FIT 3162 Final Project Report

MCS17 Ontology

Group Members Name:
Tan Jin En
Yap Rui Yi
Seow Zheng Hao

Word count = 6775

===

Table Of Contents

===

==

Introduction

==

The project topic we have been working on since the beginning of this year is about medical ontology. Ontology is a formal system for modelling concepts and their relationships; it is a way of showing or specifying the entities (concepts, categories, properties, and relations) of a domain together with a specified range. These entities are then grouped as taxonomy, which is a graph structure that describes the hierarchical relationship of a group of things. In our case, the domain and range we are interested in concerns heart disease and symptoms of various heart diseases respectively. So we have to create an ontology that focuses on providing a structured and formal representation of concepts, relationships, and properties related to heart disease .The overall goal of this project given to us by Dr Nailah is to create an application that describes how medical ontology can be used to describe the concepts of medical terminologies and the relation between them, and also demonstrate and highlight meaningful relationships between data from other potential complex diseases (e.g. diabetes) the patient has in relation to heart disease. By doing this, we can justify how clinicians are able to infer a lot more meaning from the relationship between the data, and consequently convince clinicians why and how ontology should be used in healthcare or in expert systems.

However, we realised that this goal is not easily achieved due to its inherent ambiguity. Initially, our aim is to create a non-functional prototype with a user interface that can visualise how the ontology database can be used to represent meaningful relationships so as to aid clinicians, patients and scientists in understanding heart disease progression. But this was later rejected by us as we realised that it was not technical enough. This causes us to spend an additional 4 more weeks to come up with another software deliverable. After spending much time on group meetings, research, and checking for project feasibility, we decided to use the protege application to build our ontology database, then integrate the ontology database together with the UI through the use of OWLAPI, so that we can query our ontology database through a user interface in a mobile application and then output the results of the query by displaying it inside our mobile application to showcase a proof of concept of ontology. However, we later revised our aim to focus solely on developing a desktop application, rather than a mobile one, in the last few weeks. This shift occurred because we recognized that Android Studio lacked the necessary dependencies for importing OWLAPI. Furthermore, we realised that our earlier goal was still lacking a comprehensive user interface. Hence, Our final aim is to develop a desktop application capable of taking a symptom list as input and providing a list of potential diseases as output. This information will be presented in a display featuring a visualisation of a heart, along with severity ratings and descriptions for each disease resulting from the query. The rest of the report consists of details of how we effectively accomplished this goal.

==
Project Background
==

1) Background

Ontology is a formal system for modelling concepts and their relationships. It is a way of showing or specifying the concepts, categories, properties, and relations of a subject area or a domain. Ontologies are used to limit complexity and organise data into information and knowledge in different academic disciplines or fields. Ontologies may also form a system of categories that encompasses classification of all entities. An ontology may include a taxonomy, which is a graph structure that describes the hierarchical relationship of a group of things. Medical ontology, on the other hand, is a field of study that focuses on the representation of medical knowledge in a structured and standardised way. It is essential as it is used to support medical terminologies and coding systems as well as to model the medical knowledge in clinical practice in real life. Hence, the aim of this project is to create a software which utilises an ontology based design database in order for clinicians to visualise heart disease related information, such as symptoms, treatment, or gauge the severity of the disease to prove that ontology can be used to enable the sharing of medical knowledge and improve how clinicians gain access to medical knowledge. For this project, our main users will be clinicians in the medical field. The project involves a database with an ontology based methodology with user input to process data and draw connections between sets of data. The data will then be visualised using a user interface for the users.The main data processing component is an ontology based database. The workflow of the software involves an input of data, a database model of an ontology for heart disease, and a user interface to display the relationship between input data and identify flawed data.The 3 components work together to produce a software that takes in user input and queries information from the database. The database uses an ontology based methodology to filter the data and categorise them. The output result from the database aims to help the user identify connections between data and additional information with a visualisation through a user interface. The software aims to allow novice level clinicians that do not yet have the necessary knowledge from experience compared to veteran clinicians to have a tool that can help in the querying of medical data.

2) Updated literature review

A. Background

Ontology is a knowledge representation methodology that groups related data together and is used to represent how some piece of information is related to another. A more formal definition of an ontology is the representation of knowledge with rules and automated reasoning to detect inconsistencies within a data pool. In the medical field, there is plenty of data that is used on a day to day basis. One such data is Electronic health records (EHR). EHR is a way that clinicians record down their analysis of a patient as a form of data entry. It aims to paint a story of the patient's condition that other clinicians can read and understand.

The large amount of data that is needed to be processed everyday makes it difficult for clinicians to be always updated with the latest terminologies and understandings for diseases. This creates the need for a database hub for clinicians to easily check for the definitions for diseases, especially when trying to simplify and explain to patients during normal checkups, where it is most useful to have a tool to easily retrieve reliable information on diseases. Data from patient records is one such example of a need for adequate methodology to allow the extraction of information from clinical medical text records, the input of data into a system, and the categorisation of the patient’s data gathered by clinicians. While an existing data categorization method such as taxonomies may be able to filter out the data by keywords, it lacks the ability to associate data based on the patient’s diagnosis like how veteran clinicians can. This is where Ontology comes in. It makes use of defined terminologies that describe the medical concepts and taxonomies that categorise data to find relations between data, thus, enabling the sharing of medical knowledge. Ontologies not just have to find data that can be associated together, but also data that might not necessarily be in the same type, i.e, finding disease information based on the body part in question. The system needs to know which medical terms indicate a medical condition in order to accurately match data for clinicians to use. These relations within data are usually only realised by veteran clinicians, but ontologies aim to expand this ability to find relevant knowledge elements from certain diagnosis of a patient, to other clinicians in the medical field.

B. Rationale

Ontologies are important because they provide a shared and common understanding of a domain that can be communicated between people and applications. They are used to support knowledge sharing and reuse, data integration, and semantic interoperability. Ontologies can also be used to support reasoning over data and to enable intelligent systems to make more informed decisions. In the medical field, when a medical terminology is used, it does not provide the other clinicians with any nature of the term, leading to misunderstandings. A usage of ontology in the medical field with EHR is a query for EHR records that include the phrase diabetes. This would return a list of EHR that match the query search. However, this is not a simple search for the appearance of the phrase “diabetes” in EHRs in the system. In the medical field, there are many different terms that clinicians use to provide additional details for that particular sickness. Terms like diabetes melitus, type 2 diabetes or even diagnosis results with content such as "elevated A1C or saxagliptin 5 MG”, that doesn't include the phrase diabetes should be under the general concept of diabetes. This simple example illustrates that a certain level of artificial intelligence is needed to find such data, understand and create the link between the data sets, and categorise them.

C. Related research

1. An overview of ontologies and data resources in medical domains

This journal provides an overview on the progress for the research in the field regarding the use of ontology in medical domains. It provides an overview of the required parts for ontology usage for data processing in the medical field and the progress of the research. The journal includes an analysis of a list of related work that others have completed that help with the ontology development. It aimed to analyse the progress of the research and explain their shortcomings. This journal provided a summary for research articles related to ontology. Although this journal provided an overview of many sources and their shortcomings, it did not provide an in-depth explanation of the individual research papers.

