







# Department of Computer Science & Engineering

UE17CS355 - Web Tech II Laboratory

# **Project Evaluation**

Project Title
Project Team

: Visualization of AI search Algorithms

: PES1201700003 Ashwin R Bharadwaj

PES1201700951 Athreya

PES1201700949 Sahazeer









**Project Description** 

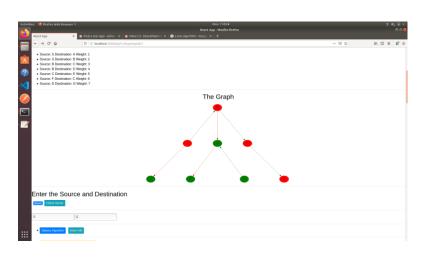
### The app we have developed in this project visualizes graph search algorithms used in Artificial Intelligence

The app consists of a single page where user can construct the graph by filling simple inputs.

As they do so, a visual representation of the graph will be generated to aid in the construction of the graph. Once the user is happy with the graph they have constructed, they can apply search algorithms on the graph by specifying a source and an optional destination, at which point the app will contact the server and retrieve the output of the particular search algorithms the user has requested.

The shortest / optimal path will be visually displayed if they exist.

For most of the algorithms the user also has the option to view each step of the algorithm they selected.



Activiti	es 🕻	Firefox Web Browser	*						22:37					•	Ŧ	41 B +	
<b>(43)</b>		's updated - a: × Rea	rt don	V Mourto	mests v l 🖦 s		Contrates Suit		Mozilla Firefox			in a second contract	□ × Depth First			-000	
		→ C ©		① localhost 3000		nyiconor 1000 in	and books of the second	A STATE OF	Jordy A	Committee of	NO O INC ADMINISTRA	- SHOOPES	© &			0' =	
								_									
181								$\overline{}$									
2								- /	\								
•									•								
<b>P</b>																	
_									7								
×									•								
	Ent	ter the Sour	rco an	d Docting	tion												
<b>Ø</b>	E		ce an	u Destille	шоп												١
	â.		d														
		Difostra Algorithm M	one Info														
		Steps for this algorithm	are unavail	atre													
		Depth First Search Depth First Traversal (c		or a second to a least	s to Brook Flore 1	E	- Parenta										ı
		boolean visited array.	ог зеатспу к	or a graph is sallina	r to Depth First I	Traversal of a tree	e. The only catch her	e is, unake trees,	graphs may contail	n cycles, so we me	ly come to the sam	e node again. To	noid processing a not	e more than on	ce, we u	ise a	ĺ
		See Each Step No Part	th Currently														
	-	Uniform Cost Search	Less Info														
***		See Fach Steel No Dat	h Carrente														









**Technologies Used** 

## The following tools have been used to

- React JS (Used for the front end)
- Node JS (Used as the server that handles the files and simple algorithms)
- Flask (Used to implement more CPU intensive algorithms)





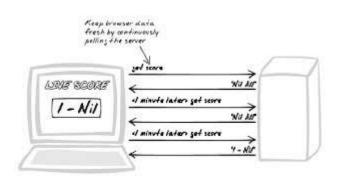




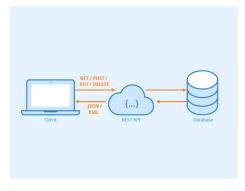


## **Techniques Implemented**

- The project has over 15 different RESTful APIs implemented.
- Some of these APIs are used to retrieve data processed by the search algorithms and others are required to retrieve the meta data required to display the graphs and others needed to retrieve data from files.
- There are some AJAX methods that have also been implemented such as predictive fetch where based on the algorithm the user chooses the client requests more information about that particular algorithm.
- Periodic refresh has also been implemented, where the client continuously sends data to the server and waits for a response. If there is no response the client assumes the server is unavailable and alerts the user.



Periodic Refresh



**RESTful APIs** 









Intelligent Functionality

- Visually displaying the graph entered by the user.
- Providing information about the Algorithms used to search for goals in a graph.
- Showing step by step visually how each of the algorithms work.
- Allows the user to dynamically change the graph and re run the algorithms.

#### The algorithms implemented are:

- A\* Algorithm
- Uniform Cost Search
- Dijkstra's algorithm
- DFS
- BFS

We are planning to add a few more algorithm and display the steps in a more visually pleasing manner. Once the above are done we plan on deploying this so that students who have take AI course can benefit from this tool and learn the algorithms more easily.

We hope that we can rely on your support to do so

#### **Libraries Used:**

- We have implemented our own libraries to represent the graphs.
- We have modified the "vis" library available for java script to run on React JS.
- We also have implemented libraries to represent and compute queues in NodeJS.











# Thank You