Ontology Graphical UI

Arona Dorin Chowdhury - 1520045642; Mohammad Raihan Sarker Razu - 1520079042; Tamim Ahmed - 1520698642

Supervisor- Dr. Mohammad Ashrafuzzaman Khan; CSE499B Project Final Report; Section- 09; Department of Electrical and Computer Engineering- North South University

ABSTRACT

The purpose of this project is to create an interface where people can draw the graphs of their model easily and can edit anytime. In short, people can make an ontology very effectively. On many websites, we can see the graphs but we cannot edit the graph any time we want. Even once we draw a graph later we cannot add other classes or sub-classes. In that case, we have to redraw the full graph again. By protege, we can easily edit the graph according to the requirements.

1 LIST OF FIGURES

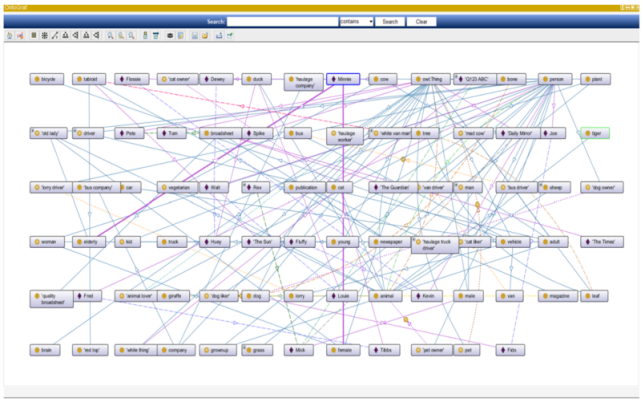

Figure 1: People ontology graph.

2 INTRODUCTION

The present is the age of information technology. Now the most valuable resource in the world is information or data. The world's largest tech companies rely on this information to make money. The most interesting thing is that the information is linked to one way or another. For example, when we say cancer, we understand that it is related to medical, any algorithm means related to computer science, processor update means related to electrical engineering, etc. These data are stored in the database in a static format. Which we do not directly understand which is associated with which.

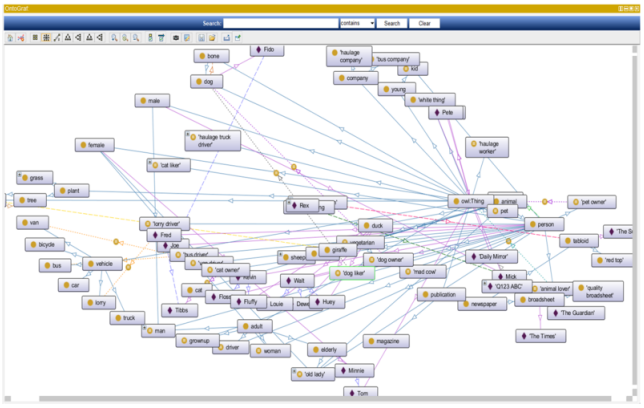

Figure 2: People ontology graph in different viewing design.

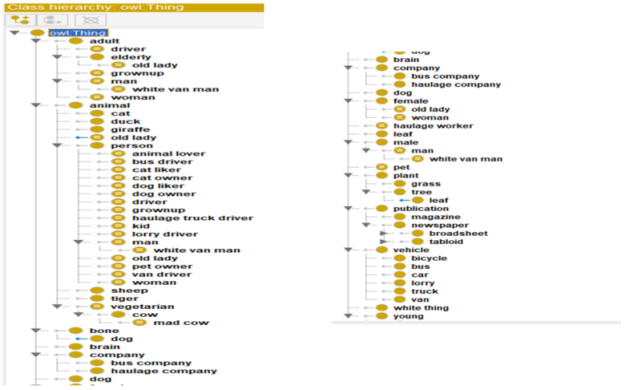

Figure 3: Class hierarchy.

, ,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

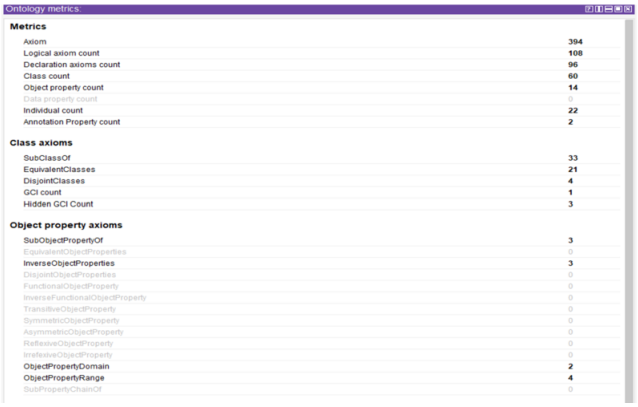

Figure 4: Ontology matrix.