2. Ontologies in Medicine

In the first chapter of the book “Ontologies in Medicine” by Pisanelli (2004) ,the authors highlight the need for ontologies to standardise medical terminologies and increase the effectiveness of information retrieval of guidelines that clinicians use. A library architecture is proposed to facilitate better retrieval of information on medical procedures. It uses 5 major categorizations to group the procedures used to simplify the process. This differs from other categorization methods found in other researches as the categorizations are clearly defined beforehand. The aim of this research is to provide justification for the usefulness and broadness of ontology usage in the medical field. This research covers ontology usage on many aspects of medicine. However, due to the complexity of the proposed system, it requires an entire systematic change in operations within the medical field, which has a low possibility of occurring and will take much more research and evidence to change the current medical field operations.

3. Ontology-based framework for electronic health records interoperability

This research journal discusses the need and difficulty to achieve computable semantic interoperability for the data within Electronic Health Records used in the medical field to hold patient’s data. It discusses the latest research on the standardisation of standards for data interpolation and an interoperable EHR framework using ontology. The proposed framework for EHR semantic interoperability references a research by Bernd Blobel that aimed to design and implement a secure and interoperable health information system. This framework makes use of an ontology mapping process to create domains from other research on the use of ontologies but for enterprise knowledge management instead. This is a universal ontology data categorization that aims to break down data into smaller sub categories to reduce the complexities of domain. This research falls short in its practicality as the EHR interoperability standards and other related work used in this proposed framework are incomplete in terms of functionality.

4. Clinical Communication Ontology for Medical Errors

This research highlights the need for ontology to reduce communication errors within the medical field as clinical communication failures caused 60% of sentinel events reported by the Joint Commission on Accreditation of Healthcare Organizations. The Protégé-OWL editor was used to build the ontology and aims to satisfy Cimino's desire for controlled medical vocabularies to provide a standardisation of medical communications. The research primarily covered the experience of the researchers in creating a prototype ontology for small sized test cases and ends with a summary of their findings and the future development by using more test cases to modify the ontology to further improve the communication quality and efficiency and thus reduce medical errors. The report assumed that communication records between clinicians within the medical field are properly recorded and can be used in the ontology, but most communications in the day to day operations in the medical field are not. The usage of a dedicated software to implement the ontology is noteworthy for our project.

5. Ontology Languages – A Review

   This paper authored by Maniraj and Sivakumar (2010) describes different features and issues of three types of ontology languages: 1) Logical languages, which includes first order predicate logic, rule based logic and description logic, 2) frame based languages, which are similar to relational databases, and 3) graph based languages, which consist of semantic networks. The two authors then proceed to describe different types of ontology languages that conform to the above three classifications. At the end, discusses whether RDF (a popular ontology language) should be considered as best base language or not for implementing ontologies on the web by detailing its pros and cons and comparing it with other ontology languages.

6. Applying Ontologies to Terminology: Advantages and Disadvantages

   From the article authored by Isabel and Maria(2013), the two authors start off by mentioning how ontologies can be used across the terminology field and the main recent uses within different applications., followed by the various advantages and disadvantages concerning the application of ontologies to terminologies. One notable example of an advantage is clear organisation of specialised knowledge, by means of a "macrostructure" - a way of representing the conceptual structure that underlies a particular domain, so that the categories that consist of that domain can be modelled, and one notable disadvantage is that there are too many ontological languages, so this makes it hard for systems that do not share the same languages to interchange or reuse data. In the end, both authors finish by discussing the significant limitations that were observed overall and making a few final comments regarding the usage of ontologies in terminology work.

7. The OWL API: A Java API for OWL ontologies

   This research article details the OWL API used for the software of this project. The article highlights how the owl api is designed based on the OWL 2 version and the specifications for its functions and the reasoning behind the design through the explanation of the models used. It goes into detail on the ontological structures and the reasoning engines involved for the implementation of OWL API.

8. Cooking the Semantic Web with the OWL API

   This paper was included in the International Semantic Web Conference 2003 and discusses the problems with the OWL API and other attempts at creating an api for ontology development. It includes an analysis on the problems and provides solutions to technical issues found in the systems that follow the OWL standard. It includes a deep dive into the usage of the OWL API which is used in the software for this project, with discussion on its advantages and disadvantages and related components like editors, query reasoner with explanations on the usages.

[removed 3 research papers that became irrelevant from updated deliverables]

D. Conclusion of literature review

Across the research papers found related to the usage of ontology and other relevant methodology, there are many well designed architectures for ontology usage. The steps for these proposed systems can be summarised as, initial data separation, a general categorization, content breakdown, keyword evaluation, domain creation, data hierarchy creation and maintenance. There has been numerous research on these individual parts on different aspects of the medical field. The testing and development of more effective methods for data interpolation are still ongoing, with seemingly positive results in the future. The usage of ontology in the medical field will be a slow change as it requires a systematic change of many aspects of the medical operations from communications to data bookkeeping.

==

Outcomes

==

a.What has been implemented

We had successfully implemented a few key things for our project, which is to create an ontology model for heart diseases and its symptoms in our project. We converted the ontology model into OWL file type so that it acts as our database for this project. Link from OWL file database to our software has been implemented through OWL API so that we can retrieve information stored. In addition, the user query for disease through symptoms has been implemented and integrated with OWL API in order to do comparison with symptoms stored in the database and eventually return the list of diseases with the input symptoms based on the order of severity rate. A very simple User Interface has been implemented in the software to ensure that it is user-friendly and easy to navigate. Unit testing, system integration testing as well as usability testing is done to ensure that our implementation of software does not have any error that can cause failure in running, as well as to ensure that expected input will provide accurate expected output.

b.Results achieved/product delivered

The results achieved is that we successfully developed a software written in Java which enables users to provide input of symptoms of disease, and then obtain the list of diseases with the input symptoms based on severity rating. We developed a simple User Interface so that there's little to no learning curve for our software. When the software opens, it will be the opening page with only a button to open the main page to input symptoms. Once the button is clicked, it will change to the page for the user to input a query and press the search button to begin searching. Once the button is clicked, if there's a match, the user will be redirected to another page showing a barchart for number of diseases based on severity rating. Each of these bar charts can be clicked to show the list of diseases and it can be clicked to show the description of the particular disease. Else, the user will be redirected to a query error page so that the user will know there's no match for heart disease with the input symptoms. Back button is implemented on all pages so that users can travel back to the query page to input another query.

c. How are requirements met

Our requirements are met by successfully implementing a software that uses an ontology model created for heart diseases which can serve as a proof of concept that ontology can be used in the medical field.

d. Justification of decisions made

- Transition from developing Android application to pc software
  Due to the OWL API being implemented for the main purpose of manipulating owl ontology models for web applications and pc software, there's been quite a number of problems with letting it run on Android platform. After searching for solutions but to no avail, we decided to change the platform of our project and hence Java based software is used as the previous development of our code was using Java as well, just that the Java versions and functionalities on Android platform is lesser.

