University System
Semantic Data Technologies
MOD004979

SID-2015832

Contents

1. Introduction……4-5
   1.1 Semantic Data Technologies……4
   1.2 Concept……4
   1.3 Aim……5
2. Design……5-7
   2.1 Semantic technologies……5
      2.1.1 Resource Description Framework (RDF)……5
      2.1.2 Language for WebOntologies -OntologyWebLanguage (OWL)..6
      2.1.3 Query Language (SPARQL) ……6
      2.1.4 User Interface……7
3. Ontology Design…… 7-14
   3.1 Class Hierarchy……7
   3.2 Class Visualization……9
   3.3 Object Properties……10
   3.4 Data Properties……12
   3.5 Individuals……13
4. Implementation……15
   4.1 Fuseki Server……15
   4.2 Project workflow……16-18
      4.2.1 ARU project folder……16
         4.2.1.1 Controller……17
      4.2.2 ARU.Source project folder……19
         4.2.2.1 Server connection……19
         4.2.2.2 Query.cs……19
   4.3 Project Execution……20
5. Evaluation……22-26
   5.1 SPARQL Query Examples……22
6. Critical review……27
7. Conclusion……28

References……29

Figures

Figure I: Building blocks of Semantic Web (Giuseppe Lugano,2005)…… 6
Figure 2: Class hierarchy……8
Figure 3: Ontology metrics……9
Figure 4: Ontology Graph……..10
Figure 5: Object properties. ……11
Figure 6: Description of Object Property *belongsto* ……..11
Figure 7: Showing Inverse Object property……..12
Figure 8: Data properties Hierarchy……...13
Figure 9: Working Hermit Reasoner……..14
Figure 10: Example for Individual……15
Figure 10: Fuseki windows batch file execution showing port number for local server……15
Figure I l: Apache Jena Fuseki server Interface……15
Figure 12: successful uploading of ontology to Fuseki Server……16
Figure 13: Overview of project folder……..17
Figure 14: View of controller folder……..18
Figure 15: Model folder in ARU.Source ……..18
Figure 16: Snippet of copy of ontology in project folder……19
Figure 17: Complete Project folder in Visual Studio……..19
Figure 18: Output after building the project – User Interface…… 20
Figure 19. Results of one query in the interface……21
Figure 20. Query to filter Distinct classes to use them as filter in UI……22
Figure 21. Results of filters used in the interface. ……22
Figure 22. Query to count the class preview ……..23
Figure 23. Result for class preview ……23
Figure 24. Query to display details of person……..24
Figure 25. Query using *filter* function to filter persons belong to Cambridge with conditions24
Figure 26. Results for the Query with filter function……25
Figure 27. Query include set of filter conditions……25
Figure 28. Results for the above query with multiple filters……25
Figure 29. Query for student being assistant to lecturer……26
Figure 30. Results of research student who are assistants……26

1. Introduction

The details of the planning and development of the University's online application has been utilized with semantic web techniques and assessment which has been discussed in this paper. As the application requires in great depth about the study which will go into the architecture of the University ontology has been followed. The report will then go through how the program was developed, as well as any design decisions made during this time.

"Displaying research student who is also project assistant to lecturer in university" is a complex question which is allowed by end-users to deliver. Through a mix of SPARQL queries and SPARQL endpoints, the Semantic Data Fusion excels at resolving such complicated inquiries. I assessed the efficacy of the SPARQL queries created to interface with the application's taxonomy. The application is then analyzed and compared to current similar solutions in a critical analysis of the endeavor.

The application was created with the goal of retrieval of information on a data set in the field of education based on user-defined queries, to contribute to the idea that it is easy to combine existing data from multiple sources by structuring it, so that when it is made available, it can provide more meaningful data on the topics began searching by the user.

1.1 **Semantic data technologies**

This application has been created to investigate the advantages of semantic data technologies in web-based businesses. The collection of approaches and technologies that support machines to comprehend the meaning and subject of data is known to be as Semantic Data Technologies.

"Semantic web is a concept of how computers, people, and the Web can work together more effectively" (Thomas B. Passin, 2004)

Semantic search, which tries to provide better suggestions by combining open data sources, is used. It uses keyword-to-concept representation to combine the context, intent, and idea of a particular request.

It promotes the use of centralized data search to obtain greater prevalence of the material by utilizing the capabilities of semantic technologies to turn this scattered information into large datasets. This might help businesses with essential parts of their operations, such as providing better services to their users and making data-driven decisions.

1.2 **Concept**

The developed UI interface is an internet application that lets users to easily navigate through information on the network. It offers a few features, including searches, that are designed to make the system as user-friendly as possible while still preserving some flexibility.

The categories are in a left-hand column, the search box is in a top-level, and the responses are to the right of the filters and under the search box. Users that utilize such services on a regular basis should be able to recognize this.

Why? - The first was a possible re-implementation of both the university's network (aru.ac.uk)

and the identification of the use of ontology to develop semantic web and semantic search.
It has a set of features, as well as text-based searches, to make the system as user-friendly as possible while yet retaining some versatility.

Because of the wide range of student kinds and classifications, the educational system is a good candidate for applying semantic data technology. Developing ontologies to encompass most of those categories and groupings as classes would help the author to connect these subclasses together in a consistent manner to connect comparable metadata to individuals. It does away with the necessity for a distinct, frequently hard-coded list of categories to classify items in a database table.

Ontology's scope and purpose:
Ontology is used to ensure that the university runs smoothly. The benefits of employing this ontology include the ability to reuse or connect it into every university to simply access and retrieve machine-process able information. It will be helpful for person searching for student or lecturer based upon their qualification, project, publications, location, and many combinations of filters. This software's main purpose is to help users browse a huge number of students and staff at a university by running complicated SPARQL queries to extract relevant information.

The ontology's targeted users: The ontology's stage of the design thinking must be identified.
The targeted users are listed below:
1. Workers of the University, for starters.
2. University students who are considering enrolling.
3. University students currently enrolled.
4. University of ARU alumnae.
7.Researchers are hard at work on their research projects.
8. Recruiters.

1.3 Aim

Similarly, the purpose of this study is to detail the development of a semantic web - based application, starting with the ontology design, query formulation, and linkage to a graphical interface for the end-user. Explicitly describing the development tools utilized, as well as the obstacles and potential solutions encountered during this procedure are given in this report.

2. Design

2.1Semantic technology concepts
2.1.1Resource Description Framework (RDF)
Many attempts to establish the underlying technologies for developing the semantic web have been made as a result of the need to include more knowledge and documents. The Resource Description Framework was by far the most prominent of them (RDF). The semantic web's evolution of RDF centers on representing arbitrary types of broad information or relating

information about the relationships between objects in the actual world such as concepts, people, and locations (Tauberer, 2006).

Triples is known as to what RDF provides a generic, flexible way for breaking down practically any data into little chunks. A statement can be as Subjects Predicate Object triple. Names such as subjects, predicate, and object which resources are known to be as Entities. Objects that represent anything, such as a person, a website, or something even more abstract, such as relations or states is known as Entities.
Individual belongs to Campus, for illustration, is a trio with Person as well as Campus as classes and belonging to as a variable that connects them. Text values, often known as literal values, can be assigned to objects. (Tauberer, 2006)

2.1.2 Language for Web Ontologies -Ontology Web Language (OWL)
The dearth of a standardized ontology vocabulary has found it challenging to exchange and utilize ontologies between various applications inside the same domain across domains that are connected. The grammar and interpretations of ontologies representing equivalent information differed greatly depending on the taxonomy language employed. The World Wide Web Consortium (W3C) devised the Web Ontology Language to overcome this interoperability challenge and propose a common paradigm for web-based interchange of ontological information (OWL)
Knowledge specialists and software developers may easily construct, alter, link, and integrate ontologies in a dispersed context using OWL.
"The Web Ontology Language (OWL) is a more extensive and complete language to model concepts as a foundation for logic reasoning and agents" (Giuseppe Lugano, 2005).

Fig 1. Building blocks of Semantic Web (Giuseppe Lugano,2005)

OWL is a natural complement of XML and RDF, which provide a syntax and a model, without referencing to the actual semantics. OWL ontologies are a synonyms of RDF vocabularies; basically, they are collections of classes of nodes and classes of properties. OWL makes it possible to reason upon metadata.