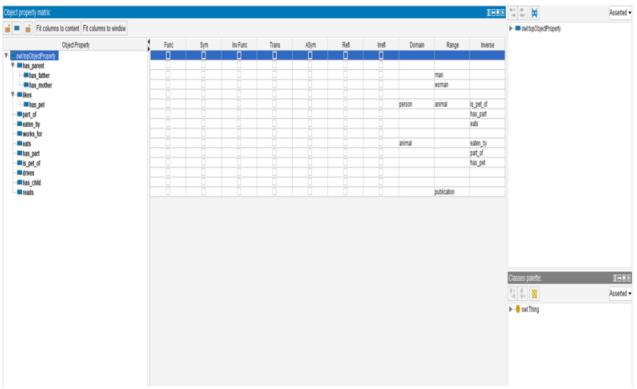

Figure 5: Object property matrix.

Figure 6: Annotation property hierarchy.

Ontology is used for a visual representation of data in dynamic form. (See figure 1 on page 1).

We can easily modify the ontology graph without any data change. (See figure 2 on page 1).

We know that some software is already being used to create data ontology [6] (such as protege) and it has some minor problems. So we initially want to create an ontology UI, so that the user can easily add, delete, rename data as class, sub-classes, or sub sub-classes. We plan to create it using web technology so that the user can use it very easily, so we don't need any PC configuration; anyone can

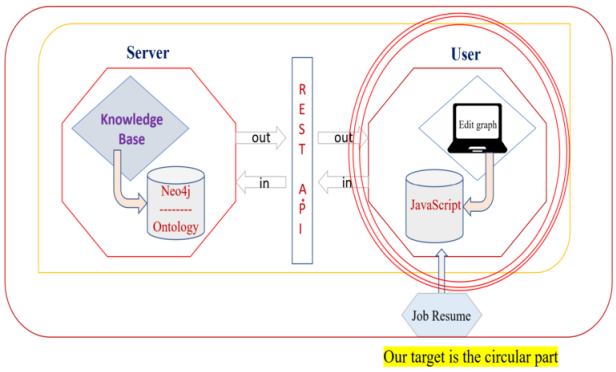

Figure 7: Project basic idea.

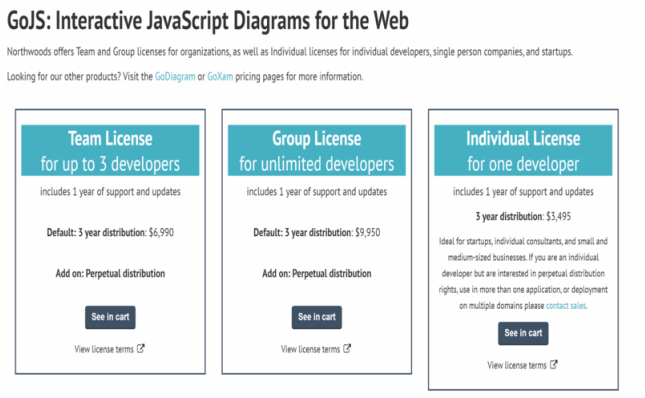

Figure 8: Price list of GOJS release version.

Figure 9: Initial value or data input in nodeDataArray.

take this facility only with the browser through the internet.

What are we trying to do-

We are trying to develop a system using the web platform, which will help the user to develop an ontology graph and can easily edit it at any time.

Why are we doing that-

We already know that there are many platforms for creating ontology. They also have some difficulties such as; Once an ontology

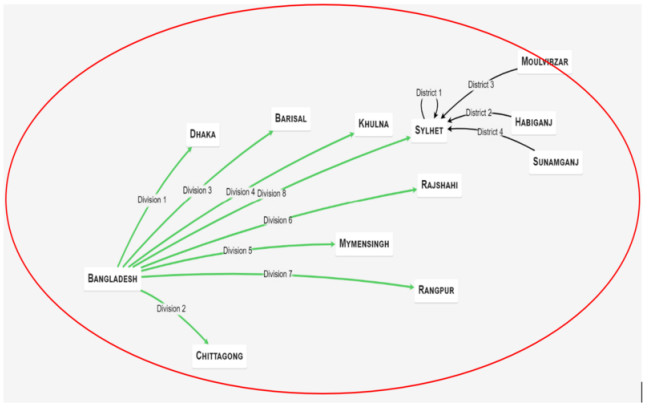

Figure 10: Initial value or data output for nodeDataArray in graph.