- Implementation of simple UI
  Due to the modification of project requirements throughout the first few weeks of this semester, we did not have enough time to delve deeper into the ontology model and UI so we agreed to have a simple UI so that users can learn to use it easily and it eased our development process.
- Decision between OWL API vs Protege OWL API
  We chose OWL API instead of Protege OWL API to manipulate our ontology model file as the support for Protege OWL API has ceased and the last update was in 2010 whereas the OWL API is still up to date and last update was earlier this year. We figured that it would have more support since it's been well maintained throughout the years so we chose to use OWL API to manipulate our ontology model file in our software.

e. Discussion of all results

Our software is able to return accurate results based on user input if it matches the data in our ontology model. This shows that the pattern matching progress is efficient and accurate as we are able to return results in a short amount of time. However, the ontology model for our project is quite small and hence it is not able to include all the existing heart diseases in the real world. The severity rating is done based on the mortality rate and death count that we did manually, so it does not reflect the actual severity of the particular heart disease from the authorities.

f. Limitations of project outcomes

Despite the fact that we successfully implemented the software aligned with requirements, there's still a lot of limitations on it. For example, it's difficult to find a real life picture to visualise those heart diseases in our software, so we ended up not getting to visualise them. Furthermore, the severity rating is what we manually did based on a number of factors, mortality rate and death count for those heart diseases, hence it did not reflect the actual severity rate from the authorities.

g. Discussion of possible improvements and future works

As for the enhancement of our project software, there are a few possible improvements and future works to improve it. The current ontology model is too small and so we can make it larger by adding more heart diseases into it. Another possible improvement will be that we can add more functionalities to the query so that it does not just allow users to input symptoms and obtain the result, it can be improved by working in the reverse way as well. In addition, in the future we may add more diseases not limited to heart disease so as to integrate the usage of ontology models in a wider medical field. Lastly, as mentioned in the limitations above regarding difficulty in visualisation of heart diseases, it can be done if we can get assistance from a heart disease specialist.

h. Any other matter of relevance and interest

One of our project requirements is to prove that ontology can be integrated into the medical field to be used as a method of classification. Despite the fact that we successfully proved it via the implementation of the software, it still has quite a number of limitations which resulted in it not being able to be utilised fully in the medical field. It is always used in the medical field as a mixture with other classification methods. For example, Systematized Nomenclature of Medicine—Clinical Terms (SNOMED CT, hereafter abbreviated SCT) is a comprehensive medical terminology used for standardising the storage, retrieval, and exchange of electronic health data. Some efforts have been made to capture the contents of SCT as Web Ontology Language (OWL), but these efforts have been hampered by the size and complexity of SCT. This fact based on a research paper showed us that the route to fully utilise ontology in the medical field is still very long as we will have to be able to generate ontology for complex SCT in order to use ontology for electronic health data.

===

Ontology Database Methodology

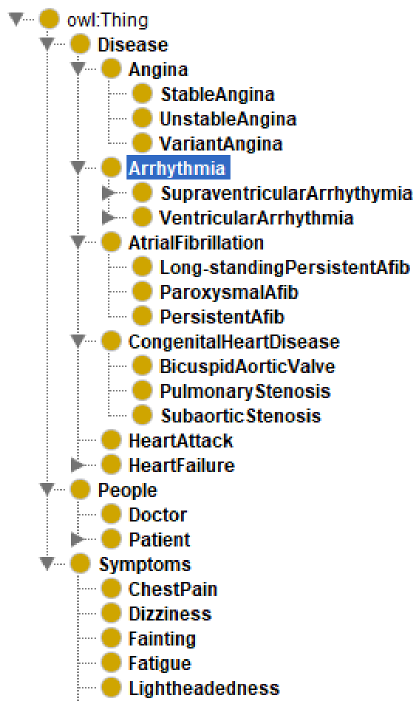
===

Our initial plan in semester 1 was to use lucidchart to design the ontology as a tree structure. In this design, nodes would symbolise concepts such as heart diseases, people, and symptoms, while edges would be used to establish various connections between these concepts. Furthermore, each node will also have the capability to incorporate additional properties for a more comprehensive description of the concepts. This tree structure will then be organised in a hierarchical order to showcase how the various concepts, relationships and data related to heart disease can be all connected together so that clinicians can extract meaningful data from the ontology structure as an effective way to understand heart disease progression. However, there are several flaws to this approach: 1) this design cannot be used to perform queries that can return diseases based on symptoms accepted as input because it is just a graphical representation, 2) the correctness of this ontology structure cannot be verified due to the lack of a "reasoner", and 3) there is little to no coding involved, because we only need to display the tree structure for our stakeholders in a user interface, which defeats the purpose of the final year project. Fortunately, in semester 2, we came up with a new design to create the ontology. Our new design utilises protege to create an ontology database instead; this design differs from the previous design because it involves us creating an ontology OWL file that can be interacted through the use of OWLAPI. In addition, we can also display the ontology structure through a built-in function in protege (OntoGraf). Essentially, our new design is capable of preserving all of the benefits of the old implementation while simultaneously addressing all the issues mentioned in the previous implementation.

1) Background

Some background knowledge on terminologies is needed before diving into the details of the ontology database methodology. Classes are interpreted as sets that contain individuals and are a concrete representation of concepts; the word concept is sometimes used in place of class. Individuals represent objects in the domain in which we are interested. Individuals are also known as instances. Properties link individuals from the domain to individuals from the range. For example, in our heart disease ontology, the property hasSymptom would link individuals belonging to the class Disease (Domain) to individuals belonging to the class of Symptom (range). Properties are binary relations on individuals. In other words, properties link two individuals together. In our case, we are mainly interested in two properties: object and annotation properties. Object properties are used to establish relationships between two individuals and annotation properties are used to add information (i.e. metadata) to classes, individuals and object/datatype properties.

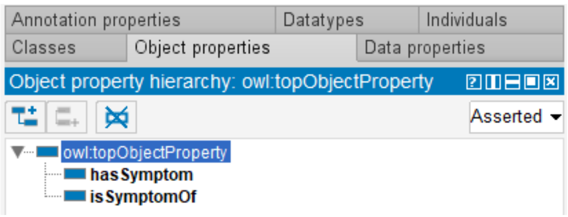
2) Create Classes



```
owl:Thing
    Disease
        Angina
            StableAngina
            UnstableAngina
            VariantAngina
        Arrhythmia
            SupraventricularArrhythymia
            VentricularArrhythmia
        AtrialFibrillation
            Long-standingPersistentAfib
            ParoxysmalAfib
            PersistentAfib
        CongenitalHeartDisease
            BicuspidAorticValve
            PulmonaryStenosis
            SubaorticStenosis
        HeartAttack
        HeartFailure
    People
        Doctor
        Patient
    Symptoms
        ChestPain
        Dizziness
        Fainting
        Fatigue
        Lightheadedness
```

Our new design is first implemented by defining diseases and symptoms and people as main classes. Each of these main classes will then be populated with classes related to the main class (e.g. angina, arrhythmia and atrial fibrillation classes will be under Disease class) and then configured to be disjoint to each other. In other words, an individual (or object) cannot be an instance of more than one of these three classes. Because some diseases can be further broken down into more specific kinds of diseases (e.g. angina can be broken down into stable angina, unstable angina or variant angina), we also need to create subclasses under each disease class so as to cater to these specific kinds of diseases. The specific disease subclasses will also be subjected to the same configuration as its parent i.e. subclasses will be disjointed with other subclasses belonging to the same disease (e.g. stable angina, unstable angina and variant angina classes under angina class will be mutually exclusive).