The next stage is to use SPARQL to access information that once ontology has been developed. The programming language for RDF is SPARQL, which treats the RDF as a repository of Subject-Predicate-Object triples.

2.1.3 Query Language (SPARQL)

The declaration of the domain/namespace where the inquiry is going to grow is associated to a brief label with a specific URI that is about a simple SPARQL question is formed of a PREFIX keyword.
A query's SELECT clause specifies the pieces of data it will output. The FROM phrase indicates the data that will be used to conduct the query. In this case, the query refers to the PREFIX declaration's URI. Numerous FROM keywords are allowed in a query. Finally, similarly with SQL, the WHERE clause specifies filters to receive just the data that meets those criteria (Dodds, 2005).

2.1.4 User Interface

Outcomes of UI

Student search - The search interface includes an input area for searching all student-related information. The output will be a list of students who match the search keyword, together with all relevant information from the Ontology. A short explanation of the student is included in the result list, as well as the student's name, Student ID, city, gender, activities, course, program manager, and publications.

A modal box appears when you click the name of one of the individuals shown in the result list. The Student ID, city, gender, projects, semester, project supervisor, and publications are all displayed in the modal window. This capability lets for the delivery of important user information without the need to hunt for further information, conserving the user time while also giving an answer.

3. Ontology Design

3.1 Class hierarchy

I created the University ontology using Protégé 3.4, an open-source tool developed by Stanford University's (Protégé, 2019). The basic goal of these ontologies is to use classes to categorize objects in terms of syntax or meaning. The Ontology generated for the student search application may be seen in Figure 1. The subclasses page on the Protégé interface displays the hierarchical system for our Ontology, including classes and subclasses.

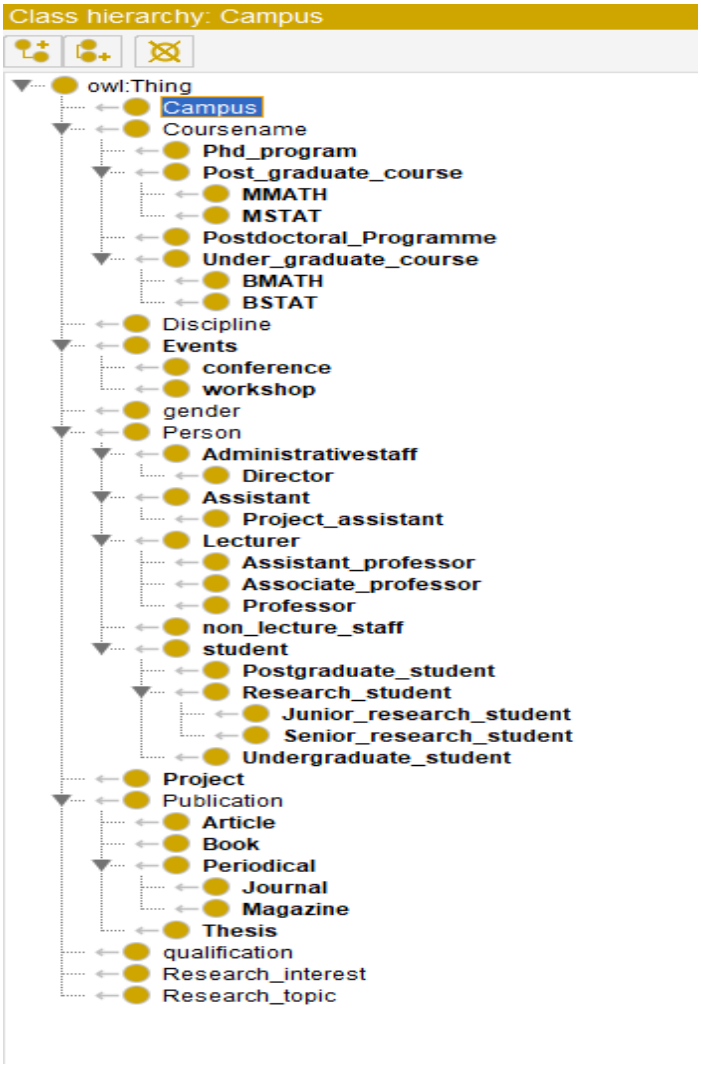

Fig 2. Class hierarchy.

The main classes are Campus, Course name, Discipline, qualification, research interest, Events, gender, Publications, and Person. The subclasses represent the category division for every class. For example, Article, Book, Periodical, and Thesis are subclasses of publications. Subclasses like Periodical can also have subclass. In this case, Journal and Magazine are subclasses for Periodical subclass.

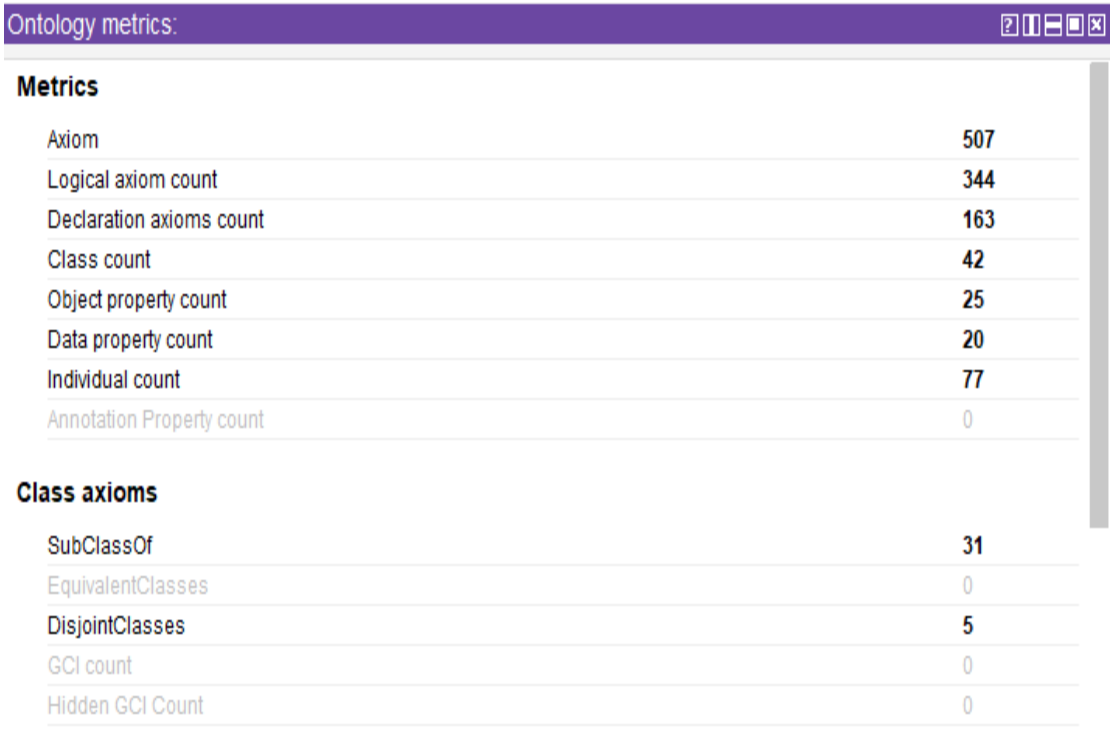

Fig 3. Ontology Metrics

There are about 507 axioms in all, 1346 triples, 77 persons, 42 classes, 25 object attributes, and 20 data properties. Thirty-one of the forty classes are subclasses for classes. 5 of the 42 classes are deemed 'disjointed,' meaning they cannot be related to the same person since they are in some manner antagonistic to one another.

3.2 Ontology visualization: An ontology graph is used to visualize the relationships between distinct concepts and their attributes. The fundamental class by which all classes are formed is campus, as shown in Figure 3. Ontology graph shows the class hierarchy with all the connections to reach any classes or subclasses created using object properties. Classes, subclasses, and their connections are clearly shown with properties.

