ?prop rdf:type ?type .
    ?prop material:has_name ?propname .
    ?prop a material:${property} .
    ?prop material:has_Measurement ?mes .
    ?mes material:has_min_value ?min_value .
    ?mes material:has_unit ?unit .
    ?unit material:has_name ?unitname .
    ${filter_statement}

}
Group by ?name ?min_value ?unitname ?producer
Order by ?name`}}).then(response => {
    console.log(response.data.results.bindings)
    setResponseData(response.data.results.bindings)
})
```

Figure 19 Axios Get function usage

For getting our query a nested if function is used for the filter statement. The if functions are used under the handle submit function which is triggered by the search button. According to user input we can generate our FILTER section in the query and select the material property in the query. The material property selection is a must otherwise we will get too much data to view and one of the differences between ontology based search and conventional search is that we get more related data when we do search in knowledge based environments than the conventional databases. For a typical user of this application the material property is something they search for in the first place.

We can see in *Figure 20* our nested if code where we created our FILTER part of our query. And the user do not need to enter any filters and thus the user can view the all the results available.

```
const handleSubmit = () => {
  var filter_statement = ''
  if((comp != null) || (comparisonInput != null) || (producer_name !=
null) || (app_area != null) || (material_type != null)){
    filter_statement += 'FILTER('
    if((comp != null) && (comparisonInput != null)){
      filter_statement += `(xsd:float(?min_value) ${comp}
${comparisonInput})`
    }
    if((producer_name != null) || (app_area != null)){
      filter_statement += '&& '
    }
  }
  if((producer_name != null)){
    filter_statement += `(?producer = '${producer_name}')`
    if((app_area != null)){
      filter_statement += '&& '
    }
  }
  if((app_area != null)){
    filter_statement += `(?Application_Area = '${app_area}'))`
  }
  if((material_type != null)){
    filter_statement += `(STRENDIS(str(?y),'${material_type}'))`
  }
  filter_statement += ')'
}
```

Figure 20 Nested IF-statement for Filter Statement

4.3 User Interface

In this section how the user interface is created will be discussed and queries used in the web application will be presented. Material UI is used vastly for the user interface for creating containers, grid items, auto complete text fields, buttons and for tables to print the results. Example codes for some of the items can be seen in *Figure 21* and *Figure 22*.

The components in the user interface are used from material UI library to make it easier to design.

```
<Grid item xs={3}>
  <Autocomplete disablePortal id="combo-box-demo"
    size="small" options={material_type_all}
    onChange={handleChangeMat_type}
    renderInput={({params}) => <TextField {...params} label="Material Type" />}
  />
</Grid>
```

Figure 21 Shows the one of the Auto Complete sections in the UI