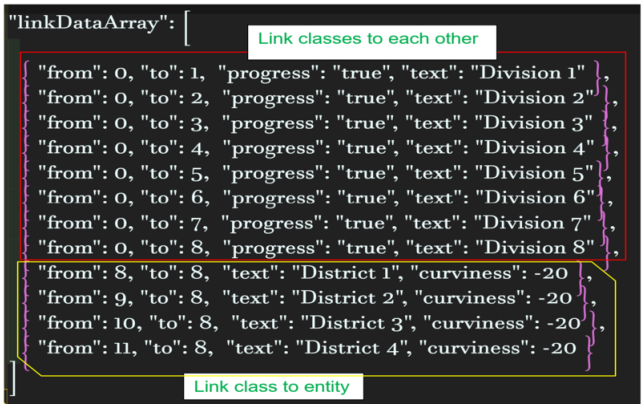

Figure 11: Initial link value or data each other in linkDataArray.

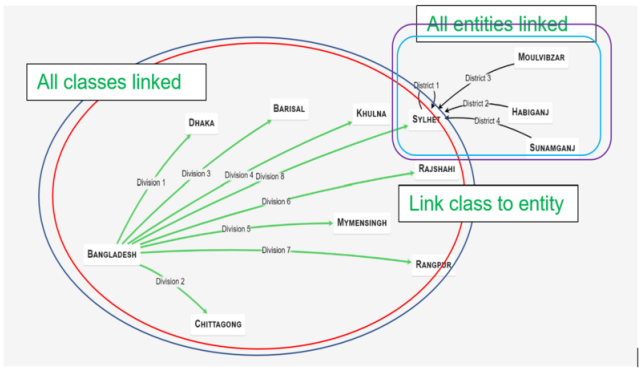

Figure 12: Output of Initial link value or data each other in the graph.

graph is created, it can no longer be edited, adding, deleting, renaming classes, or subclasses. To do this we need to create new ontology graphs; Which is a lot of hard and annoying.

How are we doing that-

Initially, we are trying to solve this with the web platform. At the very first stage, we search the libraries that are appropriate for our project. For faster and smooth development, we will use some libraries. Mainly we want to try Chartist.js or FusionCharts or Google Charts orChart.js libraries. These libraries will be used primarily to create graphs. As we do this on a web platform so we

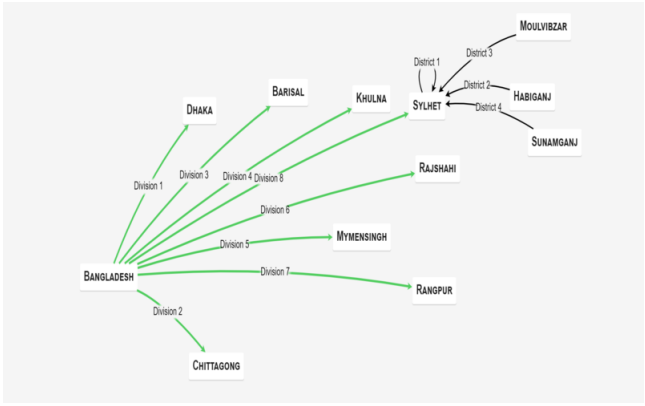

Figure 13: Mouse wheel zoom in.

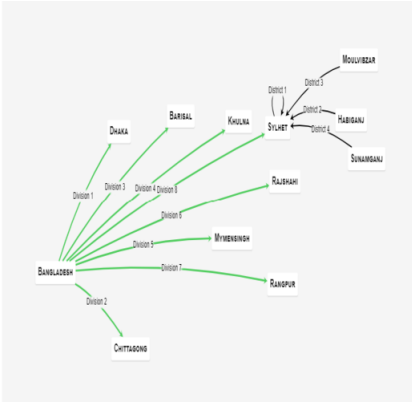

Figure 14: Mouse wheel zoom out.