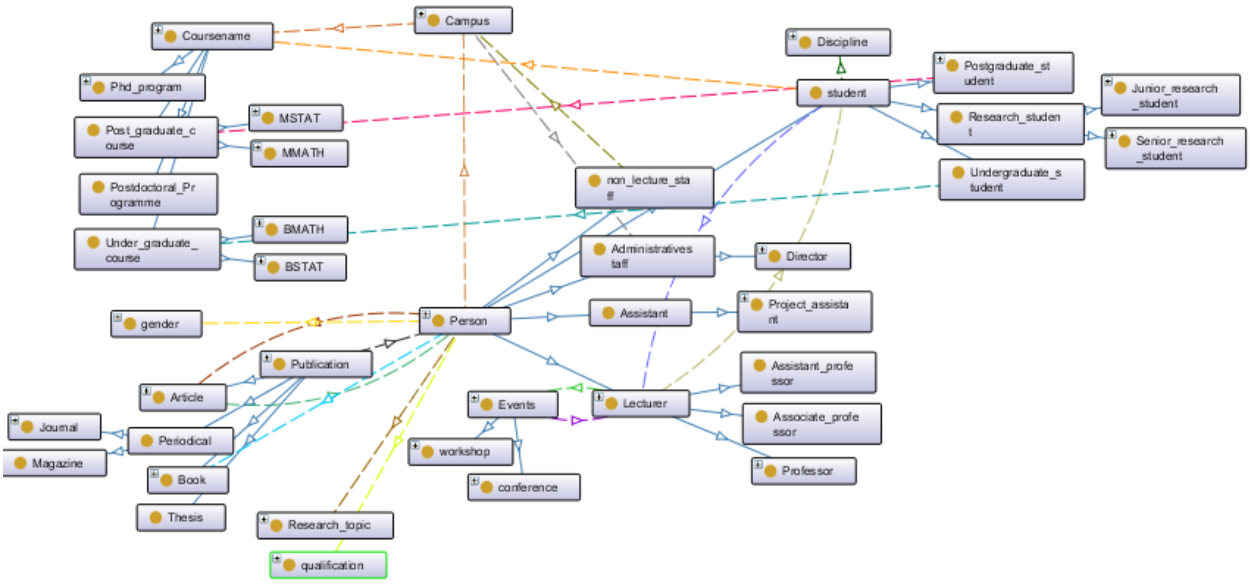

Fig 4. Ontology Graf.

As part of the campus, administrative staff, Lecturer, Students, and non-Lecturer staff are under class Person. Students of all disciplines are classified based on their level of study. And the students are categorized based on their course name. Person, whether it might be staff, non-staff or students can write publications. In this relationship, subclasses for publications are articles, books, thesis, and periodicals.

On the other hand, there is subclass called Events which is classified to conferences and workshops is conducted by lectures. Project, Research area, and research interest are the remaining classes should not be missed to include all the projects and relation between lectures and students in their level of study.

Adding object and data attributes to our Ontology architecture is the next stage. This will aid us in obtaining more exact data or search results. Classes are linked together by object properties, and this connection will be represented in the ontology as a triple. End elements that specify particular information about the class are known as data properties. To generate object and data attributes, Protégé provides a user-friendly interface.

3.3 Object Properties

The fundamental notion of semantic data is that the object attributes are used to link classes together to generate triples. The below figure.4 shows the 25 object properties
which link individuals(data) in this ontology.

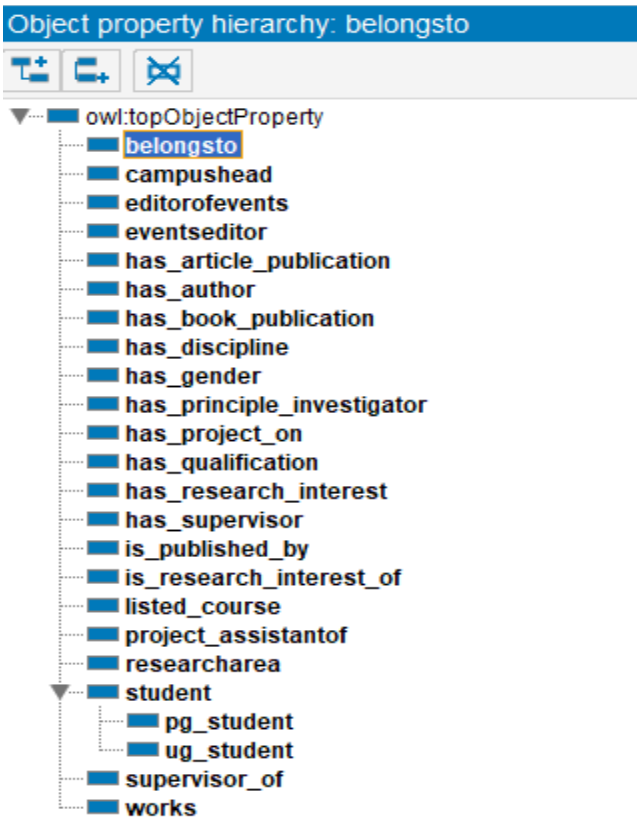

Fig 5. Object Properties.

Every Object Property is defined based on the domain, range and property (relation), all together called triple. The object attributes' domain will change based on the class relationship to be formed. For example, belongsto is defined as a relation between campus and person which says Person belongs to campus (triple).

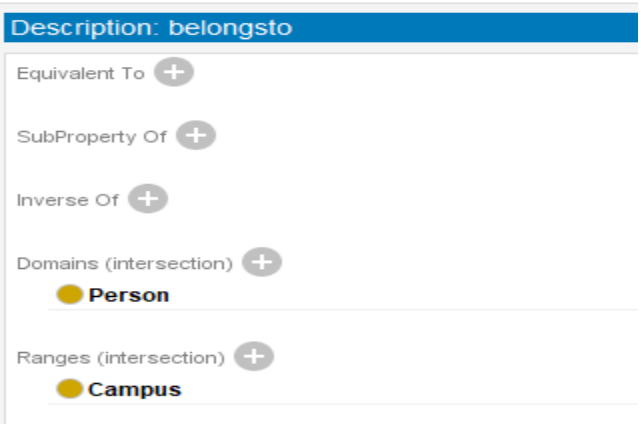

Fig 6. Description of object property belongsto

There are also few inverse object properties like has_supervisor and is_supervisor _of are inverse to each other for student and lecturer. Likewise, there are 4 inverse object preperties defined.

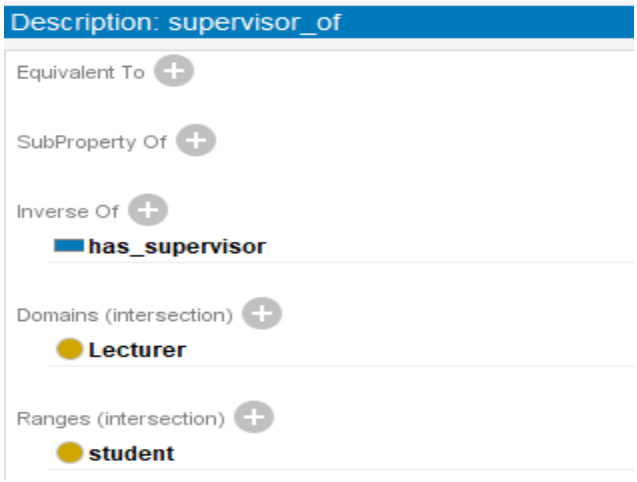

Fig 7. Showing Inverse Object property - *Supervisor_of* is inverse of *has_supervisor* object property between Lecturer and student

Functional, Inverted Functional, Symmetric, Asymmetric, Reflexive, Irreflexive, and Transitive are object traits that characterize the sort of relationship among domain and extent in a connection. For example, In the Person *belongs to* campus relates too many to one relation which says that the relation is functional. This explains Person can only have one campus but two or more can belong to same sampus.

3.4 Data Properties

A collection of data attributes is included in the ontology that allows and restricts what type of values to be connected to class variables. There are 20 data properties that specify things like a person's first and last name, city, date of birth, ID – which is used to identify everyone on campus individually – and awards, among other things.