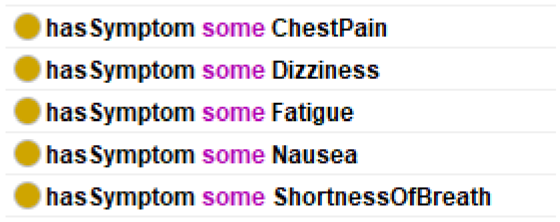
3) Create Object Properties



```
owl:topObjectProperty
    hasSymptom
    isSymptomOf
```

Within our design, we will establish a connection between the disease classes and symptom classes by creating two object properties. The first one will be named "hasSymptom," and the second one will be called "isSymptomOf." The former is used to link individuals from Disease class to individuals from Symptom class, and the latter is used to link individuals from Symptom class to individuals from Disease class. This is done so that existential restrictions can be defined for each disease class and symptom class

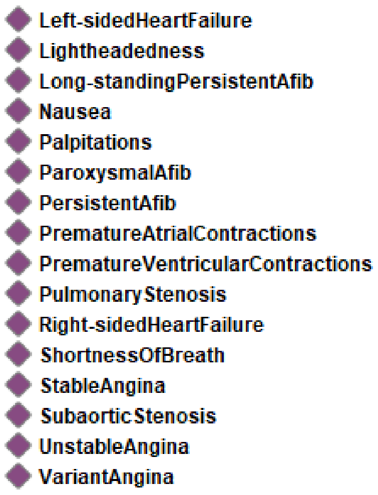
4) Create Existential Restrictions



hasSymptom some ChestPain
hasSymptom some Dizziness
hasSymptom some Fatigue
hasSymptom some Nausea
hasSymptom some ShortnessOfBreath

Existential restrictions help to describe classes of individuals that participate in at least one relationship along a specified property to individuals that are members of a specified class. For example, “the class of StableAngina has at least one (some) hasSymptom relationship to class Nausea(shown in diagram above)”. Each disease class will use the “hasSymptom” object property to link a disease to a particular symptom, and each symptom class will use the “isSymptomOf” object property to link a symptom to a particular Disease, thereby creating an existential restriction for both sets of classes.

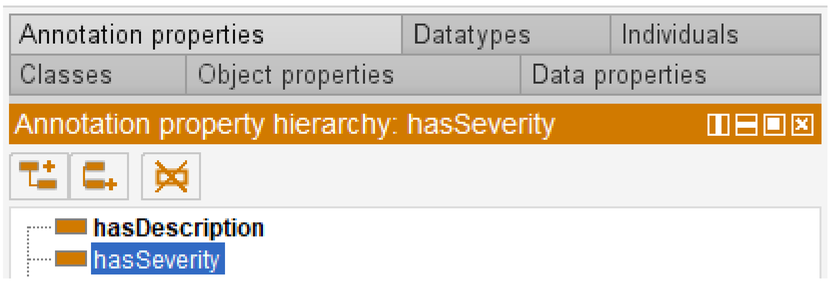
5) Create Individuals



Left-sidedHeartFailure
Lightheadedness
Long-standingPersistentAfib
Nausea
Palpitations
ParoxysmalAfib
PersistentAfib
PrematureAtrialContractions
PrematureVentricularContractions
PulmonaryStenosis
Right-sidedHeartFailure
ShortnessOfBreath
StableAngina
SubaorticStenosis
UnstableAngina
VariantAngina

Individuals are defined for each disease so that it can be returned as a list of items after performing the query. The same is done for symptoms so that we can filter out irrelevant diseases based on the symptoms inputted from the query. The string from the query can be parsed by naming every individual we have created to be identical to their corresponding disease and symptom classes (as shown in diagram above). Each individual will then be assigned to a particular type of class depending on the object property assertions that we have imposed onto.

6) Create Annotation Properties



At last, 2 annotation properties will be created. The first one will be labelled as “hasDescription”, and the second will be named “hasSeverity”. These 2 properties will then be applied to every disease class. By doing so, we can extract the description and severity defined for each disease and then display the two through our user interface.

==

UI Methodology

==

Based on the UI design methodology listed in the proposal, the stages for the development of the UI remained fairly the same. The UI design methodology follows the Product Development Life Cycle methodology involving ideation,scope,prototyping,initial iteration, testing and finalisation. However the overall design of the user interface for the software was changed due to the alteration of the project api used. Due to the switch from protege api on android studio ide, the project was transferred to the eclipse java ide due to outdated maintenance for api functionality. The layout of the overall design of the app has shifted due to an alteration of project goals. The software no longer supports the user importation of medical record files and owl ontologies to ensure that the privacy of medical records is maintained. The visualisation for the ontology database is replaced with a visual categorization of the resulting query from the ontology model. Additional details for the queried results were included in the results page to give clinicians more information about the illness.

The design of the new user interface followed the Development Life Cycle methodology, with initial plans for the overall and design of the pages followed by proposed software functionality. Lucidchart was used as the main designing software for the overall look of the software with low and high fidelity prototypes used for prototyping. For software testing, the eclipse ide was used with different java executables to isolate functions created that utilise the owl api. The objective for the design is to have a simple and straightforward interface where all clinicians can pick up and use immediately.

==

Software Deliverables

==

A. Summary of software deliverables

I. ***What is delivered***

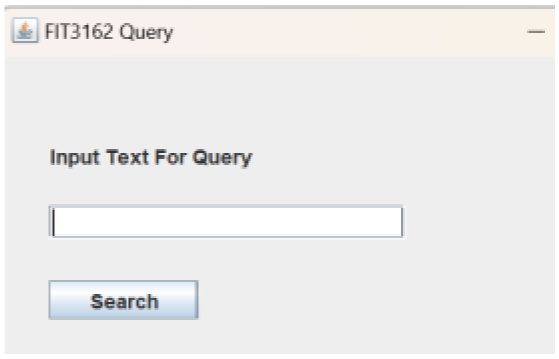
1. **Executable program that runs the software for clinicians**
   Opens a welcome page that links to the main search page, uses the owl api to query results based on user input keywords on protege created ontology model, categorisation of results based on severity, descriptions of resulting illness.

2. **Owl protege ontology model**
   Owl file that contains the custom made ontology model as a proof of concept for the project, can be imported and edited using the protege software.

3. **Project archive**
   Contains all the source files imports and user interface java files, there are many imports for the project, with most versions having conflict with multiple parts of the owl api, the functional import versions are included in the archive.

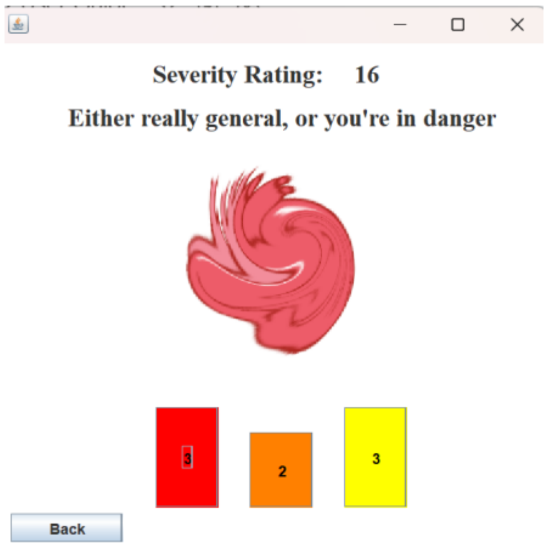
II. ***Sample screenshots and description of usage***

1. Query page



   Includes a simple and clear user interface for users to input symptoms that are case insensitive, with inclusion of multiple values separated by commas. Press the search button to open the results page if a match is found, else an error page is displayed with the option to try again.