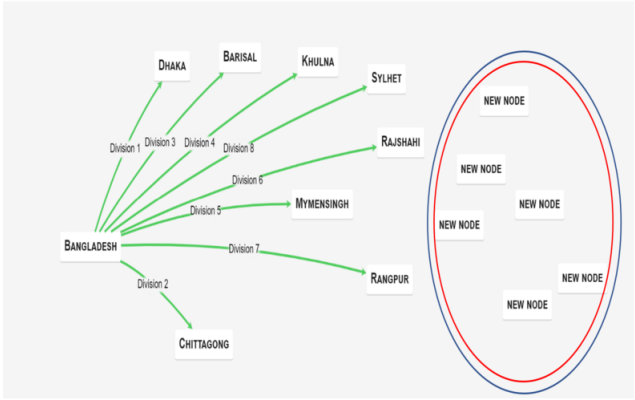

Figure 15: Mouse double click on empty space for creating new node.

use HTML5, CSS3, and JavaScript. If there is any server-side work then we will use PHP. For a text editor, we use Visual Code Studio IDE. We decide the first step we collect all the information that related our project, then analyze them to take out the raw material that we needed. In the third step, we develop the graph. After that, the most challenging part of our project is to apply the ontology rule in our graph. At the very last step, we merge all code and modify if it is necessary.

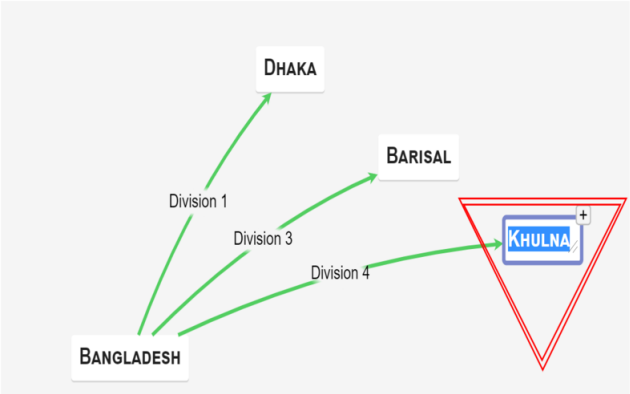

Figure 16: Mouse double click on the name.

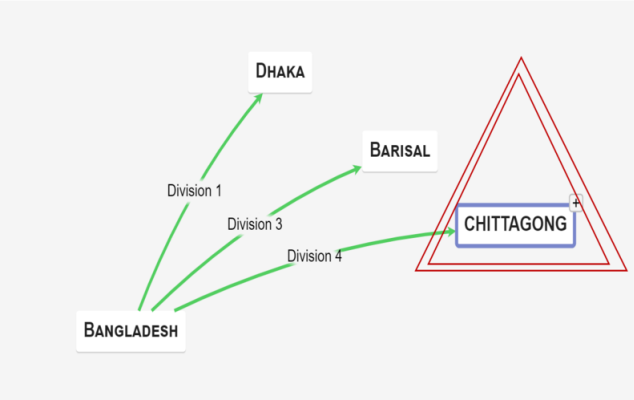

Figure 17: Writing new text.

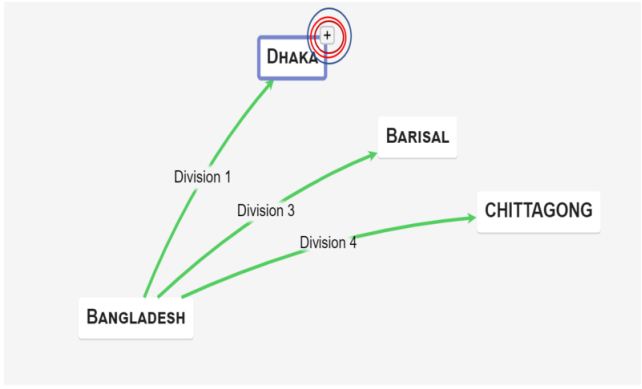

Figure 18: Adjust auto position to create new nodes inside the graph.

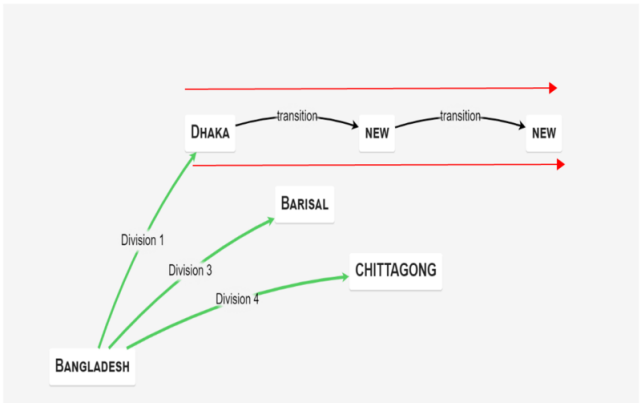

Figure 19: Create new node right side of the parent node.