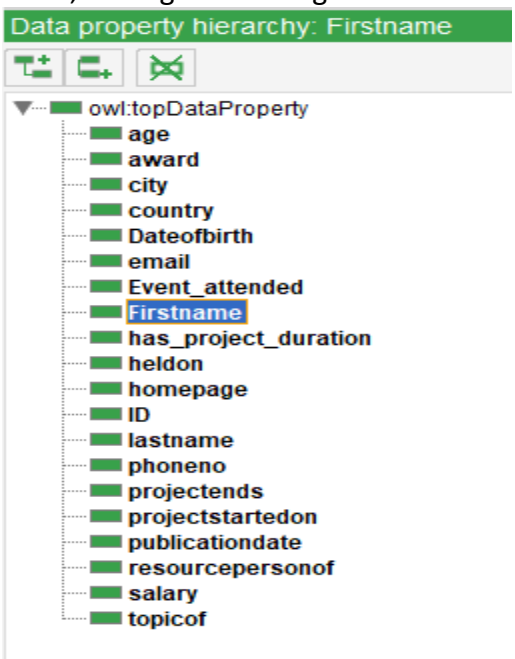

Fig 7. Data Property hierarchy

Data types:
In OWL, most of the datatypes are defined in the XML (xsd) namespace. Every data property can be set a data type while creating in protégé. Below are few commonly used datatypes:
Numeric – xsd: integer,xsd:float.
String – xsd: string
Boolean – xsd: Boolean
URI – xsd: anyUri
Time – xsd: dateTime

All of the information on the Ontology will be identified using the data feature. For identifying Persons (students, lecturers, non-staff) who have first name, last name, ID, Email-id, award, city, and some properties are set to string type based on individuals' data type. Age, salary, and project duration are integer type. The remaining data properties date of birth, project started on, and publication date are type date.

3.5 Individuals

Individuals are the occurrences of the interactions with the data and subject attributes. Memberships of a class are individuals who have similar features. For instance, name of student is an individual of the class student with all the data and object properties of the Ontology built for the application.

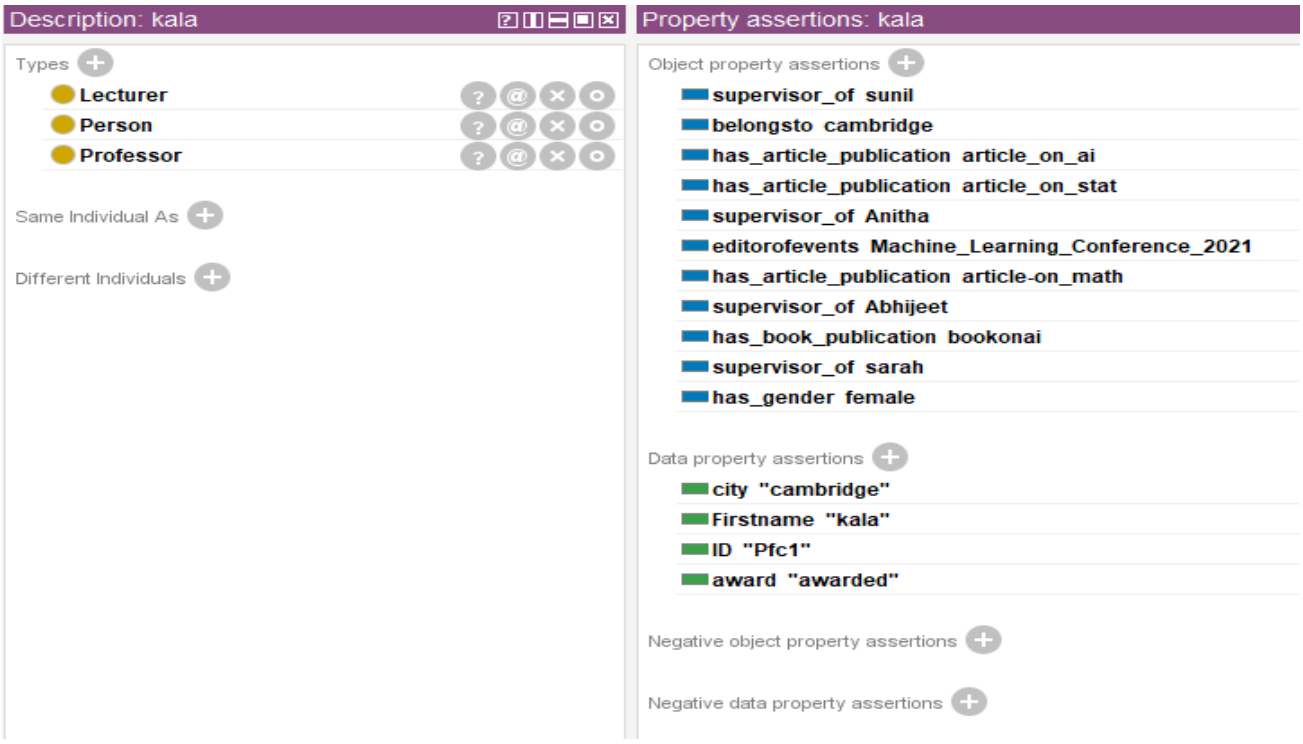

Fig 8. Example for individual

The above figure gives the description of individual named *kala* with all the object properties and data properties linked.

Reasoner

To check whether the ontology designed is functioning properly or not; I have used Hermit reasoner which is integrated in Protégé. Once I add few individuals with all the properties, I have frequently checked working of ontology and rectified errors at the same time while designing my ontology.

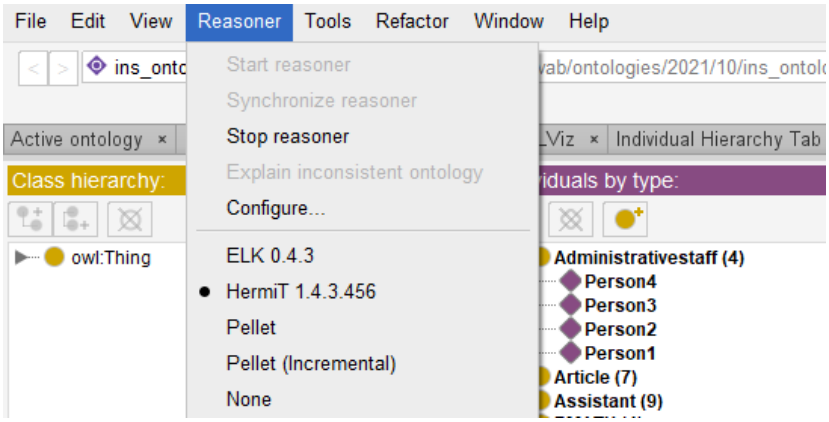

Fig 9. Working Hermit Reasoner.

4.Implementation

I choose C# programming language, using DotNet framework to develop web application. I have complied and executed all my code in Visual Studio. To check working of ontology in local server, I have used Apache Jena Fuseki to upload my ontology and execute the SPARQL queries in local server.

4.1 Fuseki server

Firstly, Apache Jena Fuseki is downloaded, installed in PC. Connection to the local server is established using port number.

After installing Fuseki, to open and test query on local server - *http://localhost:portnumber* has to be entered in search engine. The *port number* in my case is 3030 (fig. 9).

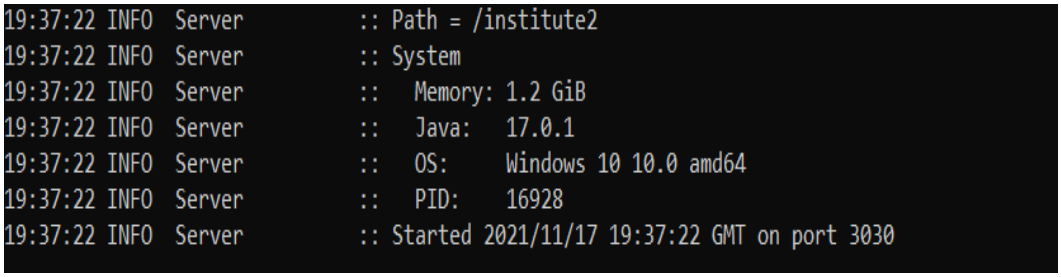

Fig 10. Fuseki windows batch file execution showing port number for local server.

