Department of Computer & Information Sciences

University of Strathclyde Glasgow

Classwork: Building and Processing an Ontology in OWL

CS549: Distributed Information Systems

Pavithran Rajan M.Sc. in Advanced Computer Science

Registration Number: 202267111
Lecturer: Dr John N Wilson
Date of Submission: 04/04/2022

Description of the Ontology:

Applying ontologies to these web pages has never been more relevant with the development and deployment of artificial intelligence to the world wide web. This classwork aims to design an ontology for the part of the business of the Strathclyde university student union. When designing an ontology, the first thing to consider is establishing the domain and scope. This ontology hopes to cover a wide range of classes and concepts, on basic inspection for nouns that could correspond to objects and verbs that could provide some relations. The terms that came across through the initial assessment were an event, news, society, organiser, club, student, teacher, community, president, advice on academic, financial, health and well-being, housing, opportunity, and course. Some instances are created to represent the self-standing classes like the societies and clubs can be represented realistically. Friend of a friend or FOAF ontology is imported to our ontology which contains concepts of people and relationships concerning a web platform.

Figure 1: The screenshot of the Strathclyde Student Union Website.

One of the best ways to establish and verify the ontology is to create competency questions that the knowledge base can answer. Some of the competency questions that these ontology design answers are:

1. What are the names of the students who study for a postgraduate degree?
2. What is the age of the teachers who teach a course?
3. What is the identification number of the student who wants advice regarding finances?
4. Where is the event held, and who organises it?
5. What is the latest news that is posted by a society?

This ontology is tested against these competency questions, and our design answers all of them.

Elements of the Ontology:

This ontology contains several self-standing classes, modifier classes, relations and definable. The self-standing classes include person, course, event, venue, and organisation—several modifiers like gender, advice kind, name, ID no, email ID, and News. The modifiers like name, and email id, contain string values, and the identification number has integer values. The advice that a student may require can be either academic-related, Finance related, Heath and well-being related or regarding Housing. The relations cover all possible object properties within this ontology.

The definitions for all definable are listed below.

a) Organiser – An organiser is a person who organises an event.
b) Advisor - Advisor is a person who advises a person.
c) President – President is a person who president an organisation.
d) Student – A student is a person who studies a course.
e) Lecturer – A lecturer is a person who teaches a course.
f) Representative – Representative is a person who represents a course.

Self-standing Classes	Modifiers	Relations	Definable
Person Course • Undergraduate Course • Postgraduate Course • Online Course • Diploma Course Event Venue Organisation • Club • Society • Community	Gender • Male • Female Advice • Academic Advice • Financial Advice • Health & Well-being Advice • Housing Advice Name Identification_No Email_ID News	Studies Teaches Organises Represents isPresidentof IsGenderof Advises wantsAdvice isPartof Heldin Hosts hasName isNameof hasIdentification_No hasEmail_ID hasNews	Organiser Advisor President Student Lecturer Representative

Table 1: Ontology Elements

The figure below contains the ontology diagram for the student union website. The classes included in the yellow outlined boxes are definables, and the modifiers are represented in blue outlined ovals.