```

<Grid item xs={12}>
  <TableContainer component={Paper}>
    <Table>
      <TableHead>
        <TableRow>
          <TableCell>Material Name</TableCell>
          <TableCell align="right">Property</TableCell>
          <TableCell align="right">Value&nbsp;(Unit)</TableCell>
          <TableCell align="right">Application Area</TableCell>
          <TableCell align="right">Producer</TableCell>
          <TableCell align="right">Location</TableCell>
        </TableRow>
      </TableHead>
      <TableBody>
        {responseData.map((responseDataInd, i) => (
          <TableRow
            key={i}
            sx={{ '&:last-child td, &:last-child th': { border: 0 } }}
          >
            <TableCell component="th" scope="row" >
              {responseDataInd.name.value}
            </TableCell>
            <TableCell align="right">{property}</TableCell>
            <TableCell align="right">{responseDataInd.value_unit.value}</TableCell>
            <TableCell align="right">{responseDataInd.Application_Areas.value}</TableCell>
            <TableCell align="right">{responseDataInd.producer.value}</TableCell>
            <TableCell align="right">{responseDataInd.Locations.value}</TableCell>
          </TableRow>
        ))}
      </TableBody>
    </Table>
  </TableContainer>
</Grid>

```

Figure 22 shows how the Tables generated the results

The first search page is shown in Figure 23 and user only needs to select the material property to get results and may select other options to FILTER the results. The user can FILTER materials according to their material type, material property, producer and application areas. The application is in its baby steps since some errors occurred trying to enhance the user experience and the simplest version is chosen. The user can view the materials according to their material properties then generate an idea on their mind about which material they want to view in detail. The query used for the first page shown in Figure 19. And also the producer names, material types, material properties were taken from the Axios get() function.

Material Name	Property	Value (Unit)	Application Area	Producer	Location
AISI 1.4307	Yield_Strength	175MPa	Aerospace, Automotive, Maritime	AMICA Steels Limited	New Delhi
AISI 439	Yield_Strength	300MPa	Maritime, Chemical Industry, Consumer Goods, Mining Equipment, Mining Industry, Sea Water Applications, Tunneling and Tunnels	Deutsche Edelstahlwerke (DEW)	Germany
AISI 429	Yield_Strength	840MPa	Maritime, Chemical Industry, Consumer Goods, Mining Equipment, Mining Industry, Sea Water Applications, Tunneling and Tunnels	Deutsche Edelstahlwerke (DEW)	Germany
INI 1.4542	Yield_Strength	500MPa	Aerospace, Oil and Gas Industry	AMICA Steels Limited	New Delhi
Sandvik S1809H	Yield_Strength	100MPa	Consumer Goods, Agriculture, Fertilizer Plants, Steel Pipes and Tubes, Energy	ALUMINA (FORMERLY SANDVIK MATERIALS TECHNOLOGY)	Sweden
Sandvik S1809H Urea Grade	Yield_Strength	190MPa	Consumer Goods, Agriculture, Fertilizer Plants, Steel Pipes and Tubes, Energy	ALUMINA (FORMERLY SANDVIK MATERIALS TECHNOLOGY)	Sweden
Sandvik S1809H Urea Grade	Yield_Strength	225MPa	Consumer Goods, Agriculture, Fertilizer Plants, Steel Pipes and Tubes, Energy	ALUMINA (FORMERLY SANDVIK MATERIALS TECHNOLOGY)	Sweden
Sandvik S1809H Urea Grade	Yield_Strength	230MPa	Consumer Goods, Agriculture, Fertilizer Plants, Steel Pipes and Tubes, Energy	ALUMINA (FORMERLY SANDVIK MATERIALS TECHNOLOGY)	Sweden
Sandvik S1809H Urea Grade	Yield_Strength	230MPa	Automotive, Chemical Industry, Burners and Boilers, Combustion Engines, Electrical Engineering, Manufacturing, Steel Pipes and Tubes, Electrical Accessories, Energy Equipment for Chemical Industry, Equipment for the Metallurgical Industry, Furnace Parts, Gases and Ceramics, Heat Exchangers, Heat	ALUMINA (FORMERLY SANDVIK MATERIALS TECHNOLOGY)	Sweden

Figure 23 General Search Page

On the second page the user can view full material data such as all the material properties related to that material, chemical composition, application areas and producer information. The user only needs to choose the material name from the first page.

Engineering Materials Search Application
 MAINPAGE
COMPLETE_MATERIAL

Material Name

Material Data		
Property	Value	Unit
Coefficient of thermal expansion	1.65e-05	1/K
Coefficient of thermal expansion	1.7e-05	1/K
Density	8	g/cm³
Elastic modulus	200	GPa
Elastic modulus	194	GPa
Elongation	40	%
Elongation A2	35	%
Recycled Content	82.1	%
Specific heat capacity	485	J/kg K
Specific heat capacity	500	J/kg K
Tensile strength	490	MPa
Tensile strength	690	MPa
Thermal conductivity	14	W/m K
Thermal conductivity	15	W/m K
Yield strength Rp0.1	235	MPa
Yield strength Rp0.1	225	MPa
Yield strength Rp0.2	190	MPa
Yield strength Rp0.2	180	MPa

Material Chemical Composition	
Element	Value (Unit)
Carbon	0.02 %
Phosphorus	0.02 %
Sulfur	0.01 %

Application Areas	
Material Name	Application Areas
Sandvik 3R60™ Urea Grade	Consumer Goods, Agriculture, Fertilizer Plants, Steel Pipes and Tubes, Energy

Material Producer Data			
Producer Name	Description	URL	Location
ALLEIMA (FORMERLY SANDVIK MATERIALS TECHNOLOGY)	Manufacturer	https://www.alleima.com/en/	Sweden

Figure 24 Complete Material page

There are 4 SPARQL queries sent to Apache Jena Fuseki server with the Axios .get() function. First one is to get all the material properties related to the material, the second one is to get chemical composition of the material, the third one is to get application areas of the material and the last one to get the producer information. The queries are shown in *Figure 25*, *Figure 26*, *Figure 27*, *Figure 28*.

```

axios.get('http://localhost:3030/Engineering_Materials_OWL_V1.2.owl/',
  {params: {query: `
PREFIX ma: <http://www.w3.org/ns/ma-ont#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX material: <urn:absolute:Engineering_Materials_OWL_V1.2#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?prop_name ?measure ?min_value ?unitname WHERE

{
  ?x a ?y.
  ?y rdfs:subClassOf* material:Eng_Material.
  ?x material:has_name ?name
  FILTER(?name = "${material_name}")
  ?x material:has_Material_Property ?c .
  ?c material:has_Measurement ?measure .
  ?c material:has_name ?prop_name .
  ?measure material:has_min_value ?min_value .
  ?measure material:has_unit ?unit .
  ?unit material:has_name ?unitname .
}
`}}).then(response => {
  setResponseData(response.data.results.bindings)
})

```

Figure 25 Material Properties Query

```

axios.get('http://localhost:3030/Engineering_Materials_OWL_V1.2.owl/',
  {params: {query: `
PREFIX ma: <http://www.w3.org/ns/ma-ont#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX material: <urn:absolute:Engineering_Materials_OWL_V1.2#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?name (GROUP_CONCAT(DISTINCT ?app_name; SEPARATOR=", ") AS
?Application_Areas) WHERE {
  ?x a ?y.
  ?y rdfs:subClassOf* material:Eng_Material.
  ?x material:has_name ?name
  FILTER(?name = "${material_name}")
  ?x material:has_Application_Area ?c .
  ?c material:has_name ?app_name .
} GROUP BY ?name
`}}).then(response => {
  setResponseData_app(response.data.results.bindings)
})

```

Figure 26 Application Areas Query