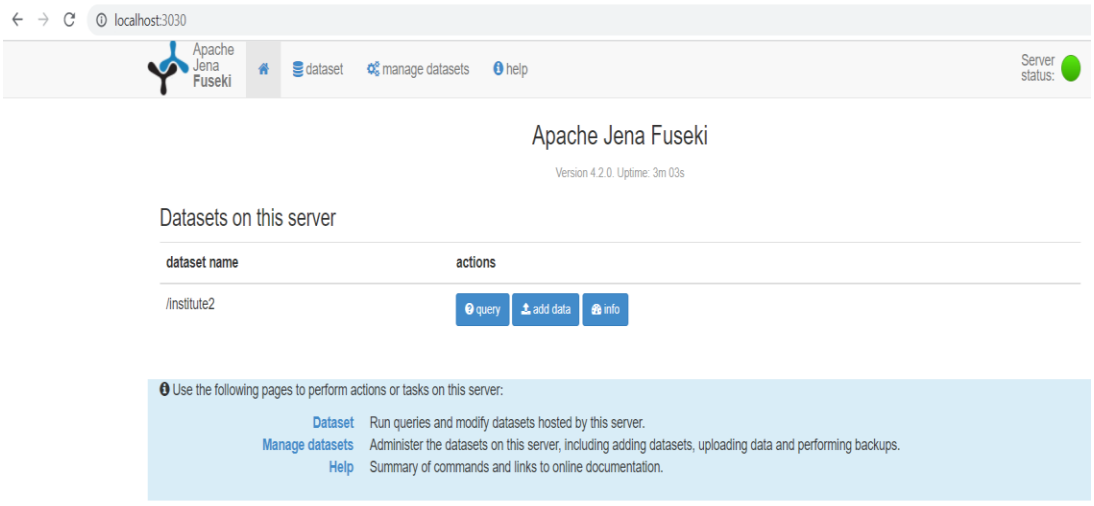

Fig 11. Apache Jena Fuseki server Interface.

After uploading the ontology in the Fuseki server, SPARQL queries can be tested online on local server. It's possible to run it as an operating software platform (Jena.apache.org, 2019).

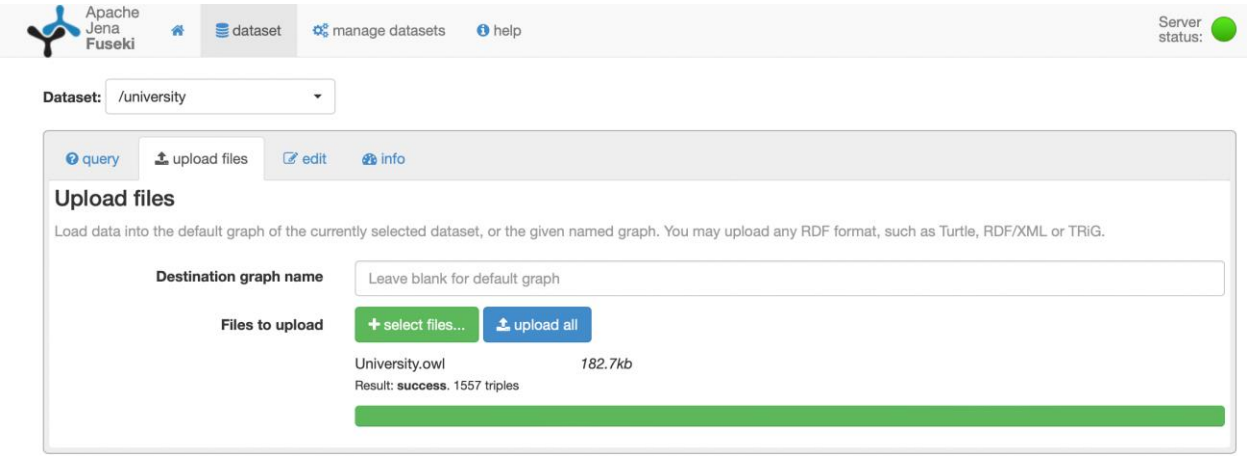

Fig 12. successful uploading of ontology to Fuseki Server

4.2 Project workflow

This model was developed in C#, and its foundation is the ASP.NET Core framework and also the dotNetRDF programming library. Using the release of.NET Core 3.0, programs created with this framework may now be packaged as 'self-contained' apps, which means that all of the software's dependencies are included in the application. It will be easier to offer them as a stand-alone application.

This employs the model-view-controller (MVC) application architecture, which helps developers to decouple the application's functionality from the user interface (the view) as well as the database models itself. Inside this ARU project folder, this expresses itself as three distinct folders:

A **model**— comprises a collection of view models capable of transporting data between the view and the actuators.

A **view** — includes elements of the user interface. The developer has deployed MVC in this project in the following manner.

Controllers — include all of the logic responsible for processing client request and returning the answers to the view.

Structure of Project:

The current task and the querying project are two nested components in this project – ARU and ARU.source.

The ASP.NET Core project, which includes all graphical interface and information processing, is located in the Main working directory, whereas the SPARQL connections and server interaction logic is located in the ARU.SOurce folder. Because dotNetRDF is now incompatible with the.NET Core technologies, including ASP.NET Core, this has been done. Because both dotNetRDF and ASP.NET Core are consistent with the.NET Standard, the inquiry project compiles to a stirring up library that links to the ARU project at build time. Because this library uses.NET Basic as its target platform, the primary job may use the college as a go-between for the SPARQL queries it wishes to make. As a result, there is no compatibility issue.

4.2.1 ARU project folder

The ASP.NET Core project is included, as well as all of the graphical interface and input handling. An overview of the folder structure is shown in figure

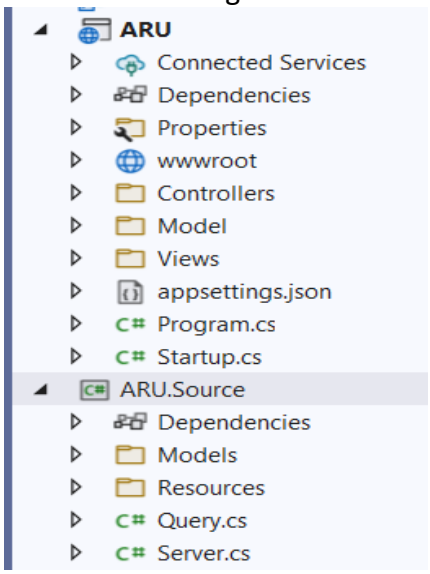

Fig 13. Overview of project folder

- javascript, Bootstrap, Cascading Style Sheets, and jQuery are all located in the **root folder**.
- HTTP GET and POST events are used to store ongoing conversation logic in **controllers**.
- **Models** – these are the view models that are used to transmit data from one view to another.
- **A view** — includes the front-end interface design.

4.2.1.1 Controller

The homecontroller.cs, which is in the Controllers directory, is worth mentioning. This object serves as a conduit between the perspective and the controller, allowing the view to receive information from the user and indeed the gamepad to act on it and respond appropriately.
This class contains GET and POST functions for each view in the home subdirectory. Because a viewpoint could only have one of each sort of response function - GET or POST — some functions, such as Search, don't have a matching view. If a perspective has many forms, each of which is meant to provide data to the controller, each form should have its own set of response procedures. The usual Index POST procedure is used in one form, while a custom search POST function is used in the other.

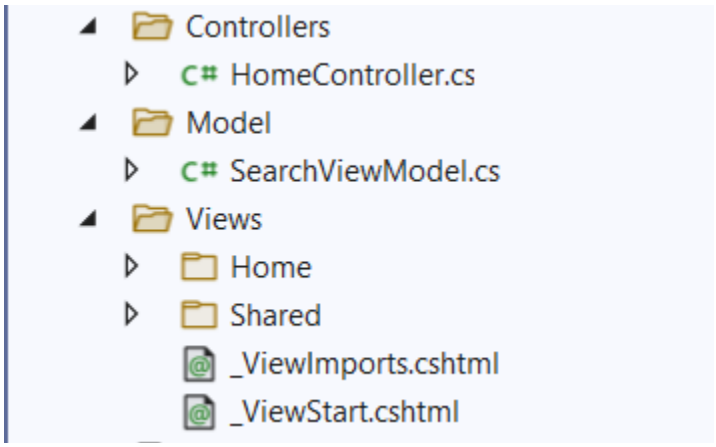

Fig 14. View of controller folder

The ARU.source is used by the Populatesearch() function to execute specific searches that complete the Index view's filter drop-down fields. A new query to the SPARQL API is required to populate each filter. Because each of the drop-down selections can contain the names (labels) of people or classes from the ontology, ARU.Source sends one of two possible requests to the endpoint, which explains why this function is so verbose. I have included 14 filters in the application (fig. 17), but I haven't fully tested each one with a query. Similar filters and the SPARQL Query have both been tested. For example, I built a SPARQL query for the Cambridge campus but not for the others because they are all comparable.