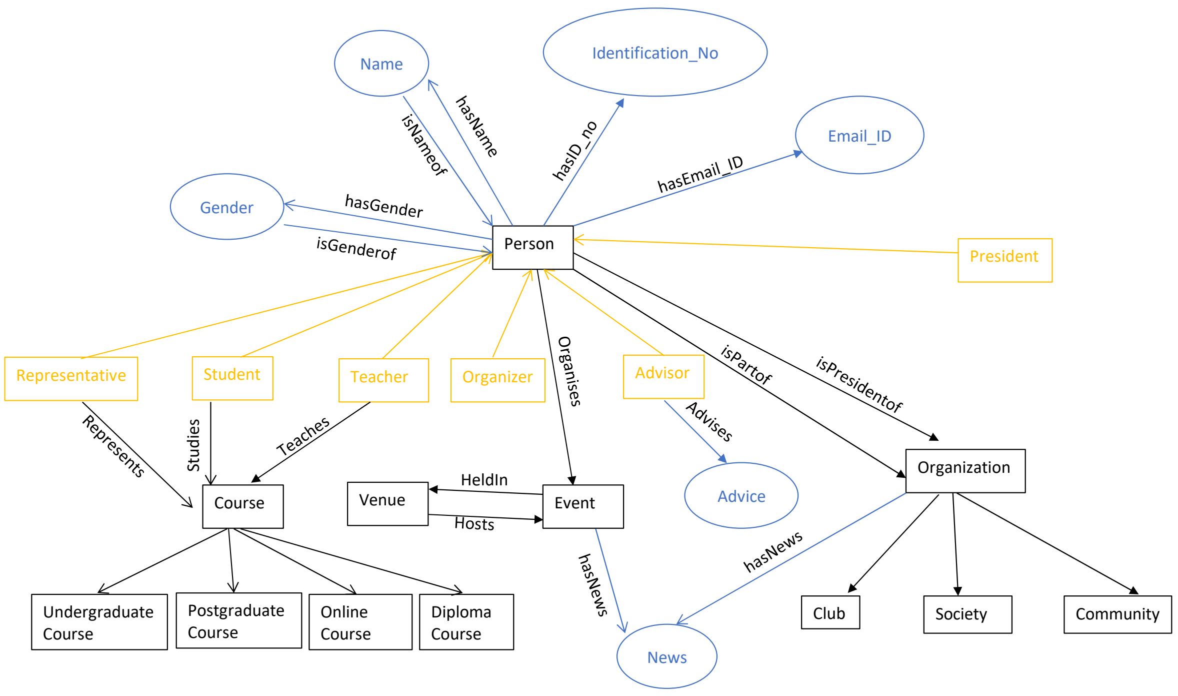


Figure 2: Ontology Diagram

The table below shows the domain and range of object properties and their functional property. Each student can study only one course for property Studies, but it can have many students; therefore, it is a functional property. The domain lecturer can teach many classes for teaches object property, and a course can have many lecturers. Consequently, it's neither function nor inverse functional property. Another example with hasIdentification_No relation, the domain person can have only one ID number, and one ID number can be assigned to only one person. There was no relation with the transitive, symmetric or reflexive property; therefore, it is not included in the table below.

Relation	Domain	Range	Functional Property	Inverse Functional Property
Studies	Student	Course	x	
Teaches	Lecturer	Course		
Organises	Person	Event		
Represents	Student	Course	x	x
Presidentof	Person	Organisation	x	x
IsGenderof	Gender	Person		x
hasGender	Person	Gender	x	
Advises	Advisor	Person		
wantsAdviceon	Person	Advice		
isPartof	Person	Organisation		
Heldin	Event	Venue	x	
Hosts	Venue	Event		x
hasName	Person	Name	x	
isNameof	Name	Person		x
hasIdentification_No	Person	Identification_No	x	x
hasEmail_ID	Person	Email_ID		x
News_is	Event, Organisation	News		

Table 2: Object properties with their Domain, Range and Properties

The below table shows the list of datatype properties and their domain and range. Where the domain is the modifier, and the ranges are some datatypes. For example, name_is had the domain name, and the range is the datatype string.

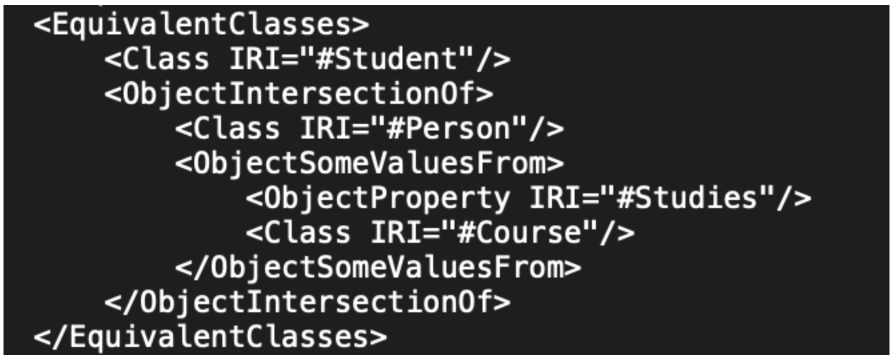
Datatype properties	Domain	Range
Name_is	Name	xsd:string
Course_is	Course	xsd:string
Identification_No_is	Identification_No	xsd:integer
Email_ID_is	Email_ID	xsd:string
News_section	News	xsd:string

Table 3: Datatype properties with its Domain and Range

Some individuals are created so that the classes are represented realistically. This includes instances of students, teachers, advisors, courses, society, events, and location. These individuals are used to describe the concepts of class hierarchy in the knowledge base.