Figure 20: Old data.

Figure 21: New Data.

3 BACKGROUND AND RELATED WORK

Protege is directly involved with our work. So for the convenience of our work we have done some research on Protege.

Protege-

is a free, open source ontology editor and a knowledge management system. Protege provides a graphic user interface to define ontologies [4]. It also includes deductive classifiers to validate that models are consistent and to infer new information based on the analysis of an ontology. Like Eclipse, Protege is a framework for which various other projects suggest plugins. This application is written in Java and heavily uses Swing to create the user interface. Protege recently has over 300,000 registered users. According to a 2009 book it is "the leading ontological engineering tool [8]. Protege is being developed at Stanford University and is made available under the BSD 2-clause license. Earlier versions of the tool were developed in collaboration with the University of Manchester.

Note-

In our project we are using latest version of Protege. Version is 5.5.0

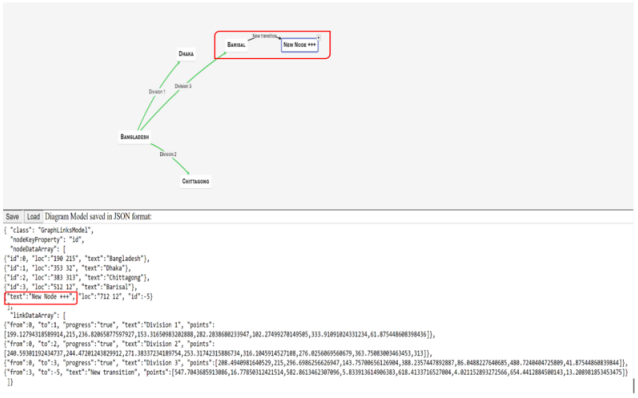

Figure 22: Update the new data.

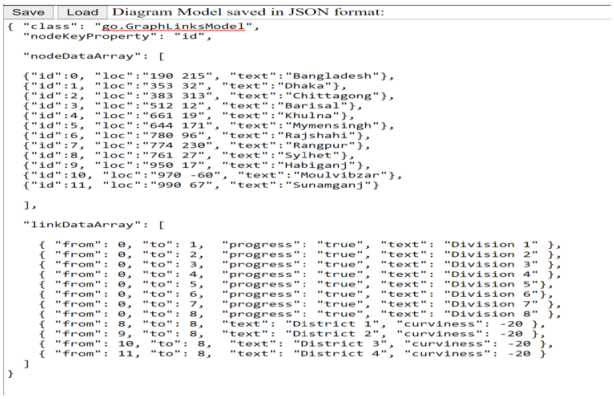

Figure 23: Showing data in JSON format.

which was last update 2019 14th march.

Using Protege-
First, make sure that "DIG Reasoner" is selected in the Reasoning menu (the default selection when you install Protege). The Reasoning menu allows you to select which is the current reasoned that should be used when the users classifies the ontology, checks the consistency, computes the inferred types, etc [12].

How works Protege-
Protege is a good exploratory and experimentation environment. Quick iterations are possible between model, data, and application changes. Database tables are designed and optimized to work with a particular application in mind. Instead access the data though the Protege API.

Some important figures for understanding Protege-
See figure 1 on page 1 shows the classes and sub-classes. This is the class hierarchy.

See figure 4 on page 2 shows the ontology matrix. It includes the properties and axioms.

See figure 5 on page 2 shows the object property matrix.

See figure 6 on page 2 shows the annotation property hierarchy.

4 METHODOLOGY/DESIGN

Here is the diagrams for the complete system. (See figure 7 on page 2).

Protege is a free open source ontology editor. It is used for drawing many graphs and easily can be edited at any time. Anyone can download this software from its official website because it is free. After installing this software we saw many libraries in this. They work differently while making the graph. We can create classes, sub-classes, and sub sub-classes. We can choose many types of viewing options as well. We create classes and add sub-classes in other sections (mentioned in the list of figures section). But at the very beginning when we tried to use this software, we faced many problems. And because this software is used in biomedicine, e-commerce, and organizational modeling, we rarely find any tutorial videos on the internet. Also, we read some papers according to our faculty's advice, but we could not find anything helpful. So it took a few weeks more than our expectations. But by the time we could figure it out by ourselves how this software works. Now we can create the graphs of any model we want. We can save a file and edit the graph anytime we want. Also, we can add or delete classes and sub-classes anytime according to our choices. By ontology-graph active ontology tab, we can show the model in graphs.