The Index POST method handles the user's search requests. It begins by evaluating each of the categories and determining which have been filled in, after which it builds a SearchModel containing the identities of the classes and persons selected from the various drop-down menus. This data is then passed on to ARU.Source, which executes the query and returns the results. The POST method then collects these results and uses the view model to return them to the view — and thus the user.

The *interface's* textual search is handled by the Search POST method. This stored procedure task is rather straightforward: it takes the user's input, cleans it, and feeds it to ARU.Source, which then runs a query that produces a collection of search results. After that, the results are delivered to the view model's view.

4.2.2 ARU.Source Project

The bulk of the SPARQL-related functionality is included in the main,uni project. It includes:

1. 'Models' - The data models used to transmit values between main and main are stored here.

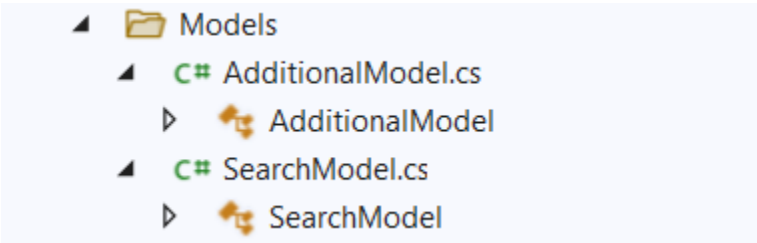

Fig 15. Model folder in ARU.Source

2. ‘Resources’ - includes a copy of the ontology in its entirety.

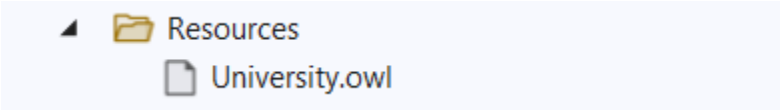

Fig 16. Snippet of copy of ontology in project folder

4.2.2.1. Server connection

The code for connecting to and disconnecting from the Fuseki server on which this application relies is all included in server..cs class. The connection function establishes the connection to the existing server using a hard-coded URI, then checks to verify if the server state is ready before returning the result to the caller class. The Disconnect class, on the other hand, safely disconnects from the server when the program is closed.

4.2.2.2 Query

The majority of the SPARQL logic is contained in the Query.cs class. All queries that can communicate with the SPARQL interface are contained in this class. Most of these searches are mostly hard-coded, implying that they are largely pre-written, while others are dynamically created, implying that they are mainly unwritten by default.

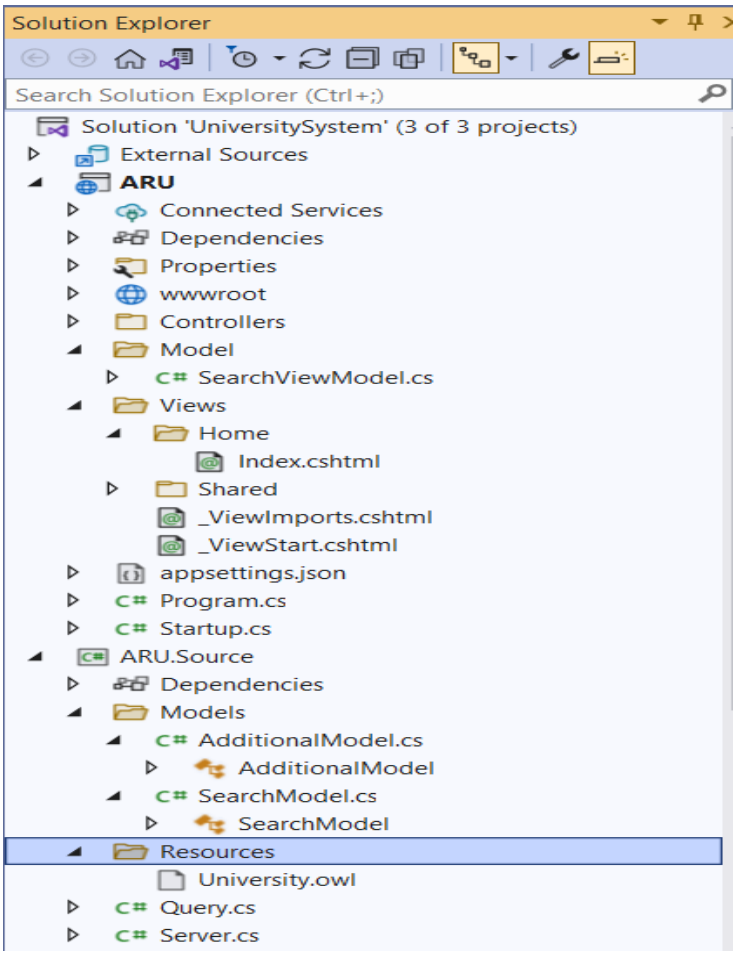

Fig 17. Complete Project folder in Visual Studio

4.3 Project Execution

To view the web application in Visual Studio once the coding part is completed; build the project and view the user interface by connecting with the local host with IIS Express button.
The successful compilation of code with no errors connects to local host and displays the user interface like shown in the below figure.

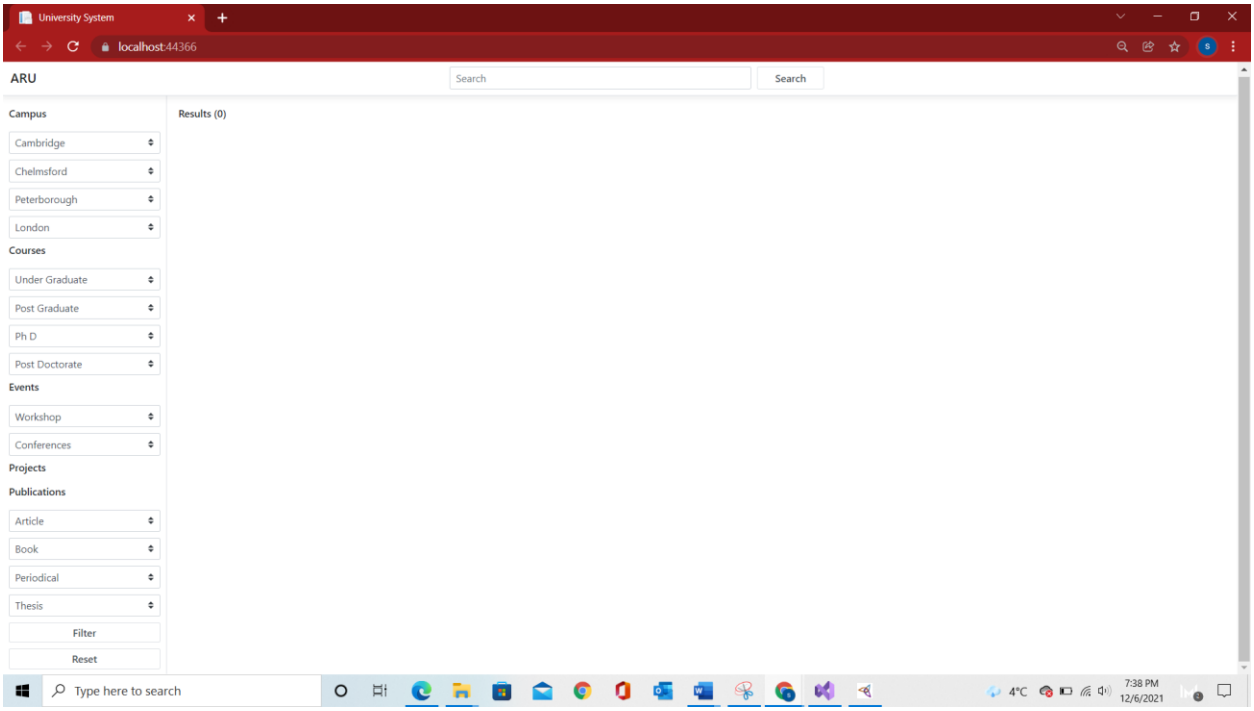

Fig 18. Output after building the project – User Interface

This is the user interface I have created which has all the filters in the left margin of page and search box with search button in the center top of the page. Filters include campus as header to Cambridge, Chelmsford, Peterborough, and London Campuses. Like wise the other filters include courses with course names, events with conference and workshops, publications, and Projects. Results can be viewed in the center of the page with all the filters selected. For now, fuseki server is connected but I am not able to retrieve all the SPARQL query results into my user interface.