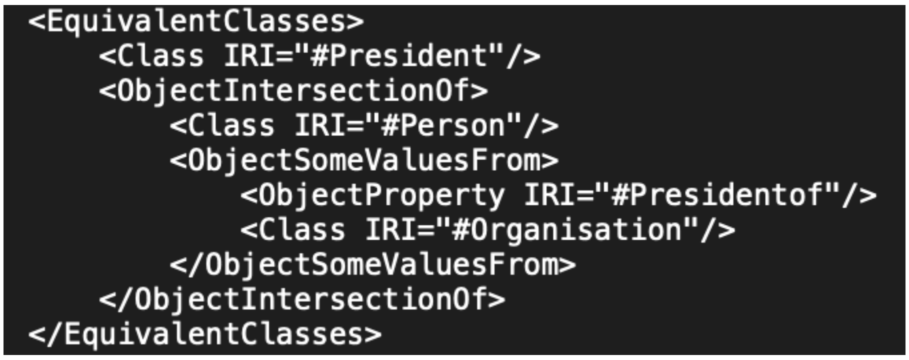
RDF/RDFS/OWL Extracts:

The RDF/OWL extract below demonstrates that the "Student" is a definable class in this Student Union ontology. The Student is defined as a person who studies a course. This is declared using 'Person and (studies some Course).' The definable class student is declared by creating an anonymous class and making it equal to the student. It should be noted that the anonymous class is constructed out of a restriction.



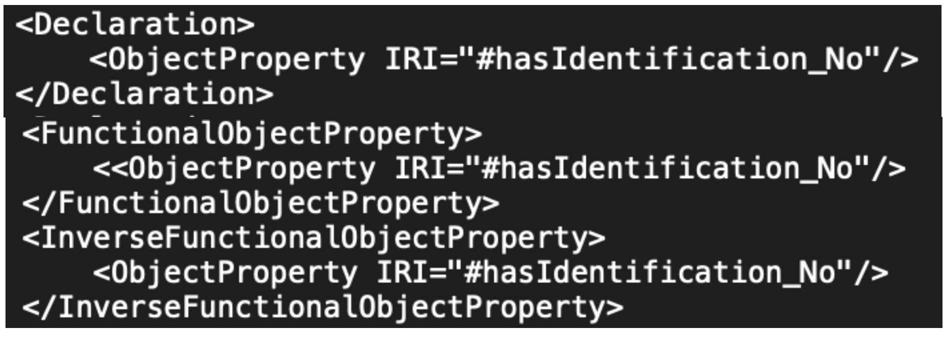
```
<EquivalentClasses>
    <Class IRI="#Student"/>
    <ObjectIntersectionOf>
        <Class IRI="#Person"/>
        <ObjectSomeValuesFrom>
            <ObjectProperty IRI="#Studies"/>
            <Class IRI="#Course"/>
        </ObjectSomeValuesFrom>
    </ObjectIntersectionOf>
</EquivalentClasses>
```

The RDF/OWL extract below demonstrates that the “President” is a definable class in this Student Union ontology. The Student is defined as a person who is a president of an organisation. This is declared using ‘Person and (isPresidentof some Organisation).’ The definable class President is declared by creating an anonymous class and making it equal to the student. It should be noted that the anonymous class is constructed out of a restriction. These sections are significant to understanding how the definables are expressed in an RDF/XML code.



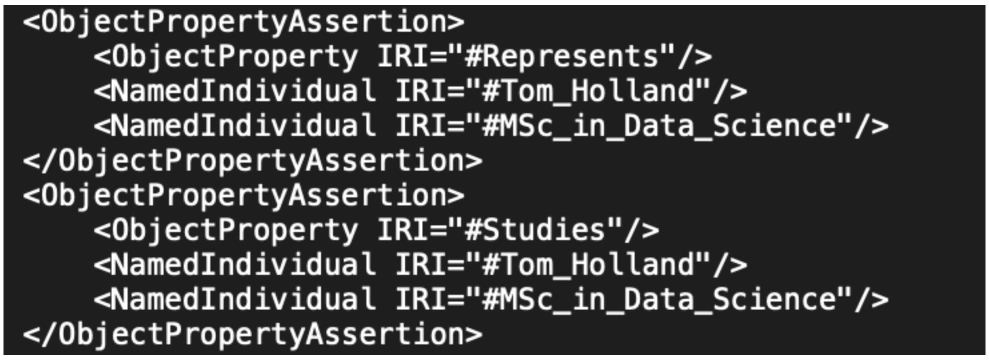
```
<EquivalentClasses>
    <Class IRI="#President"/>
    <ObjectIntersectionOf>
        <Class IRI="#Person"/>
        <ObjectSomeValuesFrom>
            <ObjectProperty IRI="#Presidentof"/>
            <Class IRI="#Organisation"/>
        </ObjectSomeValuesFrom>
    </ObjectIntersectionOf>
</EquivalentClasses>
```