Basic requirements for our application-
Our project is completely software-based, so we don't need any hardware components. That's why we don't need anything other than a personal computer, code editor, and some other helpful tools.

Mainly we try to solve the problem using below features-
01. Gathering knowledge about the idea,
02. Reading related work papers,
03. Enlisted tools, librieris that we need to solve the problem,
04. developing graph;
05. Applying ontology rule into the graph.

Tools to be used-
01. HTML5, CSS3, JavaScript,
02. PHP [If needed],
04. Library -> Chartist.js or FusionCharts or Google Charts or Chart.js or GOJS,
05. IDE -> Visual Code Studio,
06. Server -> Apache 2.4.39 [If needed],
07. Database -> MySQL. [If needed]

Procedure and development-
After gathering all the information and reviewing some papers, we decided to develop a dynamic graph at the beginning. We initially used different libraries to create graphs; Which do not match our work perfectly. Later we got good results by using the GOJS library. We have used GOJS's debug CDN link, it is free so some functions work less in it. To use GOJS release CDN link you have to pay and that is quite a good amount; Minimum 3500 dollars to 9950 dollars. (See figure 8 on page 2).

We used the canvas property inside the HTML to make the graph visible. Moreover, I have used two buttons [save load] which have been used as on script action/event. We have used AJAX = Asynchronous JavaScript And XML to make web pages dynamic. Because when we click the button so that the page is not reloaded again and again. This means action/event will work without page reload.
We used the textarea property of HTML to sort the data or to keep it initial. Where I have used 2 arrays. 1 is nodeDataArray [], the other is linkDataArray []. In nodeDataArray we have used 3 variables- id, loc, text. The id here is that the node that I will create will have 1 id. I have used the location loc variable to determine where the node will be located depending on the values of x and y-axis. I have used the text variable to name the node. (See figure 9 on page 2). (See figure 10 on page 3).

In linkDataArray we also used 4 variables for class- from, to, progress, text, and entity also used 4 variables but 1 is a little different; curviness instead of progress. Here we have used the id of nodeDataArray in from and to define which node to which arrow will go. progress is a boolean variable. The text here indicates the transaction. I have used the curviness variable in entity to separate entity from class with another color arrow. (See figure 11 on page 3). (See figure 12 on page 3). In CSS we have used 4 simple properties- border, width, height, background. border used to define the boundaries of the graph; width, height to see how much space the graph will take up along the x-axis and y-axis on the screen; I have used background so that the graph is clear. All types of action / event work are done inside JavaScript. Basically, the work of editing the graph has been done here. The method or action / event that we have used is mentioned.
We use mouseWheelBehavior method to zoom in and out the graph with the mouse wheel. **Code-**

```
"toolManager.mouseWheelBehavior": go.ToolManager.WheelZoom,
```

(See figure 13 on page 3). (See figure 14 on page 3).

Using clickCreatingtOOL this method of GOJS library is called to create new nodes by double-clicking on empty space. **Code-**

```
"clickCreatingTool.archetypeNodeData":  text: "new node" ,
```

(See figure 15 on page 3).

Sometimes mistakenly we delete something or adding something wrong then we need to undo and redo option. that's why we use positionComputation for undo redo operation. If we press keyboard Ctrl+z then it action undo; and Ctrl+Y for the redo.
We also use textblock editable feature for editing the existing text. **Code-**

```
"undoManager.isEnabled": true,
positionComputation: function (diagram, pt)
return new go.Point(Math.floor(pt.x), Math.floor(pt.y));
,
$(
go.TextBlock,
font: "bold small-caps 11pt helvetica, bold arial, sans-serif",
margin: 7,
stroke: "rgba(0, 0, 0, .87)",
editable: true,
,
```

(See figure 16 on page 4). (See figure 17 on page 4).