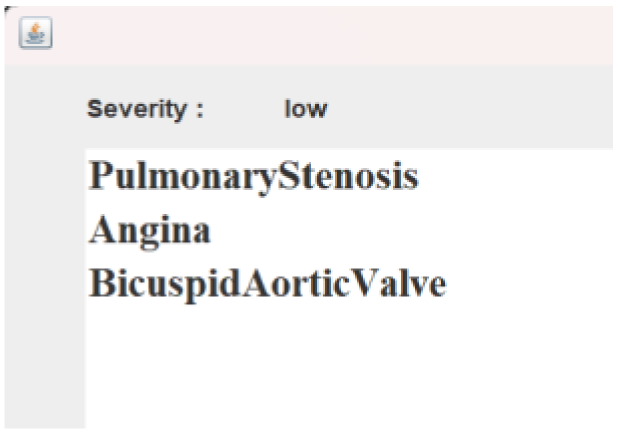
2. Results page


Severity Rating: 16

Either really general, or you're in danger

Displays the results from the input query, with a severity rating score based on the returned results, categorisation of results based on severity (high, medium, low) with different colours. Select a column to open the list view.

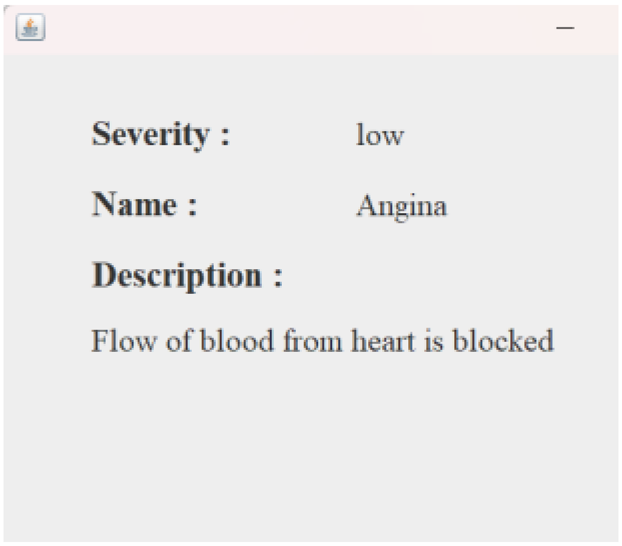
3. List view


Severity : low

PulmonaryStenosis
Angina
BicuspidAorticValve

Displays results with the same severity in a list, select an item to open details page.

4. Details page


Severity : low

Name : Angina

Description :

Flow of blood from heart is blocked

Displays additional information on the selected result.

B. Summary and discussion of software qualities

i. Robustness

To improve the software’s robustness, exception handling is implemented in our code as shown in the Appendix section. Furthermore, when a user inputs a symptom that is not in our ontology model, an error message will be shown in the User Interface and the user will be prompted to provide input for the symptom again. All these will be able to ensure our software’s robustness as it is just a simple software. However, there’s no protection for the ontology model file that’s included in the software file path. Hence any malicious modifications made to the ontology model file will have risks of making it output undesired and unexpected results.

ii. Security

The software is vulnerable to software supply chain risk as there’s dependencies used in our software. If the Maven dependencies for OWL API are vulnerable, it will make our software vulnerable as well. Apart from that, the software itself operates only on local machines with the ontology model file which is included in the file path, so it is secure unless there’s other people using the same machine and he or she modifies the ontology model file which in turn will be a threat.

iii. Usability

All pages include a back button to return the previous page, nested pages after the result pages will have the search query saved to reduce querying multiple times and loss of query result.

iv. Scalability

In terms of scalability, the ontology model itself is scalable as it allows easier addition of data into the model itself and it does not require much effort as well as impact the performance of the software.

<u>v. Documentation and Maintainability</u>

We made use of the Github repository to manage our documentation and update our code implementation. Branches were created so that we can have a good collaboration while splitting tasks. We also generated Javadocs automatically using Eclipse for documentation purposes. Comments were added on important parts of code so that team members could easily understand the code and would not get confused.

==
Critical Discussion
==
Our initial project proposal is to create an application that describes how medical ontology can be used to describe the concepts of medical terminologies and the relation between them, and convince clinicians why and how ontology should be used in healthcare or in expert systems. In the end, we created a desktop application that can receive a list of symptoms as input and generate a list of potential diseases as output. This information will be displayed with a graphical representation of a heart, alongside severity ratings and detailed descriptions for each disease identified through the query. However, the path to reach this point is not straightforward. At the beginning, the team thought that an ontology is simply just an organised tree structure that can demonstrate the interconnected nature of various concepts, relationships, and data associated with heart disease. So we chose an easy way out by creating a UI that shows an image of the ontology tree structure. This was later rejected by us because it is not technical enough. Shortly after, we proposed an idea to build an ontology database that exploits Natural Language Processing (NLP) algorithms to make free text description in electronic health records machine-interpretable. But we soon realised this is not a task that is feasible within 12 weeks, because it is very hard to learn NLP concepts from scratch without any guidance at all, much less understand research papers that focus on NLP concepts. By the fourth week, a meeting took place during which our supervisor suggested that we explore the use of Protege to construct an ontology database. Subsequently, all members of the group agreed and committed to dedicating the next two weeks to studying the ProtegeOWL tutorial. After the completion of this tutorial, everyone developed an understanding of how ontology not only could be used as a graphical representation to aid in understanding heart diseases, but to also execute queries that return diseases based on specific symptoms as input. Based on this understanding, we can finally devise a plan that can achieve our project proposal. This plan is to create an application capable of receiving a symptom list as input and producing a list of potential diseases as output. This data will be presented in a display that includes a heart visualisation, along with severity ratings and detailed descriptions for each disease identified through the query.

However, there are many obstacles we have faced before we can successfully implement our plan. Initially, we wanted to create a mobile application but later changed to a java-based desktop application instead due to dependency issues. We approached our project plan by first dividing 3 main tasks to each member separately using a gantt chart. The main tasks are: 1) Create UI, 2) Create ontology database, and 3) utilise OWLAPI to integrate the ontology database together with the UI. The priority is to complete the ontology database first, as without it, we cannot test the OWLAPI inside eclipse IDE. Unfortunately, we later realised that there are no up-to-date guides to be found for OWLAPI. The few resources that are accessible are outdated by at least a decade. So it is very hard to test and use the functionalities provided by OWLAPI. Hence, we restrategized our approach by making everyone focus on testing the OWLAPI before creating the UI. As a result, everyone can understand the functionalities provided by OWLAPI, and this understanding can later enable us to create a more complex ontology database. This, in turn, can enable the development of a more sophisticated user interface. For example, at first our ontology database only consisted of classes, object properties and instances; this is good enough to create a basic query function, where a set of diseases is returned after inputting a set of symptoms. However, our final implementation also includes the use of annotation properties, which enable us to display the description and severity rating for each disease output from the query in the UI. This can only happen because everyone understands the capability and functionalities provided by OWLAPI.

Overall, our final implementation just barely met the original project proposal and there is much more room for improvement in our current work. Currently, the biggest flaw in our implementation is the fact that we don't use any clinically validated dataset to establish the relationship between symptoms and diseases. In other words, the selection of diseases provided as output when you input symptoms is determined by the data we have manually curated ourselves. We could have used data science techniques to analyse and extract the relationship between symptoms and heart diseases from datasets. Unfortunately, no one in the team is trained on data science, so this option is not doable by us given the time restraints. Another flaw is that our UI has not been validated by an actual clinician; thus making it uncertain whether our UI is sufficiently user-friendly for practical clinical use. This problem can be solved by frequently communicating with an actual clinician, so that feedback can be obtained to improve on our UI.