```

axios.get('http://localhost:3030/Engineering_Materials_OWL_V1.2.owl/',
  {params: {query: `
PREFIX material: <urn:absolute:Engineering_Materials_OWL_V1.2#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?name ?descr ?url (GROUP_CONCAT( ?loc;
SEPARATOR=", ") AS ?Location) WHERE {
  ?prod a material:Producer .
  ?prod material:has_name ?name .
  ?prod material:has_description ?descr .
  ?prod material:has_Location ?lo .
  ?lo material:has_name ?loc .
  ?prod material:has_url ?url .
  ?prod material:produces ?mat .
  ?mat material:has_name ?mat_name
  FILTER(?mat_name = "${material_name}")
}
GROUP BY ?name ?descr ?url
`}}).then(response => {
  setResponseData_mat(response.data.results.bindings)
})

```

Figure 27 Producer Information Query

```

axios.get('http://localhost:3030/Engineering_Materials_OWL_V1.2.owl/',
  {params: {query: `
PREFIX material: <urn:absolute:Engineering_Materials_OWL_V1.2#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?name (CONCAT(?value," ", ?unitname) as
?value_unit) WHERE {
  ?Che a material:Chemical_Composition .
  ?Che material:has_Element_Couple ?EC .
  ?EC material:has_Element ?El .
  ?El material:has_name ?name .
  ?EC material:has_value ?value .
  ?EC material:has_unit ?u .
  ?u material:has_name ?unitname .
  ?EC material:has_comment ?comm .
  ?Che material:belongs_to ?mat .
  ?mat material:has_name ?mat_name
  FILTER(?mat_name = "${material_name}")
}
ORDER BY ?name
`}}).then(response => {
  setResponseData_chem(response.data.results.bindings)
})

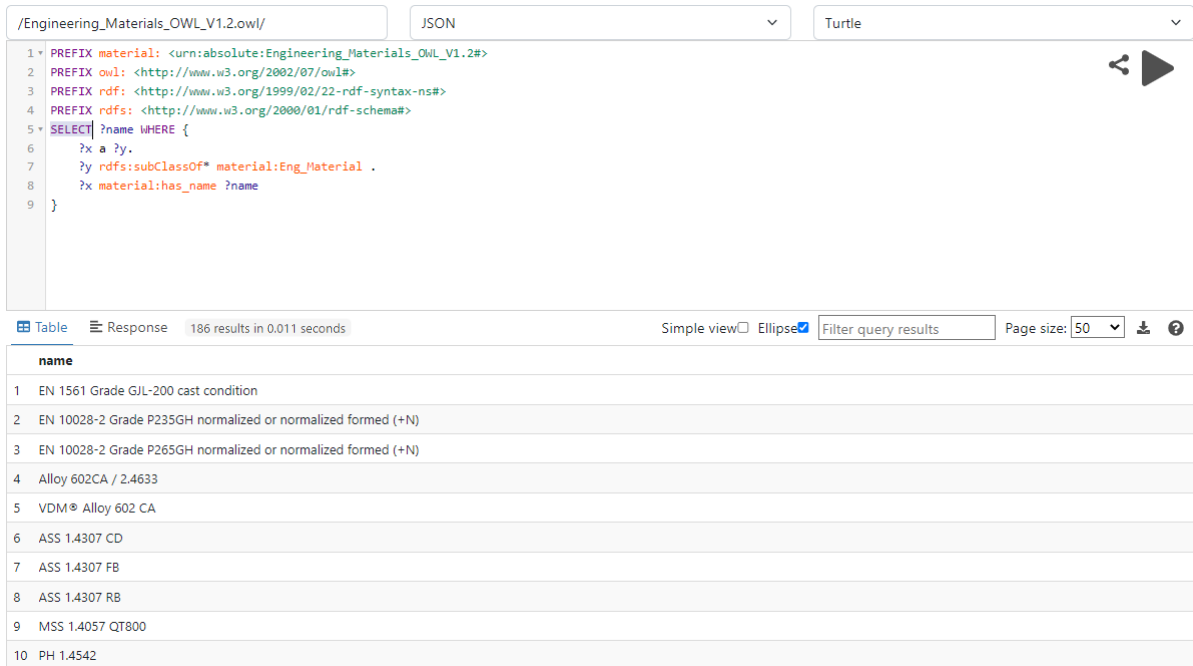
```

Figure 28 Chemical Composition Query

5. Evaluation

In this section SPARQL queries for getting the data from the Ontology using FUSEKI server will be presented. The server is used to create necessary queries for the web application and explore the ontology to get and filter different types of data's in the ontology.

5.1 Query to Select all the Materials



