**INT404 Project**

**(SIMPLE NAVIGATOR FOR LPU)**

**Section: K18LC**

|  |  |  |
| --- | --- | --- |
| Team Members Detail: | | |
| VAISHALI | 11808811 | 03 |
| ROOPAK PARASHAR | 11808599 | 05 |
| MANIDEEP ALLA | 11808651 | 09 |

**Git Hub File Link:**

**Introduction**

Navigation is a field of study that focuses on the process of monitoring and controlling the movement of a craft or vehicle from one place to another.The field of navigation includes four general categories: land navigation, marine navigation, aeronautic navigation, and space navigation. It is also the [term of art](https://en.wikipedia.org/wiki/Term_of_art" \o "Term of art)  used for the specialized knowledge used by [navigators](https://en.wikipedia.org/wiki/Navigator" \o "Navigator) to perform navigation tasks.All navigational techniques involve locating the navigator's position compared to known locations or patterns.Navigation, in a broader sense, can refer to any skill or study that involves the determination of position and direction. In this sense, navigation includes [orienteering](https://en.wikipedia.org/wiki/Orienteering" \o "Orienteering) and pedestrian navigation.

**What Is this lpu navigator?**

This code is useful in finding shortest ways within campus as well as it gives the list of places you will come across, in your way to your destination.

In this code we have used python language and some concept of A\* algorithm

**Concept used:-**

A\* ALGORITHM:-

A\* is a graph traversal and path search algorithm, which is often used in computer science due to its completeness , optimality, and optimal efficiency. one major practical drawback is its space complexity, as it stores all generated nodes in memory.

Worst case complexity:- O(|E|) = O(b^d)

space complexity:- O(|V|) = O(b^d)

**METHODOLOGY**

1. **) We have used A\* algorithm to use the concept of shortest and more economic path :**

Informally speaking, A\* Search algorithms, unlike other traversal techniques, it has “brains”. What it means is that it is really a smart algorithm which separates it from the other conventional algorithms.  
And it is also worth mentioning that many games and web-based maps use this algorithm to find the shortest path very efficiently (approximation).

**B.) Some of the packages used in the code:**

**1.)numpy:** NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various

features including these important ones:

**·** A powerful N-dimensional array objects

**·** Sophisticated (broadcasting) functions

**·**  Tools for integrating C/C++ and Fortran code

· Useful linear algebra, Fourier transform, and random number capabilities

**2.)Defaultdict:** Itis a container like [dictionaries](https://www.geeksforgeeks.org/python-dictionary/" \t "https://www.geeksforgeeks.org/defaultdict-in-python/_blank) present in the module **collections**. Defaultdict is a sub-class of the dict class that returns a dictionary-like object. The functionality of both dictionaries and defualtdict are almost same except for the fact that defualtdict never raises a KeyError. It provides a default value for the key that does not exists.

**Syntax:** defaultdict(default\_factory)

**default\_factory:** A function returning the default value for the dictionary defined. If this argument is absent then the dictionary raises a KeyError.

***Code(Python)***

*print("\n\n\t\t\t\t Hello, Welcome to the Lovely Professional University Navigation system\n\n\n");*