==

Conclusion

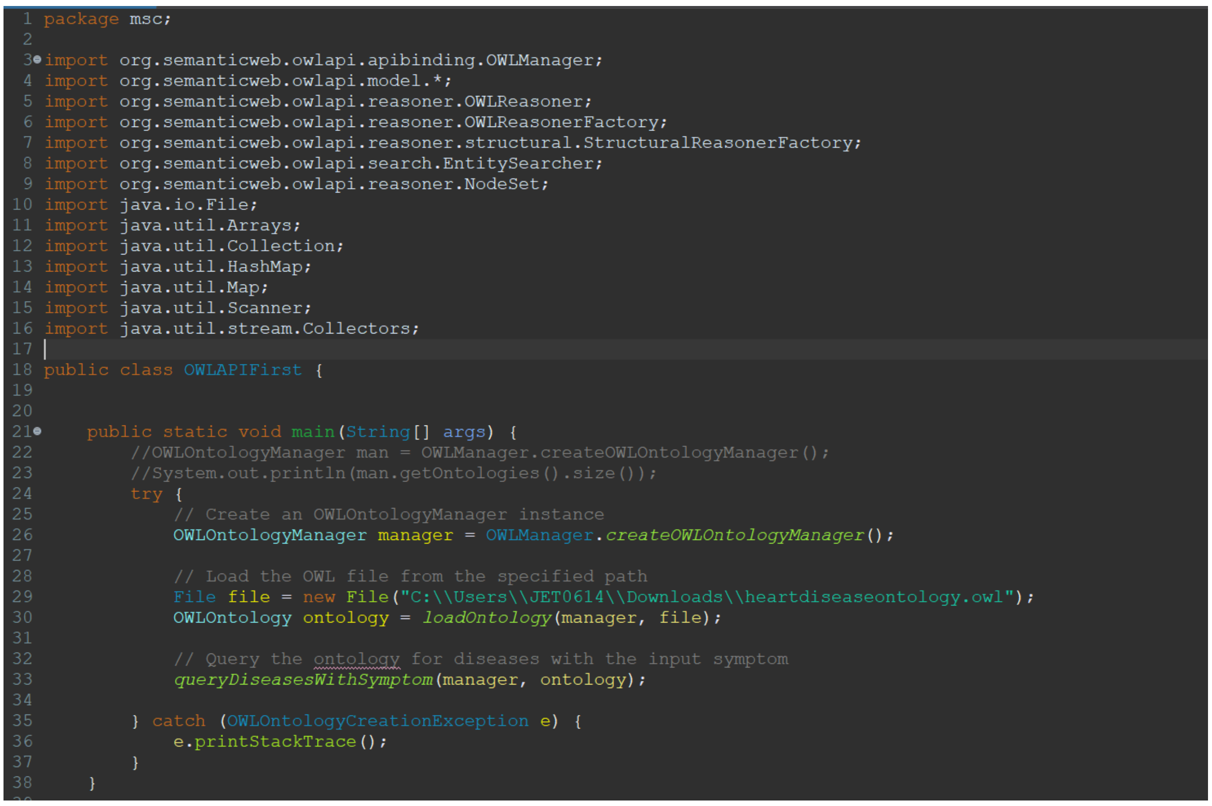
==

In conclusion, this project has successfully demonstrated the potential of ontologies in the medical field. The project has shown that medical ontologies can significantly enhance data interoperability and enable sophisticated knowledge management. The ontology developed in this project has the potential to be integrated into various healthcare systems to improve the patient diagnosis process. However, it is important to note that the effectiveness of a medical ontology largely depends on its continuous refinement and adaptation to evolving medical knowledge and practices. Therefore, future work should focus on expanding and updating the ontology to ensure its relevance and utility in the ever-changing landscape of healthcare. This project serves as a stepping stone towards the broader application of ontologies in the medical field. As a team, we learned a lot throughout all the phases of this project. We would like to specifically thank our project supervisor, Dr Nailah and the unit coordinator, Dr Vishnu for their support and guidance throughout this project.