Results of one query is shown in below figure.

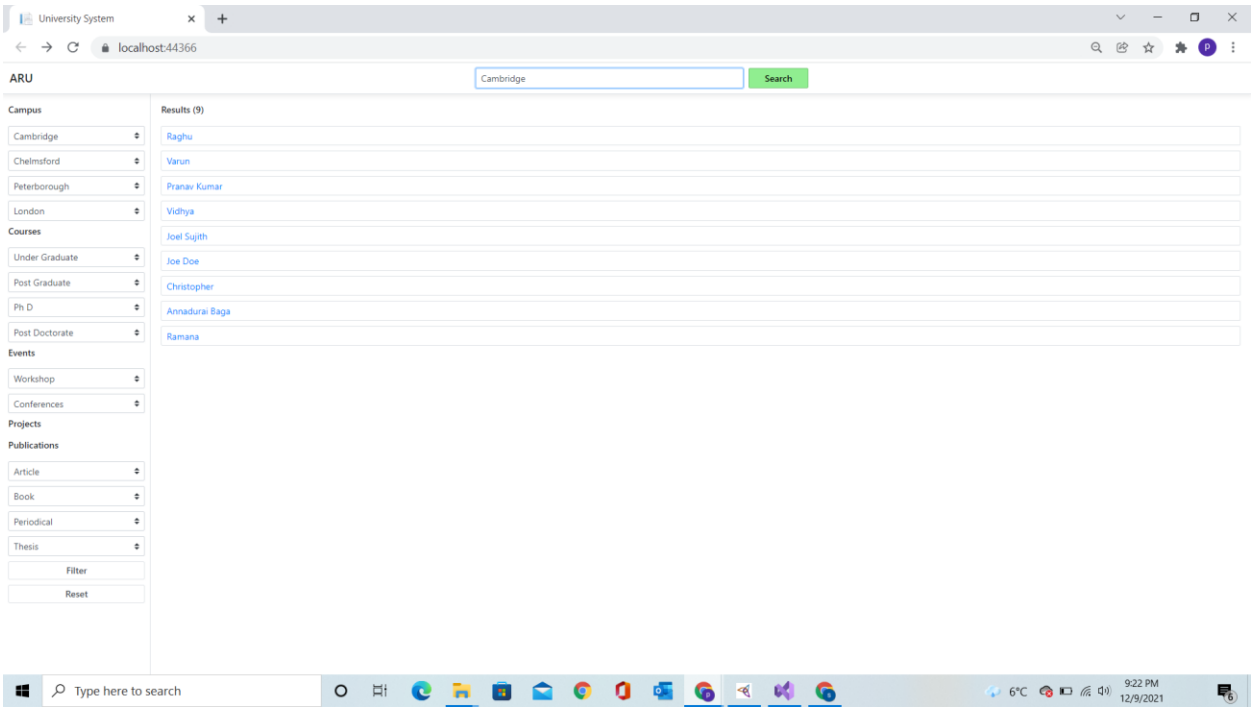

Fig 19. Results of one query in the interface

5. Evaluation

This part examines and assesses the results of each query sent to the SPARQL endpoint. There are seven inquiries in all, divided into six functions, each with its own purpose.

5.1 SPARQL Query Examples

SPARQL query for filters

The University user interface with two views – the browser view to provide search functionality and product view to show the item selected from the search results. Left part of the browser view contains filters with course names to filter all the persons based on course name.

SPARQL query:

```
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 table: <http://www.semanticweb.org/devab/ontologies/2021/10/ins_ontology-50#>
SELECT DISTINCT ?object ?subject
        WHERE { ?subject rdfs:subClassOf ?object }
ORDERBY(?object)
```

Fig 20. Query to filter Distinct classes to use them as filter in UI.

object	subject
Coursename	Postdoctoral_Programme
Coursename	Under_graduate_course
Coursename	Phd_program
Coursename	Post_graduate_course
Events	workshop
Events	conference
Lecturer	Professor
Lecturer	Assistant_professor
Lecturer	Associate_professor
Periodical	Magazine
Periodical	Journal
Person	student
Person	Administrativestaff
Person	Lecturer
Person	Assistant
Person	non_lecture_staff
Post_graduate_course	MSTAT
Post_graduate_course	MMATH
Publication	Article
Publication	Periodical
student	Research_student
student	Postgraduate_student
student	Undergraduate_student

Fig 21. Results of filters used in the interface.

The filters on the left include campus, courses based on level of study – undergraduate courses, post graduate courses, PhD courses and post-doctoral courses. By putting the student's or

professor's name into the search bar in the view's center and choosing search to query the terminal, the user can find a specific student or professor. When you click the reset button, the website will reload, and any specified filters will be reset. Publications and projects are the other filters that user can search person details about.

Query to count the class preview

This simple query is to retrieve how many instances are present for each class. This will answer how many persons are there in the campus? what are the courses in campus for course name? how many articles, books are published? how many junior,senior research scientists are present in the campus? who are the project supervisors? what are the events (workshops, conferences) going on or conducted? Likewise, the questions related to the classes are answered.

SPARQL query:

```
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 table:<http://www.semanticweb.org/devab/ontologies/2021/10/ins_ontology-50#>
# Count the occurences of all RDF classes
#
select ?class (str(count(?class)) as ?count)
where {
  # query the default graph, or uncomment and insert graph name
  ##graph <http://dummy.org/graph> {
    ?x a ?class
  ##}
}
group by ?class
order by (?class)
```

Fig 22. Query to count the class preview

class	count
Campus	"4"
Coursename	"15"
Director	"4"
Discipline	"7"
Events	"12"
Journal	"3"
Junior_research_student	"9"
Lecturer	"8"
MMATH	"2"
MSTAT	"2"
Periodical	"3"
Person	"17"
Phd_program	"3"
Post_graduate_course	"4"
Postgraduate_student	"9"
Professor	"9"
Project	"13"

Fig 23 . Result for class preview.

This query is helpful for student, researchers, recruiters answering all the questions mentioned above. From the results, few of answers are 4 campus, 13 projects, 12 events, 7 articles on my ontology.

Query for presenting data

This following query (Figure 14) is intended to return all of the person personalities, and that it is the question that, by default, clogs up the Browse view with all of them. It displays person's first and last names, student ID, email address, and date of birth, location and contact details of the person searched.

SPARQL query:

```
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 table: <http://www.semanticweb.org/devab/ontologies/2021/10/ins_ontology-50#>
SELECT DISTINCT ?Firstname ?lastname ?ID ?email ?Dateofbirth ?city ?phoneno
WHERE {
?Person a table:campus.
OPTIONAL { ?Person table:Firstname ?Firstname; }
OPTIONAL { ?Person table:lastname ?lastname; }
OPTIONAL { ?Person table:ID ?ID; }
OPTIONAL { ?Person table:email ?email; }
OPTIONAL { ?Person table:Dateofbirth ?Dateofbirth; }
OPTIONAL { ?Person table:city ?city; }
OPTIONAL { ?Person table:phoneno ?phoneno.}
}
ORDER BY(?ID)
```

Firstname	lastname	ID	email	Dateofbirth	city	phoneno

Fig 24. Query to display details of person

Query individuals with conditions

This query enables user to search based on preferences in the browser view. It is designed to filter all the person based on publications and location of study. This query returns the person's name, publication name and campus of study.