```

1 PREFIX material: <urn:absolute:Engineering_Materials_OWL_V1.2#>
2 PREFIX owl: <http://www.w3.org/2002/07/owl#>
3 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
4 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
5 SELECT ?name WHERE {
6   ?x a ?y.
7   ?y rdfs:subClassOf* material:Eng_Material .
8   ?x material:has_name ?name
9 }

```

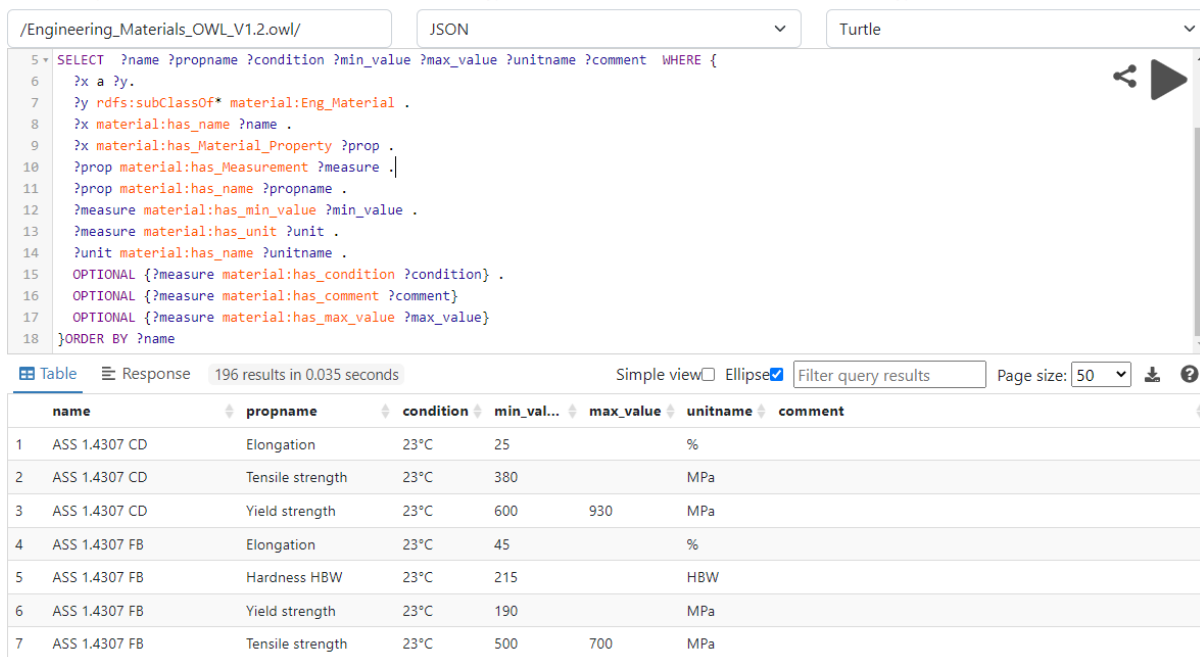
Table Response 186 results in 0.011 seconds

name
1 EN 1561 Grade GJL-200 cast condition
2 EN 10028-2 Grade P235GH normalized or normalized formed (+N)
3 EN 10028-2 Grade P265GH normalized or normalized formed (+N)
4 Alloy 602CA / 2.4633
5 VDM® Alloy 602 CA
6 ASS 1.4307 CD
7 ASS 1.4307 FB
8 ASS 1.4307 RB
9 MSS 1.4057 QT800
10 PH 1.4542

Figure 29 Query to select all the material names

The query in Figure 29 is a basic query where * function is used to select every sub class under the *Eng Material* super class in nested way that it defines all of the classes under the super class. In the first line we are defining x as type y. The response data is a table with only one column with the material names.

5.2 Query to Select all the Material Properties Related to All Materials



```

5 SELECT ?name ?propname ?condition ?min_value ?max_value ?unitname ?comment WHERE {
6   ?x a ?y.
7   ?y rdfs:subClassOf* material:Eng_Material .
8   ?x material:has_name ?name .
9   ?x material:has_Material_Property ?prop .
10  ?prop material:has_Measurement ?measure .
11  ?prop material:has_name ?propname .
12  ?measure material:has_min_value ?min_value .
13  ?measure material:has_unit ?unit .
14  ?unit material:has_name ?unitname .
15  OPTIONAL { ?measure material:has_condition ?condition } .
16  OPTIONAL { ?measure material:has_comment ?comment } .
17  OPTIONAL { ?measure material:has_max_value ?max_value } .
18 } ORDER BY ?name

```

Table Response 196 results in 0.035 seconds

name	propname	condition	min_val...	max_value	unitname	comment
1 ASS 1.4307 CD	Elongation	23°C	25		%	
2 ASS 1.4307 CD	Tensile strength	23°C	380		MPa	
3 ASS 1.4307 CD	Yield strength	23°C	600	930	MPa	
4 ASS 1.4307 FB	Elongation	23°C	45		%	
5 ASS 1.4307 FB	Hardness HBW	23°C	215		HBW	
6 ASS 1.4307 FB	Yield strength	23°C	190		MPa	
7 ASS 1.4307 FB	Tensile strength	23°C	500	700	MPa	

Figure 30 Query to select all the material properties related to all the materials in the ontology

In this query we choose all the individuals under the all of the subclasses under the *Eng Material* super class where we have our material individuals are created. OPTIONAL is used in this query since some *measurement's* do not have all the data properties. This allows us to see the all meaningful results in this query since the variables used in the OPTIONAL are not the at most importance for the users. ORDER BY is also used for the convenience. The result in this query is a table where we can see all the attributes related to the individuals of the sub classes of *Eng Material* super class.

5.3 Query to FILTER The Material Property Yield Strength