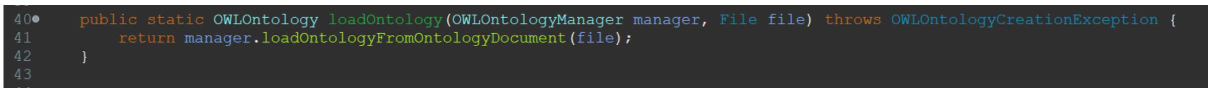
===

Appendix

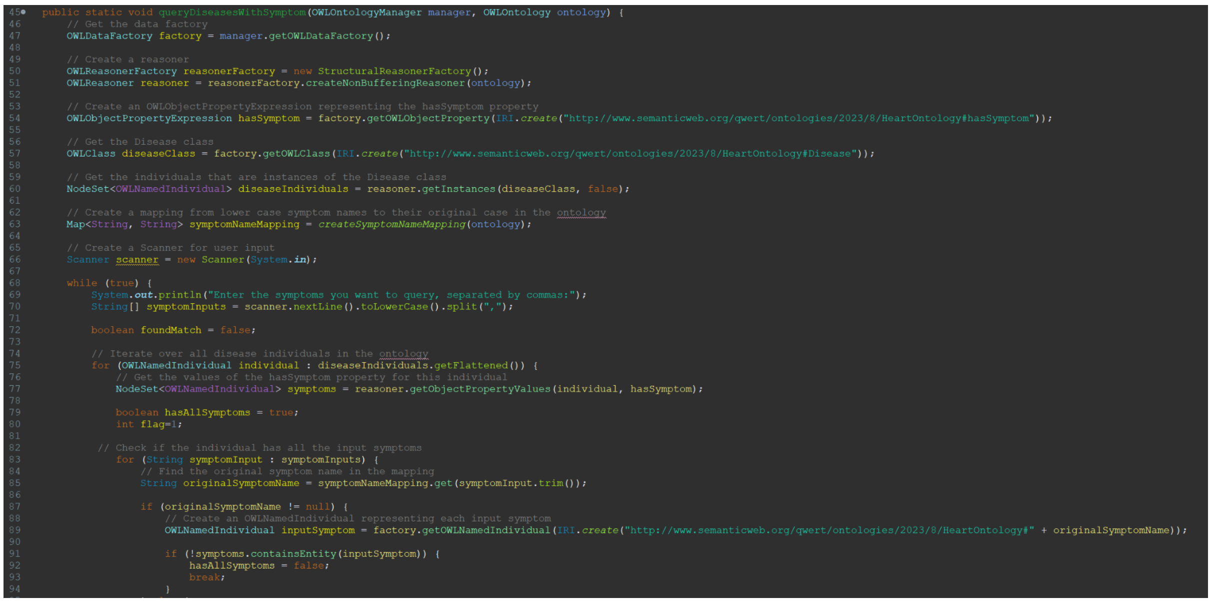
===



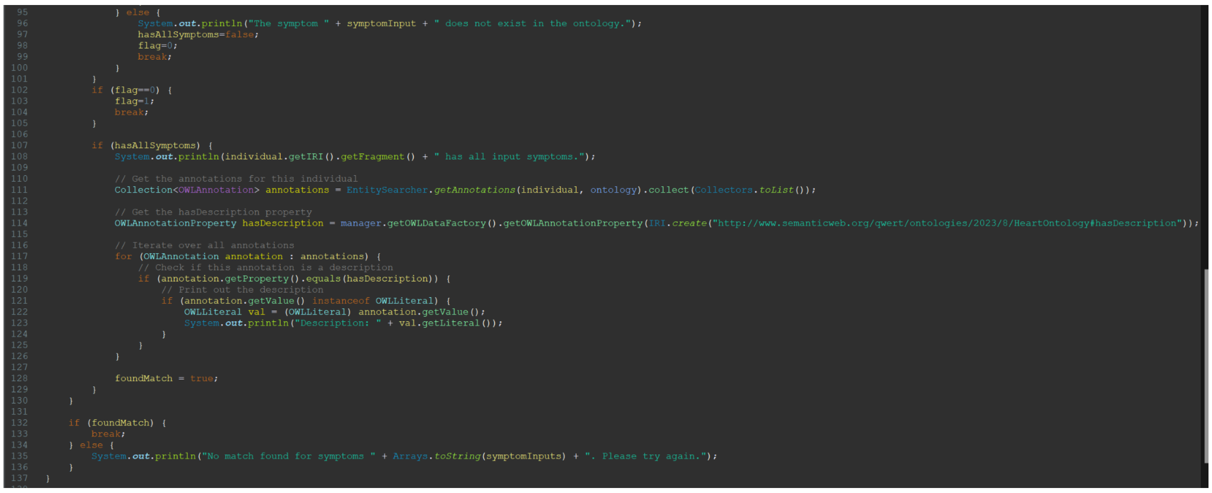
```java
package msc;

import org.semanticweb.owlapi.apibinding.OWLManager;
import org.semanticweb.owlapi.model.*;
import org.semanticweb.owlapi.reasoner.OWLReasoner;
import org.semanticweb.owlapi.reasoner.OWLReasonerFactory;
import org.semanticweb.owlapi.reasoner.structural.StructuralReasonerFactory;
import org.semanticweb.owlapi.search.EntitySearcher;
import org.semanticweb.owlapi.reasoner.NodeSet;
import java.io.File;
import java.util.Arrays;
import java.util.Collection;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
import java.util.stream.Collectors;

public class OWLAPIFirst {


    public static void main(String[] args) {
        //OWLOntologyManager man = OWLManager.createOWLOntologyManager();
        //System.out.println(man.getOntologies().size());
        try {
            // Create an OWLOntologyManager instance
            OWLOntologyManager manager = OWLManager.createOWLOntologyManager();

            // Load the OWL file from the specified path
            File file = new File("C:\\Users\\JET0614\\Downloads\\heartdiseaseontology.owl");
            OWLOntology ontology = loadOntology(manager, file);

            // Query the ontology for diseases with the input symptom
            queryDiseasesWithSymptom(manager, ontology);

        } catch (OWLOntologyCreationException e) {
            e.printStackTrace();
        }
    }
```



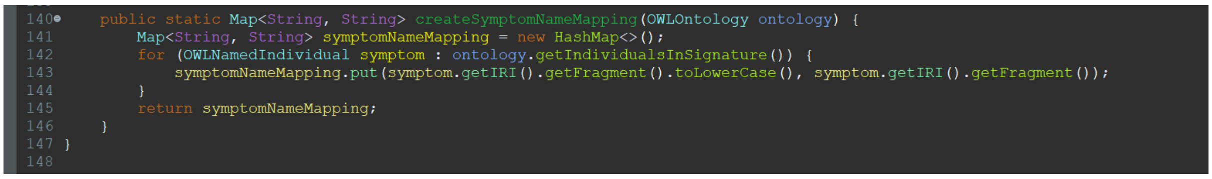
```java
    public static OWLOntology loadOntology(OWLOntologyManager manager, File file) throws OWLOntologyCreationException {
        return manager.loadOntologyFromOntologyDocument(file);
    }

```



```java
public static void queryDiseasesWithSymptom(OWLOntologyManager manager, OWLOntology ontology) {
    // Get the data factory
    OWLDataFactory factory = manager.getOWLDataFactory();

    // Create a reasoner
    OWLReasonerFactory reasonerFactory = new StructuralReasonerFactory();
    OWLReasoner reasoner = reasonerFactory.createNonBufferingReasoner(ontology);

    // Create an OWLObjectPropertyExpression representing the hasSymptom property
    OWLObjectPropertyExpression hasSymptom = factory.getOWLObjectProperty(IRI.create("http://www.semanticweb.org/qwert/ontologies/2023/8/HeartOntology#hasSymptom"));

    // Get the Disease class
    OWLClass diseaseClass = factory.getOWLClass(IRI.create("http://www.semanticweb.org/qwert/ontologies/2023/8/HeartOntology#Disease"));

    // Get the individuals that are instances of the Disease class
    NodeSet<OWLNamedIndividual> diseaseIndividuals = reasoner.getInstances(diseaseClass, false);

    // Create a mapping from lower case symptom names to their original case in the ontology
    Map<String, String> symptomNameMapping = createSymptomNameMapping(ontology);

    // Create a Scanner for user input
    Scanner scanner = new Scanner(System.in);

    while (true) {
        System.out.println("Enter the symptoms you want to query, separated by commas:");
        String[] symptomInputs = scanner.nextLine().toLowerCase().split(",");

        boolean foundMatch = false;

        // Iterate over all disease individuals in the ontology
        for (OWLNamedIndividual individual : diseaseIndividuals.getFlattened()) {
            // Get the values of the hasSymptom property for this individual
            NodeSet<OWLNamedIndividual> symptoms = reasoner.getObjectPropertyValues(individual, hasSymptom);

            boolean hasAllSymptoms = true;
            int flag=1;

         // Check if the individual has all the input symptoms
            for (String symptomInput : symptomInputs) {
                // Find the original symptom name in the mapping
                String originalSymptomName = symptomNameMapping.get(symptomInput.trim());

                if (originalSymptomName != null) {
                    // Create an OWLNamedIndividual representing each input symptom
                    OWLNamedIndividual inputSymptom = factory.getOWLNamedIndividual(IRI.create("http://www.semanticweb.org/qwert/ontologies/2023/8/HeartOntology#" + originalSymptomName));

                    if (!symptoms.containsEntity(inputSymptom)) {
                        hasAllSymptoms = false;
                        break;
                    }
```



```java
                } else {
                    System.out.println("The symptom " + symptomInput + " does not exist in the ontology.");
                    hasAllSymptoms=false;
                    flag=0;
                    break;
                }
            }
            if (flag==0) {
                flag=1;
                break;
            }

            if (hasAllSymptoms) {
                System.out.println(individual.getIRI().getFragment() + " has all input symptoms.");

                // Get the annotations for this individual
                Collection<OWLAnnotation> annotations = EntitySearcher.getAnnotations(individual, ontology).collect(Collectors.toList());

                // Get the hasDescription property
                OWLAnnotationProperty hasDescription = manager.getOWLDataFactory().getOWLAnnotationProperty(IRI.create("http://www.semanticweb.org/qwert/ontologies/2023/8/HeartOntology#hasDescription"));

                // Iterate over all annotations
                for (OWLAnnotation annotation : annotations) {
                    // Check if this annotation is a description
                    if (annotation.getProperty().equals(hasDescription)) {
                        // Print out the description
                        if (annotation.getValue() instanceof OWLLiteral) {
                            OWLLiteral val = (OWLLiteral) annotation.getValue();
                            System.out.println("Description: " + val.getLiteral());
                        }
                    }
                }

                foundMatch = true;
            }
        }

        if (foundMatch) {
            break;
        } else {
            System.out.println("No match found for symptoms " + Arrays.toString(symptomInputs) + ". Please try again.");
        }
    }
```



```java
    public static Map<String, String> createSymptomNameMapping(OWLOntology ontology) {
        Map<String, String> symptomNameMapping = new HashMap<>();
        for (OWLNamedIndividual symptom : ontology.getIndividualsInSignature()) {
            symptomNameMapping.put(symptom.getIRI().getFragment().toLowerCase(), symptom.getIRI().getFragment());
        }
        return symptomNameMapping;
    }
}