I have used Panel function to adjust its auto position to create new nodes inside the graph. **Code-**

```
$(
go.Panel,
"Auto",
(go.Shape, "RoundedRectangle",
roundedRectangleParams, fill : null, stroke : "7986cb", strokeWidth : 3,),
(go.Placeholder)
),
$(
"Button",

alignment: go.Spot.TopRight,
click: addNodeAndLink,
,
$(go.Shape, "PlusLine",  width: 6, height: 6 )
)
);
```

(See figure 18 on page 4).

Use addNodeAndLink for inserting a new node to the right of the selected node and add a link to that new node. We use position 200 unite right form the parent node. **Code-**

```
function addNodeAndLink(e, obj){
var adornment = obj.part;
var diagram = e.diagram;
diagram.startTransaction("Add State");
```

(See figure 19 on page 4).

Get the node data for which the user clicked the save button. **Code-**

```
var fromNode = adornment.adornedPart;
var fromData = fromNode.data;
```

Create a new "State" data object, positioned off to the right of the adorned Node. **Code-**

```
var toData =  text: "new" ;
var p = fromNode.location.copy();
p.x += 200;
toData.loc = go.Point.stringify(p);
```

Add the new node data to the model. Code-

```
var model = diagram.model;
model.addNodeData(toData);
```

Create a link data from the old node data to the new node data. **Code-**

```
var linkdata =
from: model.getKeyForNodeData(fromData),
to: model.getKeyForNodeData(toData),
text: "transition",
;
```

(See figure 20 on page 4). (See figure 21 on page 4).

Editing the text automatically updates the model data. **Code-**

```
new go.Binding("text").makeTwoWay()
)
)
);
load();
```

(See figure 22 on page 5).

We can easily DELETE a node using keyboard delete button. First single mouse clicks on the node then delete it.

Show the diagram's model in JSON format. **Code-**

```
function save()
document.getElementById("mySavedModel")
.value = myDiagram.model.toJson();

function load()
myDiagram.model = go.Model.fromJson(
document.getElementById("mySavedModel").value
);
```

(See figure 23 on page 5).
After completing the graph development our next goal was implement the ontology rule into the graph. But unfortunately, my [Mohammad Raihan Sarker Razu] sickness I can not able to work to implement this part.

Challenges and Difficulties-
The first problem we face is when we are going to gather information about our related project we find a few of data. Because there is not much ontology related work. Later we get some papers and web site links with the help of our esteemed factory, from which we collect a lot of information. Then you have to try many libraries to create graphs. The ones I used on May 1, such as Chartist.js or FusionCharts or Google Charts or Chart.js, were static graphs in the form of plots; But we needed a dynamic graph. After 10 days of searching, I found JavaScript's GOJS library. Which will help us to create the desired graph. Later with GOJS library we were able to create graphs for Ontology.
We have also had a lot of environmental problems. Corona Pandemic has had a lot of impact on our group work. Everyone had more or less internet problems when they went to their homes. It's a good time to do things that would have been done in a short time if we had been together. Before the graph was created, I [Mohammad Raihan Sarker Razu] had a fever for about 10 days in early August. After recovering, I completed the work. I [Mohammad Raihan Sarker Razu] have been sick again since September 10th; This time he is suffering from fever, muscle pain and sore throat. My father has been suffering from the same disease since, 5 days after I got sick. When he called the doctor later, he gave him some medicine and forbade him to go out of the house and ask him to rest in bed. I am still ill, unable to fully implement the ontology rules.

5 RESULTS

After a lot of effort and hard work, we have been able to collect a lot of ontology information and create diagrams for our project. By analyzing this diagram, we have divided our project into 5 features. 3 members of our group worked parallel. At the same time, everyone did different things. As a result, work has been done very quickly. From the 1st to the middle of our work was going quite well. After the middle, the speed of our work slowed down due to the increase in the effect of Covid-19. One of our members has been seriously ill twice. As a result, we have not been able to apply for ontology roles. Although the project could not be completed completely, through this project the scope of our knowledge has increased a lot, work skills have increased, communication skills have increased.

Suggestions for improvements-
If anyone in our project wants to make improvements, they can color the nodes according to the group, which will help in better visualization. If you have a better library than GOJS or if you can create a better dynamic graph in some other way, you can apply it. This project can be completed by implementing the rules of ontology.