```

/Engineering_Materials_OWL_V1.2.owl/sparql
JSON
Turtle

1 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
2 PREFIX material: <urn:absolute:Engineering_Materials_OWL_V1.2#>
3 PREFIX owl: <http://www.w3.org/2002/07/owl#>
4 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
5 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
6 SELECT ?name ?propname (CONCAT(?min_value, " ", ?unitname) as ?value_min_unit) (CONCAT(?max_value, " ", ?unitname) as ?value_max_unit)
   (GROUP_CONCAT( ?app_name; SEPARATOR="," ) AS ?APP_AREAS) WHERE {
7   ?x a ?y.
8   ?y rdfs:subClassOf* material:Eng_Material.
9   ?x material:has_name ?name .
10  ?x material:has_Application_Area ?app .
11  ?app material:has_name ?app_name.
12  ?x material:has_Material_Property ?prop .
13  ?prop material:has_name ?propname .

```

Figure 31 Query to Filter the Yield Strength Value of all materials

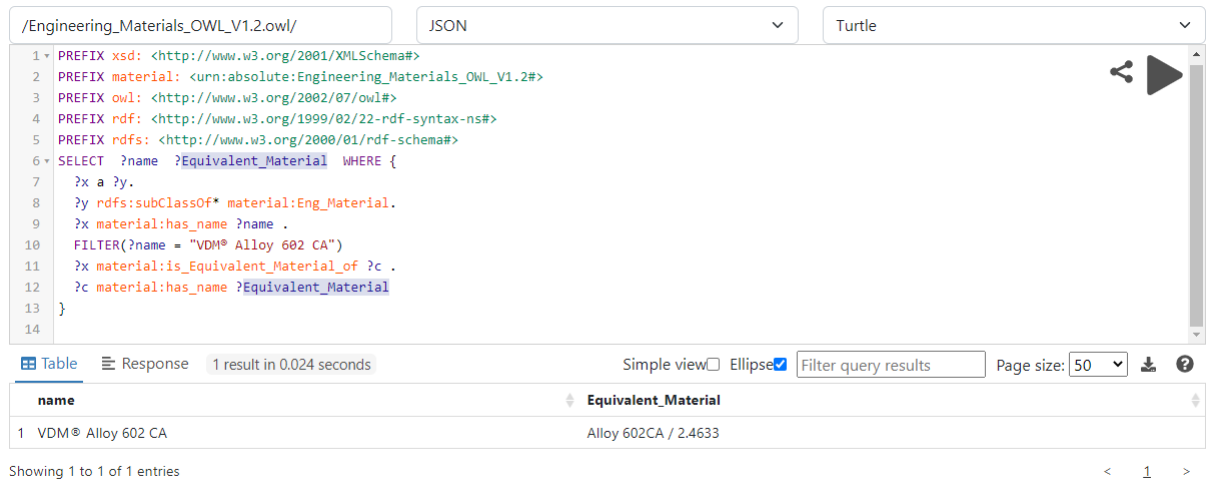
The query shown in *Figure 31* is selecting same variables, but this time it FILTERS the Yield Strength value of the materials. The query uses 2 CONCAT filter functions which unite the value and unit with a separator and a GROUP_CONCAT to unite the application areas with a comma separator.

/Engineering_Materials_OWL_V1.2.owl/sparql		JSON	Turtle
<pre> 10 ?x material:has_Application_Area ?app . 11 ?app material:has_name ?app_name. 12 ?x material:has_Material_Property ?prop . 13 ?prop material:has_name ?propname . 14 ?prop a material:Yield_Strength . 15 ?prop material:has_Measurement ?mes . 16 ?mes material:has_min_value ?min_value . 17 OPTIONAL { ?mes material:has_max_value ?max_value } . 18 ?mes material:has_unit ?unit . 19 ?unit material:has_name ?unitname 20 FILTER(STRENGTHS(str(?y),"Stainless_Steel")) 21 22 } Group by ?name ?propname ?min_value ?max_value ?unitname 23 ORDER BY DESC (xsd:float(?min_value)) </pre>			
Table		Response	
15 results in 0.013 seconds		Simple view Ellipse Filter query results	
Page size: 50			
name	propname	value_min_unit	value_max_unit APP_AREAS
1 UNS S3275...	Yield strengt...	550 MPa	Chemical Industry, Energy, Equipment For Chemical Indsutry, Equipment For Petroleum and Natural Gas ...
2 Sandvik SA...	Yield strengt...	550 MPa	650 MPa Chemical Industry, Steel Pipes and Tubes, Equipment For Chemical Indsutry
3 PH 1.4542	Yield strength	520 MPa	1000 MPa Aerospace, Oil and Gas Industry
4 Sandvik SA...	Yield strengt...	500 MPa	550 MPa Chemical Industry, Steel Pipes and Tubes, Equipment For Chemical Indsutry
5 Acidur 4529	Yield strength	340 MPa	Maritime, Chemical Industry, Consumer Goods, Mining Equipment, Mining Industry, Sea Water Applicati...
6 Sandvik 4C54	Yield strengt...	320 MPa	Automotive, Chemical Industry, Burners and Boilers, Combustion Engines, Electrical Engineering, Manufa...
7 Acidur 4529	Yield strength	300 MPa	Maritime, Chemical Industry, Consumer Goods, Mining Equipment, Mining Industry, Sea Water Applicati...

Figure The results of the query in Fig. 31

The query results are ordered in a descending order by ORDER BY DESC modifier function. OPTIONAL is used to get all the relevant data even if the individual does not have the maximum value. Also in this query xsd:float() used to convert the string variable min value to float so that we can reliably do the filtering.

5.4 Query to Find The Equivalent Material of a Material



The screenshot shows a SPARQL query interface with the following query:

```

1 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
2 PREFIX material: <urn:absolute:Engineering_Materials_OWL_V1.2#>
3 PREFIX owl: <http://www.w3.org/2002/07/owl#>
4 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
5 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
6 SELECT ?name ?Equivalent_Material WHERE {
7   ?x a ?y.
8   ?y rdfs:subClassOf* material:Eng_Material.
9   ?x material:has_name ?name .
10  FILTER(?name = "VDM® Alloy 602 CA")
11  ?x material:is_Equivalent_Material_of ?c .
12  ?c material:has_name ?Equivalent_Material
13 }
14

```

The interface shows the query results in a table view:

name	Equivalent_Material
1 VDM® Alloy 602 CA	Alloy 602CA / 2.4633

Showing 1 to 1 of 1 entries

Figure 32 Query to find the equivalent material

Figure 32 shows the query to find the equivalent material of a material. This query uses a simple FILTER clause where we filter the name of the material which we are looking for its equivalent material in the ontology. The result of this query gives only one variable in this case, but when the materials can have more than one equivalent materials and when the ontology got bigger in the individual count.

5.5 Query to Select the Values Bigger Than The Average



The screenshot shows a SPARQL query interface with the following query:

```

1 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
2 PREFIX material: <urn:absolute:Engineering_Materials_OWL_V1.2#>
3 PREFIX owl: <http://www.w3.org/2002/07/owl#>
4 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
5 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
6 SELECT ?name ?propname ?condition ?min_value ?max_value ?unitname ?comment ?avg WHERE {
7   ?x a ?y.
8   ?y rdfs:subClassOf* material:Eng_Material .
9   ?x material:has_name ?name .
10  ?x material:has_Material_Property ?prop .
11  ?prop material:has_Measurement ?measure .
12  ?prop a material:Tensile_Strength .
13  ?prop material:has_name ?propname .
14  ?measure material:has_min_value ?min_value .

```

Figure 33 Query to Select the values bigger than the average

In this query a nested SELECT function is used to get the average value. AVG aggregator and FILTER clause used to filter the results so that we get the data points which has a minimum value bigger than the average of the minimum values. The inner SELECT function in the nested SELECT query operates first. The first SELECT function then sends the data to the outer and the outer SELECT function calculates and gets the other values such as property name, condition, unit name and comment for displaying in the results. The results are shown in the Figure 34.

/Engineering_Materials_OWL_V1.2.owl/ JSON Turtle

```

14 ?measure material:has_min_value ?min_value .
15 ?measure material:has_unit ?unit .
16 ?unit material:has_name ?unitname .
17 OPTIONAL {?measure material:has_condition ?condition} .
18 OPTIONAL {?measure material:has_comment ?comment} .
19 OPTIONAL {?measure material:has_max_value ?max_value} .
20 {SELECT (AVG(xsd:float(?min_value))as ?avg) WHERE{
21   ?x a ?y.
22   ?y rdfs:subClassOf* material:Eng_Material .
23   ?x material:has_Material_Property ?prop .
24   ?prop material:has_Measurement ?measure .
25   ?prop a material:Tensile_Strength .
26   ?measure material:has_min_value ?min_value .}}
27 FILTER(xsd:float(?min_value) > ?avg)}

```

Table Response 15 results in 0.047 seconds Simple view Ellipse Filter query results Page size: 50

	name	propname	condition	min_value	max_value	unitname	comment	avg
1	MSS 1.4057 QT800	Tensile stren...	23°C	800	950	MPa		"564.375"^^<http://w...
2	PH 1.4542	Tensile stren...	23°C	800	1270	MPa		"564.375"^^<http://w...
3	Acidur 4529	Tensile stren...	20°C	650	850	MPa		"564.375"^^<http://w...
4	UGI® 4545 AIR H900	Tensile stren...		1310		MPa	min.	"564.375"^^<http://w...
5	Sandvik 3R60™ Urea ...	Tensile stren...	23°C	690		MPa		"564.375"^^<http://w...
6	UGIMA® 4542 +P930...	Tensile stren...		930	1100	MPa	min., ASTM A564, EN 10088-3	"564.375"^^<http://w...
7	UNS S32750 hot rolle...	Tensile stren...	23°C	750	930	MPa		"564.375"^^<http://w...
8	EDX 2304 L Profile	Tensile stren...	23°C	630		MPa		"564.375"^^<http://w...
9	Sandvik SAF 2906™	Tensile stren...	20°C	750	800	MPa	min for thickness > 10mm, max for < 10mm	"564.375"^^<http://w...
10	Sandvik SAF 2906™	Tensile stren...	100°C	730	750	MPa	min for thickness > 10mm, max for < 10mm	"564.375"^^<http://w...

Figure 34 Query in Figure 33 continued

5.6 Query to Get The Chemical Composition of A Material

/Engineering_Materials_OWL_V1.2.owl/sparql JSON Turtle