```

==

References

==

El-Sappagh, S., Franda, F., Ali, F., & Kwak, K.-S. (2018). SNOMED CT standard ontology based on the ontology for general medical science. BMC Medical Informatics and Decision Making, 18(1), 76. https://doi.org/10.1186/s12911-018-0651-5

Lit review references

IMO Staff. (n.d.). SNOMED CT 101: A guide to the international terminology system. https://www.imohealth.com/ideas/article/snomed-ct-101-a-guide-to-the-international-terminology-system/

Loriaux, A. (2022, March 3). Ontology 101: Why knowledge representation is important for clinical information systems. IMO. https://www.imohealth.com/ideas/article/ontology-101-why-knowledge-representation-is-important-for-clinical-information-systems/

Meraji, M., Sadoughi, F., Ahmadi, M., & Kahani, M. (2013). Designing an ontology-based health information system: A systematic approach. Life Science Journal, 10, 735–740.

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering: Principles and methods. Data & Knowledge Engineering, 25(1), 161–197. https://doi.org/https://doi.org/10.1016/S0169-023X(97)00056-6

Keet, M. (2018, December 18). 9: Ontology-Based Data Access. Engineering LibreTexts. https://eng.libretexts.org/Bookshelves/Computer_Science/Programming_and_Computation_Fundamentals/Book%3A_An_Introduction_to_Ontology_Engineering_(Keet)/09%3A_Ontology-Based_Data_Access

Cedeno-Moreno, D., & Vargas-Lombardo, M. (2018). An Ontology-Based Knowledge Methodology in the Medical Domain in the Latin America: the Study Case of Republic of Panama. Acta informatica medica : AIM : journal of the Society for Medical Informatics of Bosnia & Herzegovina : casopis Drustva za medicinsku informatiku BiH, 26(2), 98–101. https://doi.org/10.5455/aim.2018.26.98-101

Journal of Computer Science IJCSIS, J., & Manika, P. (2021). Building an ontology for cardiovascular diseases. Vol. 19 No. 1 JANUARY 2021 International Journal of Computer Science and Information Security (IJCSIS). https://doi.org/10.5281/zenodo.4533391

Prcela, M., Gamberger, D., & Jovic, A. (2008). Semantic web ontology utilization for heart failure expert system design. Studies in health technology and informatics, 136, 851–856.

Corporation, A. D. S. (n.d.). Five Benefits of Keeping Electronic Health Records. Retrieved 29 May 2023, from https://www.adsc.com/blog/five-benefits-keeping-electronic-health-records

Natural Language Processing in Healthcare Medical Records. (n.d.). ForeSee Medical.
Retrieved 29 May 2023, from
https://www.foreseemed.com/natural-language-processing-in-healthcare

Predictive Analytics in Healthcare—Benefits & Regulation. (n.d.). ForeSee Medical. Retrieved 29 May 2023, from
https://www.foreseemed.com/predictive-analytics-in-healthcare

Bechhofer, S. (2009). OWL: Web Ontology Language. In L. LIU & M. T. ÖZSU (Eds.), Encyclopedia of Database Systems (pp. 2008–2009). Springer US.
https://doi.org/10.1007/978-0-387-39940-9_1073

Manu Siddhartha. (2020). Heart Disease Dataset (Comprehensive). IEEE Dataport. https://dx.doi.org/10.21227/dz4t-cm36

Duque-Ramos, A., Fernandez-Breis, J., Stevens, R., & Aussenac-Gilles, N. (2011). OQuaRE: A SQuaRE-based approach for evaluating the quality of ontologies. Journal of Research and Practice in Information Technology, 43, 159–176.

Blomqvist, E., Seil Sepour, A., & Presutti, V. (2012). Ontology Testing - Methodology and Tool. In A. ten Teije, J. Völker, S. Handschuh, H. Stuckenschmidt, M. d'Acquin, A. Nikolov, N. Aussenac-Gilles, & N. Hernandez (Eds.), Knowledge Engineering and Knowledge Management (pp. 216–226)
https://doi.org/10.1007/978-3-642-33876-2_20

The Agile Software Development Life Cycle | Wrike Agile Guide. (n.d.). Retrieved 29 May 2023, from
https://www.wrike.com/agile-guide/agile-development-life-cycle/

Free Online Gantt Chart Software. (n.d.). Retrieved 29 May 2023, from
https://www.onlinegantt.com

Work Breakdown Structure. (n.d.). Workbreakdownstructure.Com. Retrieved 29 May 2023, from
https://www.workbreakdownstructure.com

Noy, N., & McGuinness, D. (n.d.). What is an ontology and why we need it. Retrieved

29 May 2023,
from
https://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html

Munir, K., & Sheraz Anjum, M. (2018). The use of ontologies for effective knowledge modelling and information retrieval. Applied Computing and Informatics, 14(2), 116–126. owl
https://doi.org/10.1016/j.aci.2017.07.003

González C, Blobel BG, López DM. Ontology-based framework for electronic health records interoperability. Stud Health Technol Inform. 2011;169:694-8. PMID: 21893836. https://pubmed.ncbi.nlm.nih.gov/21893836/

Blobel B. Analysis, design and implementation of secure and interoperable distributed health information systems. Stud Health Technol Inform. 2002;89:1-329. PMID: 15455834.
https://pubmed.ncbi.nlm.nih.gov/15455834/

Wong, W., Liu, W., & Bennamoun, M. (2012). Ontology learning from text. *ACM Computing Surveys*, *44*(4), 1–36.
https://doi.org/10.1145/2333112.2333115

A. Maedche, B. Motik, L. Stojanovic, R. Studer and R. Volz, "Ontologies for enterprise knowledge management," in IEEE Intelligent Systems, vol. 18, no. 2, pp. 26-33, March-April 2003, doi: 10.1109/MIS.2003.1193654.
https://ieeexplore.ieee.org/document/1193654

Maniraj, V., & Ramakrishnan, S. (2010). Ontology Languages – A Review. International
Journal of Computer Theory and Engineering, 2, 887–891.
https://doi.org/10.7763/IJCTE.2010.V2.257

Durán-Muñoz, I., & Bautista-Zambrana, M. R. (2017). Applying Ontologies to Terminology: Advantages and Disadvantages. HERMES - Journal of Language and Communication in Business, 26(51), 65–77.
https://doi.org/10.7146/HJLCB.V26I51.97438

Pisanelli, D. M. (2004). *Ontologies in Medicine*. IOS Press.

Barcellos, A. M., De Freitas Carneiro, P. a. B., Jiye, A., & Smith, B. (2011). The Blood Ontology: An ontology in the domain of hematology.
ICBO.
http://ceur-ws.org/Vol-833/paper29.pdf

Gong, Y., Zhu, M., Li, J., Turley, J. P., & Zhang, J. (2006). Clinical communication ontology for medical errors. *PubMed*, 930.
https://pubmed.ncbi.nlm.nih.gov/17238549

Baneyx A, Charlet J, Jaulent MC. Methodology to build medical ontology from textual resources. AMIA Annu Symp Proc. 2006;2006:21-5. PMID: 17238295; PMCID: PMC1839277.
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1839277/