SPARQL query:

```
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 table: <http://www.semanticweb.org/devab/ontologies/2021/10/ins_ontology-50#>
SELECT ?Person ?Publication ?Campus
WHERE {
        ?Person table:belongsto ?Campus.
        ?Person table:has_article_publication ?Publication.
        FILTER(regex(str(?Campus), 'cambridge'))
}
```

Fig 25. Query using *filter* function to filter persons belong to Cambridge with conditions

This is simple query for implementation of *filter* function and retrieving persons on specified location. Individuals are filtered by person with Cambridge as location of study condition. The results can be seen in the following figure.

Person	Publication	campus
kala	article-on_math	cambridge
kala	article_on_ai	cambridge
kala	article_on_stat	cambridge
vinay	article-on_math	cambridge
rakesh	article_on_ai	cambridge
John	article_on_ai	cambridge
Anitha	article-on_math	cambridge
kashvi	article_on_ai	cambridge
vinay	article_on_ai	cambridge

Fig 26. Results for the Query with filter function

Query individuals with selected filter:

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX table:<http://www.semanticweb.org/devab/ontologies/2021/Institute#>
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#>
SELECT DISTINCT ?Person  ?campus  ?Events
WHERE {
            ?Person table:belongsto ?campus.
            ?Person table:has_article_publication ?Article.
            ?Person table:has_book_publication ?Book.
            ?Lecturer table:editorofevents ?Events.
            FILTER(regex(str(?campus),'cambridge'))
            FILTER(regex(str(?Article),'article_on_ai') || regex(str(?Article),'article-on_math') || regex(str(?Article),'article_on_stat'))
            FILTER(regex(str(?Book),'bookonai') ||  regex(str(?Book),'bookonmath') || regex(str(?Book),'bookonstat'))
}
ORDER BY ASC(?Person)
```

Fig 27. Query include set of filter conditions.

This query is intended to return all of the people who meet a broad set of restricted criteria, depending on the filters specified in the browser window.
Set of conditions include: to retrieve person who have published both articles and books on set of subjects, also being the editor of events, and from Cambridge.
This is helpful for recruiters, students, professors who are particularly looking for person working in Cambridge on specified field of study.
Following these conditions *kala* and *john* are two professors as results for this query.

Person	campus	Events
John	cambridge	Machine_Learning_Conference_2021
kala	cambridge	Machine_Learning_Conference_2021

Fig 28. Results for the above query with multiple filters

Results of individuals who have published both articles and books on Artificial Intelligence, Mathematics, and statistics. Also being the editor for events in Cambridge.

Query for research student being assistant to lecturer

This Query is to retrieve information of research student who are also assistant project student for Lecturer in specified campus.

I have included few conditions for research student being assistant:

1. Should belong to one campus
2. Should have same project supervisor
3. Research student should have article or journal publication

SPARQL query:

```
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 table: <http://www.semanticweb.org/devab/ontologies/2021/10/ins_ontology-50#>
SELECT ?Person ?Publication ?Campus
        WHERE {
                SELECT  ?Research_student ?Lecturer as ?Research_lecturer ?Article
                        WHERE {
                                ?Person a table:?Research_student.
                                ?Person table:belongsto ?Campus.
                                OPTIONAL{
                                        ?person table:has_article_publication ?Article.
                                        ?person table:has_journal_publication ?Journal.
                                        ?Person table:has_supervisor ?Lecturer.
                                        FILTER(regex(str(?Campus),'cambridge'))}
                                }
                ?Person a table:?Assistant.
                ?Person table:belongsto ?Campus.
                ?Person table:has_project_on ?Project.
                ?Person table:has_supervisor ?Lecturer.
                ?Project_assistant table:project_assistantof ?Lecturer.
                FILTER(regex(str(?Campus),'cambridge'))
                FILTER(regex(str(?Lecturer=?Research_lecturer)
}
```

Fig 29. Query for student being assistant to lecturer

Person	Publication	campus
rakesh	article_on_ai	cambridge
John	article_on_ai	cambridge
Anitha	article-on_math	cambridge
kashvi	article_on_ai	cambridge
vinay	article_on_ai	cambridge

Fig 30. Results of research student who are assistants.

Results show that there are 5 students who are assistants and research students both at a time in Cambridge.

6. Critical Reflection

The closest analogy would be the mechanisms used to discover professors or courses on the ARU webpage while the system was not designed to be patterned after any specific program.
However, there are few ontologies like university ontology those are made public. Course ontology and university ontology are very similar, in some way or other later includes parts of course ontology. Relationships and comparisons can be made to improve ontology functionality and draw conclusions.

This information needs to be updated from time to time which I have not implemented now in my interface. In future it can be done with the information provided in this report.

There are some privacy concerns regarding information published. It can be serious problem if there is any data leak. And also, there in a need for data protection where only authorized person can access the details of persons. For example, it is not necessary to publicly display all details of persons in campus. All measures need to be taken before building fully model and continuous monitoring is needed.

I have found that this application can be used from anywhere as long as the device has a browser, and it works on all devices: There are numerous advantages of becoming a web-based application. Because the dotNetRDF library does not support the ASP.NET Core development framework, the author had to work around it. The author would avoid using the platform totally in the future due to the lack of alternative RDF support in the.NET environment.

Only a few university libraries that would be found in a typical university system are supported by the ontology. During its construction, I found that adding persons to the ontology — in Protege — takes a long time owing to the fragmented workflow it employs. Adding each student's information to Excel and loading it into Protégé took a long time. This type of procedure would be inconvenient for non-technical field workers; hence any future version of the system should have a way to automate individual creation, either within the program or by other means.

Some of the similar ontology available online:
(Shubhi Shrivastava et al., 2018) has highlighted how university ontology works by employing ideas and sub conceptions, as well as defining their features and relationships between them. It has also provided context for the notions. Testing of the ontology's functionality is done by using the FACT++ reasoner depending on a DL query.

Using University Ontology, (Navin Malviya et al. 2011) defined Information University. They have thoroughly defined many linkages in various topics found in universities. They used Protégé to create university ontology, employing Rajiv Gandhi School of Bhopal as an example, and assessed the Query recovery procedure visualization view and network view.

I feel that constructing a fully functional online university system employing semantic data capabilities would be advantageous. However, its present solution isn't 'fully-featured' enough then to allow production usage, owing in part to the site's scope constraints in this situation

7. Conclusion

To summarize, the creation of the student searching application met the Concept and Aim sections' objective of producing a functioning application employing semantic and web capabilities.

Design and implementation of ontology is explained clearly in respective sections. The SPARQL queries used with the application's endpoints were thoroughly explained in the Assessment The queries were checked for accuracy on the Fuseki server. Talking about user interface, I have created and executed successfully but I was not able to retrieve results in the user interface using Visual Studio.

Finally, in the critical reasoning part, I have compared online applications and concluded that there is a lot of room for programming improvement. In future editions of the app, more comprehensive information like as credits, tests, and placements might be added on the Ontology. This will allow for the creation of more complicated queries, resulting in additional data search choices for the user.

References

Giuseppe Lugano, December 15, 200. *Semantic Web Technologies and the FOAF Project*. [online] Available at: <https://www.researchgate.net/profile/Giuseppe-Lugano/publication/265188150_Semantic_Web_Technologies_and_the_FOAF_Project/links/54e485960cf2dbf60696c7fd/Semantic-Web-Technologies-and-the-FOAF-Project.pdf>

Naveen Malviya, Nishchol Mishra, Santosh Sahu, 2011. *Developing University Ontology using protégé OWL Tool: Process and Reasoning*, International Journal of Scientific & Engineering Research Volume 2, Issue 9, September-2011 1 ISSN 2229-5518. [online] Available at: < https://www.ijser.org/researchpaper/Developing_University_Ontology.pdf>

Protégé.stanford.edu,2019. *Protégé.* [online] Available at:https://protégé.stanford.edu

Shubhi Shrivastava, Iti Mathur, and Nisheeth Joshi, 2018. *AN ONTOLOGY DEVELOPMENT FOR UNIVERSITY*. [online] Available at: < https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8933803>

Tauberer, J., 2006.*What is RDF*. [online] Available at: <https://www.xml.com/pub/a/2001/01/24/rdf.html>

Thomas B. Passin, 2004. *Explorers Guide to the Semantic Web.* Chapter 1. Available at:< https://www.devx.com/assets/download/11209.pdf>