The code below demonstrates the RDF/OWL code of the object property hasIdentification_No declared and has both functional and inverse functional properties. This means that for each instance of domain person has only one identification number as a range; therefore, it is a functional property. At the same time, each identification number can only be assigned to one person. Consequently, it is an inverse functional property. This section is noteworthy because it depicts how the characteristics of relations like functional property are expressed in an RDF/XML code.



```
<Declaration>
    <ObjectProperty IRI="#hasIdentification_No"/>
</Declaration>
<FunctionalObjectProperty>
    <<ObjectProperty IRI="#hasIdentification_No"/>
</FunctionalObjectProperty>
<InverseFunctionalObjectProperty>
    <ObjectProperty IRI="#hasIdentification_No"/>
</InverseFunctionalObjectProperty>
```

The code extract below demonstrates some of the assertions that are made for the instance of the student Tom Holland. This indicates that tom holland studies the MSc in data science course, and he is the student representative of that course. This section is significant to understanding how individuals and their object property assertions are represented in an RDF/XML code.



```
<ObjectPropertyAssertion>
    <ObjectProperty IRI="#Represents"/>
    <NamedIndividual IRI="#Tom_Holland"/>
    <NamedIndividual IRI="#MSc_in_Data_Science"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="#Studies"/>
    <NamedIndividual IRI="#Tom_Holland"/>
    <NamedIndividual IRI="#MSc_in_Data_Science"/>
</ObjectPropertyAssertion>
```

SPARQL Queries:

1) Traverse at least three predicates in the graph and involve at least one predicate from an imported ontology.

   Question: What are the name and the age of the teachers who are part of the sports club?

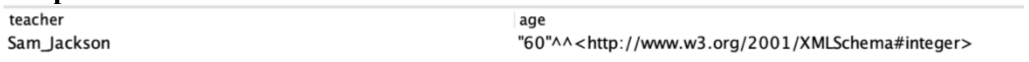
   Code:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX union: <http://www.StudentUnion.com/ontologies/Union.owl/#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>


SELECT DISTINCT  ?teacher ?age

WHERE
{
  ?teacher union:Teaches ?Course;
           union:isPartof ?Sports_Club;
           foaf:age ?age.
}
```

   Output:



```
teacher                                   age
Sam_Jackson                               "60"^^<http://www.w3.org/2001/XMLSchema#integer>
```

2) Use a FILTER

   Question: Display the person's name and age whose age is above 25.

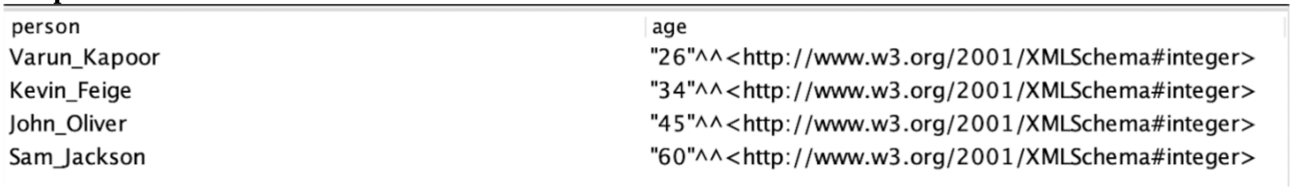
   Code:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX union: <http://www.StudentUnion.com/ontologies/Union.owl/#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>


SELECT DISTINCT ?person  ?age

WHERE
{
  ?person foaf:age ?age
  FILTER(?age > 25)
}
ORDER BY ?age
```

Output:



```
person                       age
Varun_Kapoor                 "26"^^<http://www.w3.org/2001/XMLSchema#integer>
Kevin_Feige                  "34"^^<http://www.w3.org/2001/XMLSchema#integer>
John_Oliver                  "45"^^<http://www.w3.org/2001/XMLSchema#integer>
Sam_Jackson                  "60"^^<http://www.w3.org/2001/XMLSchema#integer>
```

3) Carry out an aggregate operation using a 'group by' clause.