6 CONCLUSION

As mentioned earlier, we faced some problems in 499A, while finding library's or API's to include protégé into the web platform. Also, there is very little information on the internet about how protégé works. We read some research papers according to our course instructor's advice which helped us a lot [7] [10] [11] [1] [9] [**?**] [3] [5] [2]. We have learned how protégé creates a graph according to our given classes and subclasses. We can also edit, add, and delete the classes by our preferences using the protege. We learned the importance of a stable data graph in various work sectors, like in biomedicine, e-commerce, and organizational modeling. Then in 499B, we started to works on implementing the dynamic graph in the web platform by using HTLM, CSS, and JavaScript (GOJS

library). We have successfully created dynamic graphs. Unfortunately for the illness, we could not apply the ontology rules to the project.

7 ACKNOWLEDGMENT

By the kindness of the Almighty, we have completed our senior design project entitled "Ontology Graphical UI" Our deep gratitude goes first to my faculty advisor Dr. Mohammad Ashrafuzzaman Khan, Assistant Professor of North South University for giving this project idea and having faith in us with this project. Our faculty expertly guided us in our senior design project A throughout the whole CSE499A CSE499B. His guidance helped us in all types of research, writings, and completing the first part of this project. Our sincere thanks also go to North South University, Dhaka, Bangladesh for giving us such a platform where we can have an industrial level experience as a part of our academics. This part of our project is completed by Mohammad Raihan Sarker Razu (1520079042), Tamim Ahmed (1520698642), and Arona Dorin Chowdhry (1520045642). Each has contributed to this project equally by completing their part. Last but not the least, we would like to thank our family as their inspiration and guidance kept us focused and motivated. We learned more about how to share work in groups through this project. How to collect a lot of information to start the work of the project. We also learned how to manage everything in bad times. All in all, we have learned a lot academically, environmentally by doing this project inside this pandemic.

REFERENCES

[1] Kahani M. Abdoli, F. 2009, October. Ontology-based distributed intrusion detection system. In 2009 14th International CSI Computer Conference. (2009, October), 65–70.

[2] Arogundade O. T. Misra S. Akinwale A. T. Adeniran O. J. Abioye, T. E. 2020. Toward ontology-based risk management framework for software projects: An empirical study. *Journal of Software: Evolution and Process* e2269 (2020).

[3] Kumar A. Beniwal R. Malik T. Bhatia, M. P. S. 2020. Ontology driven software development for automatic detection and updation of software requirement specifications. Journal of Discrete Mathematical Sciences and Cryptography,. 1, 23 (2020), 197–208.

[4] John H Gennari, Mark A Musen, Ray W Fergerson, William E Grosso, Monica Crubézy, Henrik Eriksson, Natalya F Noy, and Samson W Tu. 2003. The evolution of Protégé: an environment for knowledge-based systems development. *International Journal of Human-computer studies* 58, 1 (2003), 89–123.

[5] Li T. Zhao H. Liu X. Gao Z. Huang, M. 2020, July. Immune-Based Network Dynamic Risk Control Strategy Knowledge Ontology Construction. In Science and Information Conference. (2020, July), 420–430.

[6] Holger Knublauch. 2004. Ontology-driven software development in the context of the semantic web: An example scenario with Protege/OWL. In *1st International workshop on the model-driven semantic web (MDSW2004)*. Monterey, California, USA.[WWW document] http://www. knublauch. com …, 381–401.

[7] Mark A Musen. 1998. Domain ontologies in software engineering: use of Protege with the EON architecture. Methods of Information in Medicine-Methodik der Information in der Medizin. *AI matters* 4, 37 (1998), 540–550.

[8] Mark A Musen. 2015. The protégé project: a look back and a look forward. *AI matters* 1, 4 (2015), 4–12.

[9] Tudose I. Svatek V. Boeker M. Schober, D. 2012, September. OntoCheck: verifying ontology naming conventions and metadata completeness in Protégé 4. In Journal of biomedical semantics. 3, S2 (2012, September), S4.

[10] Musen M. Silva J. Best C. Ernst N. Fergerson R. Noy N. Storey, M. A. 2001, October. Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé. In Workshop on interactive tools for knowledge capture. *AI matters* 73 (2001, October).

[11] Ding Z. Jiang C. Wang, J. 2005, November. An ontology-based public transport query system. In 2005 First International Conference on Semantics, Knowledge and Grid. 73 (2005, November), 62–62.

[12] Pornpit Wongthongtham, Elizabeth Chang, Tharam S Dillon, and Ian Sommerville. 2005. Software engineering ontologies and their implementation. In *Proceedings of the IASTED International Conference on Software Engineering (SE)*. ACTA Publishing, 208–213.