```

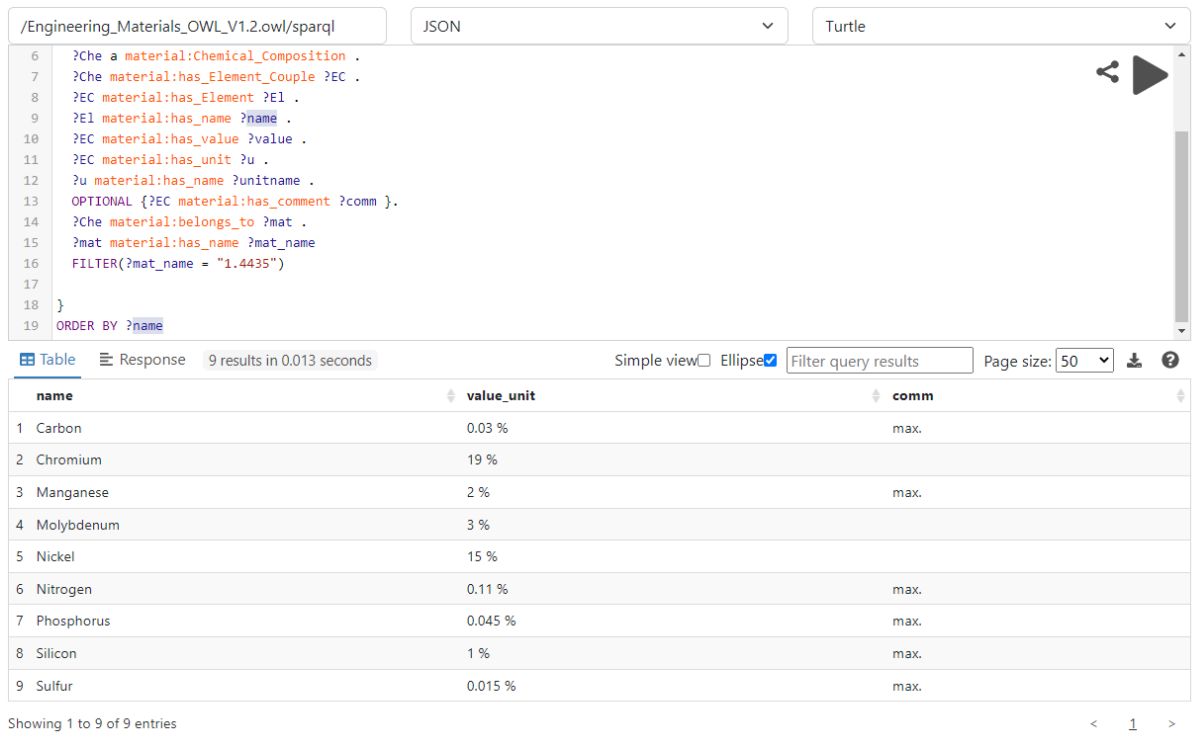
1 PREFIX material: <urn:absolute:Engineering_Materials_OWL_V1.2#>
2 PREFIX owl: <http://www.w3.org/2002/07/owl#>
3 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
4 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
5 SELECT DISTINCT ?name (CONCAT(?value," ",?unitname) as ?value_unit) ?comm WHERE {
6   ?Che a material:Chemical_Composition .
7   ?Che material:has_Element_Couple ?EC .
8   ?EC material:has_Element ?E1 .
9   ?E1 material:has_name ?name .
10  ?EC material:has_value ?value .
11  ?EC material:has_unit ?u .
12  ?u material:has_name ?unitname .
13  OPTIONAL {?EC material:has_comment ?comm} .
14  ?Che material:belongs_to ?mat .

```

Table Response 9 results in 0.013 seconds Simple view Ellipse Filter query results Page size: 50

Figure 35 Query to get the chemical composition of a material

In the query shown in *Figure 35* CONCAT is used for convenient of seeing the value and the unit in one column side by side in space separated format. OPTIONAL is also used as before in previous queries since at most importance value is not comment in this case. FILTER is used to select the names of the materials which equal to "1.4435" and the results shown in *Figure 36* are ordered by the *Element* name in ASC manner. Even though that we did not explicitly wrote ASC after the ORDER BY modifier, the results are in ascending order since this is the default value of ORDER BY.



The screenshot shows a SPARQL query interface. At the top, there is a text input field containing the query path `/Engineering_Materials_OWL_V1.2.owl/sparql`, a dropdown menu set to `JSON`, and another dropdown menu set to `Turtle`. Below these is a text area containing a SPARQL query. The query is as follows:

```

6  ?Che a material:Chemical_Composition .
7  ?Che material:has_Element_Couple ?EC .
8  ?EC material:has_Element ?El .
9  ?El material:has_name ?name .
10 ?EC material:has_value ?value .
11 ?EC material:has_unit ?u .
12 ?u material:has_name ?unitname .
13 OPTIONAL {?EC material:has_comment ?comm }.
14 ?Che material:belongs_to ?mat .
15 ?mat material:has_name ?mat_name
16 FILTER(?mat_name = "1.4435")
17
18 }
19 ORDER BY ?name

```

Below the query, there are tabs for `Table` and `Response`. The `Table` tab is selected, showing 9 results in 0.013 seconds. There are also checkboxes for `Simple view` and `Ellipse`, a text input for `Filter query results`, and a `Page size` dropdown set to 50. The results are displayed in a table with the following columns: `name`, `value_unit`, and `comm`.

	name	value_unit	comm
1	Carbon	0.03 %	max.
2	Chromium	19 %	
3	Manganese	2 %	max.
4	Molybdenum	3 %	
5	Nickel	15 %	
6	Nitrogen	0.11 %	max.
7	Phosphorus	0.045 %	max.
8	Silicon	1 %	max.
9	Sulfur	0.015 %	max.

At the bottom left, it says "Showing 1 to 9 of 9 entries". At the bottom right, there are navigation arrows and the page number "1".

Figure 36 The results and the rest of the query shown in Figure 35

6. Literature Review

In this section, articles in similar topics and existing databases regarding engineering materials are discussed and compared with the *Engineering Materials Ontology* which is discussed in this report.

The first materials ontology which we will discuss is “An ontology-based knowledge framework for engineering material selection” (Yingzhong Zhang, Xiaofang Luo, 2015). The proposed material ontology is created with Protégé and designed to be able to querying via SQWRL language. The ontology in the article contains the engineering material classification, material properties and also it contains the manufacturing process concepts which are also a very important parameter in material selection which that is not included in the *Engineering Materials Ontology*. For the reasons that it goes beyond the aim of this project and for including such classes one needs to make an in depth research about the complex relations between the manufacturing processes and material classification domains which is hard to understand the after effects of the relations. Since each of the relations itself can become another topic of an ontology design. Also in the same journal engineering product concept is included. It is a concept where one can make relations between a finished product and the materials used for the parts of the complete product. The introduced concept is a similar concept to *Application Areas* concept, but in a more detailed way. Such as Combustion Engine as an application area which is included in the proposed ontology in this report, is converted into their parts such as cylinder, piston, exhaust valves, crank shaft and can be related to materials in a similar way like the application area. In the article, they divided the engineering material class into two classes which are rough material and engineering product. In the *Engineering Materials Ontology* under the material class we have only the rough materials in a categorized form. The benefits of such a taxonomy could be creating better relations than the similar approach of application areas class. Also the manufacturing processes class which they included is a big gain regarding the relations in the ontology resulting with a better ontology design.

In this report main focus is to create an ontology which can be generalized to be able to reuse in other ontologies or in some other applications. In other words, the aim was to make a vocabulary of engineering materials which is an open and a shared knowledge system. In the article, the importance of such a semantic data structure is emphasized and claimed that it was the first attempt to create such a system. Most of the citations used in the same article are not semantic web applications and some of them trying to create an algorithm which makes the material selection process automatic.

Most of the existing engineering materials databases such as Matmatch, Totalmateria, knovel are not using semantic web. Although they manage to give a good user experience, it is clear that the field itself lacks the semantic web technology.

The most similar ontology frameworks are Yingzhong Zhang, Xiaofang Luo(2015) and Ashino(2010)'s proposed ontologies. The structure wise they are similar except Yingzhong Zhang, Xiaofang Luo(2015) has more general classifications related to materials domain.

Comparing the first ontology with our ontology about user interface. The proposed ontology's user interface fails since it is too simple for such an ontology which requires detailed information while selecting the materials.

Ashino (2010) describes an ontology structure for the materials ontology where they structure the ontology very similar to the structure of the proposed ontology in this report. In the report, Ashino (2010) uses an approach where he categorizes the engineering materials by their chemical compounds. That is one of the differences between the ontology he proposes and the ontology proposed in this report.

If we needed to make a general comparison between 2 of the mentioned ontologies and the proposed ontology discussed in this report, one can say that the proposed ontology could be a transition ontology between those two ontologies.

The mentioned articles and already existing websites are the primary inspiration for this project. Even after looking at the some of the first attempts in this field, one can see that it is still a field that is improving over time.

7. Conclusion

In this report, a knowledge based framework for engineering materials is designed. The aim and concept of the ontology covered. How the classes, data properties, object properties and rules are created were discussed. The data connection to the web application and the user interface is discussed. SPARQL query examples are shown to analyse the effectiveness of the ontology and to discover the flexibility of the SPARQL.

Furthermore, we can discuss a few points where we can improve our ontology. The first one is that the spectrum of the material types, material properties and application areas could be increased. This will allow the ontology to hold more information without changing the structure. Secondly, while creating individuals, a more structured ways should be followed in the future, so that error margin for mistypes is lowered.

Unit conversion could be added to the ontology as another class which could be used as follows:

HBW -> has unit conversion coefficient-> HBW to H

Then the unit conversion can be made on the user interface, so that users can view the values in according to their metric systems.

The similar concept to manufacturing processes could be added to the ontology in a simplified version such that creating new relations in the ontology regarding the which materials could be used in a specific manufacturing process. Enriching the ontology will result in a better result oriented system.

Since the materials ontology is designed to select a material according to the design requirements, in the future *Design Requirements* class could be added so that users could search directly through the requirements input and alongside the SPARQL queries an algorithm could help to identify the required SELECT queries according to design requirements.

Other than these, the project was successful in the end. Since the aim of the project was to discover the ontology structure, gain a familiarity with rdf, rdfs, OWL and SPARQL languages.

8. References

1. Reactjs, 2022. Getting Started. [online] Available at: <<https://reactjs.org/docs/getting-started.html>> [Accessed 5 December 2022].
2. Ashino, T., 2010. Materials Ontology: An Infrastructure for Exchanging Materials Information and Knowledge. *Data Science Journal*, 9, pp.54–61. DOI: <http://doi.org/10.2481/dsj.008-041>
3. Yingzhong Zhang, Xiaofang Luo, Yong Zhao, Hong-chao Zhang, 2015. An ontology-based knowledge framework for engineering material selection. *Advanced Engineering Informatics*, Volume 29, Issue 4, Pages 985-1000. DOI: <https://doi.org/10.1016/j.aei.2015.09.002>
4. W3C, 2015. SEMANTIC WEB. [online] Available at: <<https://www.w3.org/standards/semanticweb/>> [Accessed 5 December 2020].
5. Borst, WN & Borst, WN 1997, 'Construction of Engineering Ontologies for Knowledge Sharing and Reuse', University of Twente, Enschede.