Question: List the Identification Number of teachers.

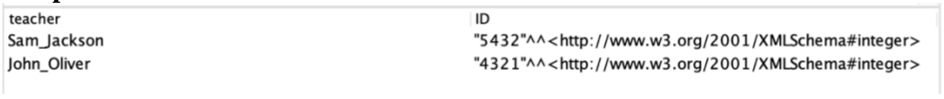
Code:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX union: <http://www.StudentUnion.com/ontologies/Union.owl/#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>


SELECT  ?teacher ?ID

WHERE
{
  ?teacher union:teaches ?Course;
           union:Identification_No ?ID.
}
GROUP BY ?teacher
```

Output:



```
teacher                      ID
Sam_Jackson                  "5432"^^<http://www.w3.org/2001/XMLSchema#integer>
John_Oliver                  "4321"^^<http://www.w3.org/2001/XMLSchema#integer>
```

4) Check for the existence of a fact in your collection.

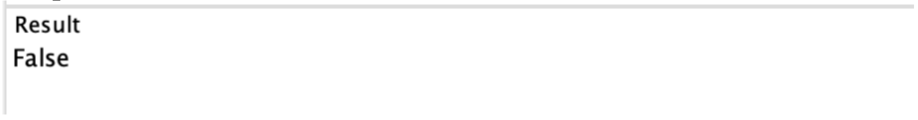
Question: Is Chris Evans older than Tom Holland.

Code:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX union: <http://www.StudentUnion.com/ontologies/Union.owl/#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>


ASK
{
  ?x foaf:age ?Chris_Evans .
  ?y foaf:age ?Tom_Holland .
  FILTER(?x > ?y) .
}
```

Output:



```
Result
False
```

5) Generate RDF output from a query result.

Question: Generate an RDF graph AttendedBy that links Courses to Students.

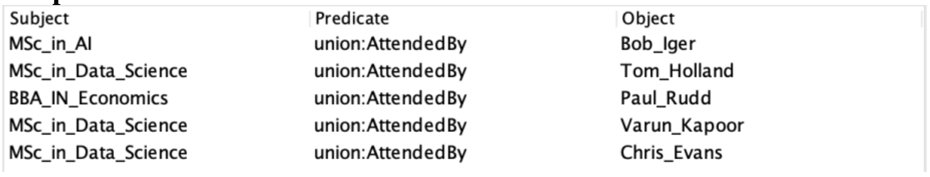
Code:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX union: <http://www.StudentUnion.com/ontologies/Union.owl/#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>


CONSTRUCT
 { ?Object union:union:AttendedBy ?Subject}

WHERE
{
   ?Subject union:Studies ?Object.
}
```

Output:



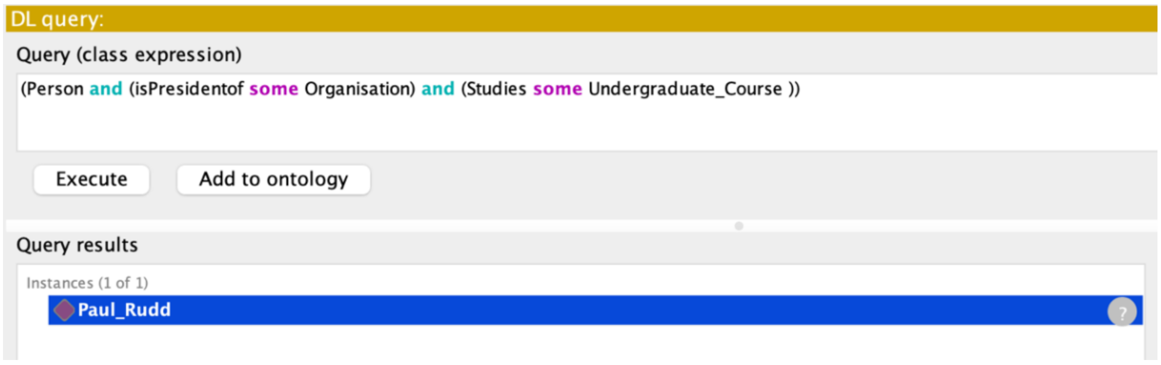
Subject	Predicate	Object
MSc_in_AI	union:AttendedBy	Bob_Iger
MSc_in_Data_Science	union:AttendedBy	Tom_Holland
BBA_IN_Economics	union:AttendedBy	Paul_Rudd
MSc_in_Data_Science	union:AttendedBy	Varun_Kapoor
MSc_in_Data_Science	union:AttendedBy	Chris_Evans

Description Logic Query:

The description logic query is used to check concepts and assertions in the ontology. The below query looks for instances of undergraduate students that are presidents of any organisation. In this ontology, the instance Paul Rudd is returned as the output. This is an expected result as the individual Paul Rudd studies BBA in economics and is the president of the yoga club. This is only possible because the class president is a subset of the class student. This means that only students can be presidents of clubs and societies in the union.

Query: `(Person and (isPresidentof some Organisation) and (Studies some Undergraduate_Course ))`

Output:



```
(Person and (isPresidentof some Organisation) and (Studies some Undergraduate_Course ))
```

Instances (1 of 1)
Paul_Rudd

Critique of the Ontology Design:

There is no one correct method for designing an ontology. There could always be a solution that could develop better alternatives to an ontology. The ontology is created and processed using protégé. This ontology design is intuitive and works well enough to address the problem domain. The ontology is a preliminary design because it does not include all elements of the student union website like job search, feedback, and student union store. The part-time and volunteering opportunities of the section to apply for jobs within the student union also need to be included to cover all the areas of the student's union website. Even with the included concepts like advice or course, the design is pretty concise and is the most straightforward representation of these concepts. The method assumes only a few roles within the student union. Their hierarchy of positions, like the vice president or assistant lecturer etc., is virtually non-existent in my design. The advice section needs to contain the link to contact the advisor and obtain available resources for students to access. Additional concepts need to be established to fully realise this section of the student union website. There are few classes like the online course has no individual course created or has any individual taking this course or teaching it. This creates problems when creating axioms and representing the ontology unrealistically. No hierarchy within the definable class person shows who has higher privileges to access this ontology.

The imported friend of a friend (FOAF) is a small ontology but has other object and data properties that can be incorporated into my design. It has many valuable concepts like date of birth and social media IDs relevant to our design. Even though it comes with minor syntactical issues, it is beneficial because there is implicit knowledge of the triples in this ontology. Friend of a friend ontology is used by many websites, particularly some that hosts blogs and journals. This ontology is particularly useful when incorporated into a workplace or an organisation's platforms. It's noteworthy to mention that we use age data type property in our design to assign age for individuals like students.

The successful execution of the SPARQL query and the description logic query is an indication that the design is well defined and that there are no inconsistencies in our design. Especially while running the HermiT reasoner, some of the discrepancies that were raised were addressed and produced meaningful inferences for the ontology. The successful execution of the description logic query is also indicative of a competent design.

The news section, a significant section on the student union website, is reduced to a string datatype. The news is not only restricted to text. It also has no concept to specify who can post this news and moderate it. Several kinds of news need to be identified like long term information boards like covid-19 information, learning resources or short time news regarding events around the university. There is no design consideration for logging in to the student union portal to access its many enquiries. We can improve this design by including other information to store passwords so that we can use it to log in to the website.

The design could include concepts incorporated from other domains within the university like the Strathclyde library or Pegasus that contain many of the concepts present in this design. One example of this consideration can be that the Strathclyde library includes a list of venues and has some of its events that pertain to clubs and societies of the student union. Another example is the use of personal information like name, ID, or email address could be incorporated from the Pegasus domain. One of the alternate approaches is to entirely eliminate information about students as it can be obtained from the Pegasus domain and improve access to resources pertaining to academic, financial, health and wellness, or other enquires while reducing the complexity of this ontology.

The ontology design is not necessarily a one-step process. The procedure can be iterated further to produce a much better plan that represents the domain in an elegant way and in a way that can be maintained easily by the administrators. A better ontology can be created by improving the existing concepts and importing useful concepts from other domains.