*import numpy*

*from collections import defaultdict*

*class NavigationSystem():*

*def \_\_init\_\_(self):*

*self.routes = defaultdict(list)*

*self.weights = {}*

*def add\_route(self, source, your\_destination, weight):*

*self.routes[source].append(your\_destination)*

*self.routes[your\_destination].append(source)*

*self.weights[(source, your\_destination)] = weight*

*self.weights[(your\_destination, source)] = weight*

*NavSystem = NavigationSystem()*

*routes = [*

*('main\_gate', 'Baldevraj\_park', 2),*

*('main\_gate', 'Uni\_hospital', 3),*

*('main\_gate', 'cc\_cafe', 6),*

*('Baldevraj\_park', 'pharmacy\_block', 2),*

*('pharmacy\_block', 'Baldevraj\_hall', 4),*

*('Uni\_hospital', 'Animation\_block', 3),*

*('Uni\_hospital', 'GH-2,3,4', 5),*

*('Animation\_block', 'GH-2,3,4', 2),*

*('GH-2,3,4', 'Uni\_mall', 2),*

*('Uni\_mall', 'unipolis', 7),*

*('unipolis', 'Second\_gate', 4),*

*('unipolis', 'Dsw', 1),*

*('unipolis', 'business\_block', 7),*

*('unipolis', 'Health\_center', 7),*

*('cc\_cafe', 'business\_block', 2),*

*('cc\_cafe', 'Baldevraj\_hall', 2),*

*('Baldevraj\_hall', 'cc\_foodcourt', 2),*

*('business\_block', 'Baldevraj\_hall', 1),*

*('business\_block', 'cc\_foodcourt', 4),*

*('cc\_foodcourt', 'physical\_education', 1),*

*('cc\_foodcourt', 'hexi\_stand', 4),*

*('hexi\_stand', 'Admission\_block', 3),*

*('hexi\_stand', 'Admission\_block', 2),*

*('hexi\_stand', 'Shantidevi\_auditorium', 4),*

*('Physical\_Education', 'microscope\_park', 2),*

*('Physical\_Education', 'microscope\_park', 2),*

*('microscope\_park', 'Petrochemical\_Engineering', 3),*

*('Petrochemical\_Engineering', 'bioscience\_block', 2),*

*('bioscience\_block', 'computer\_science', 5),*

*('bioscience\_block', 'health\_block', 5),*

*('computer\_science', 'Shantidevi\_auditorium', 3),*

*('Admission\_block', 'boys\_hostel\_123', 3),*

*('Admission\_block', 'Law', 1),*

*('Admission\_block', 'Shantidevi\_auditorium', 3),*

*('Law', 'Main\_library', 1),*

*('boys\_hostel\_123', 'Main\_library', 1),*

*('Main\_library', 'boys\_foodcourt', 3),*

*('Shantidevi\_auditorium', 'Main\_library', 2),*

*('Shantidevi\_auditorium', 'boys\_foodcourt', 5),*

*]*

*for route in routes:*

*NavSystem.add\_route(\*route)*

*def navigate(graph, beginning, dest):*

*shortest\_routes = {beginning: (None, 0)}*

*current\_vertex = beginning*

*visited\_vertex = set()*

*while current\_vertex != dest:*

*visited\_vertex.add(current\_vertex)*

*destinations = graph.routes[current\_vertex]*

*weight\_to\_current\_vertex = shortest\_routes[current\_vertex][1]*

*for next\_vertex in destinations:*

*weight = graph.weights[(current\_vertex, next\_vertex)] + weight\_to\_current\_vertex*

*if next\_vertex not in shortest\_routes:*

*shortest\_routes[next\_vertex] = (current\_vertex, weight)*

*else:*

*current\_shortest\_weight = shortest\_routes[next\_vertex][1]*

*if current\_shortest\_weight > weight:*

*shortest\_routes[next\_vertex] = (current\_vertex, weight)*

*next\_destinations = {vertex: shortest\_routes[vertex] for vertex in shortest\_routes if vertex not in visited\_vertex}*

*if not next\_destinations:*

*return "Sorry, Route Not Possible Within Unilag Or Please Check Your Spellings"*

*current\_vertex = min(next\_destinations, key=lambda k: next\_destinations[k][1])*

*shortest\_path = []*

*while current\_vertex is not None:*

*shortest\_path.append(current\_vertex)*

*next\_vertex = shortest\_routes[current\_vertex][0]*

*current\_vertex = next\_vertex*

*shortest\_path = shortest\_path[::-1]*

*return shortest\_path*

*CurrentLocation = input("Input your current location: ")*

*print("\n\t\t\t\tYour source is " + CurrentLocation)*

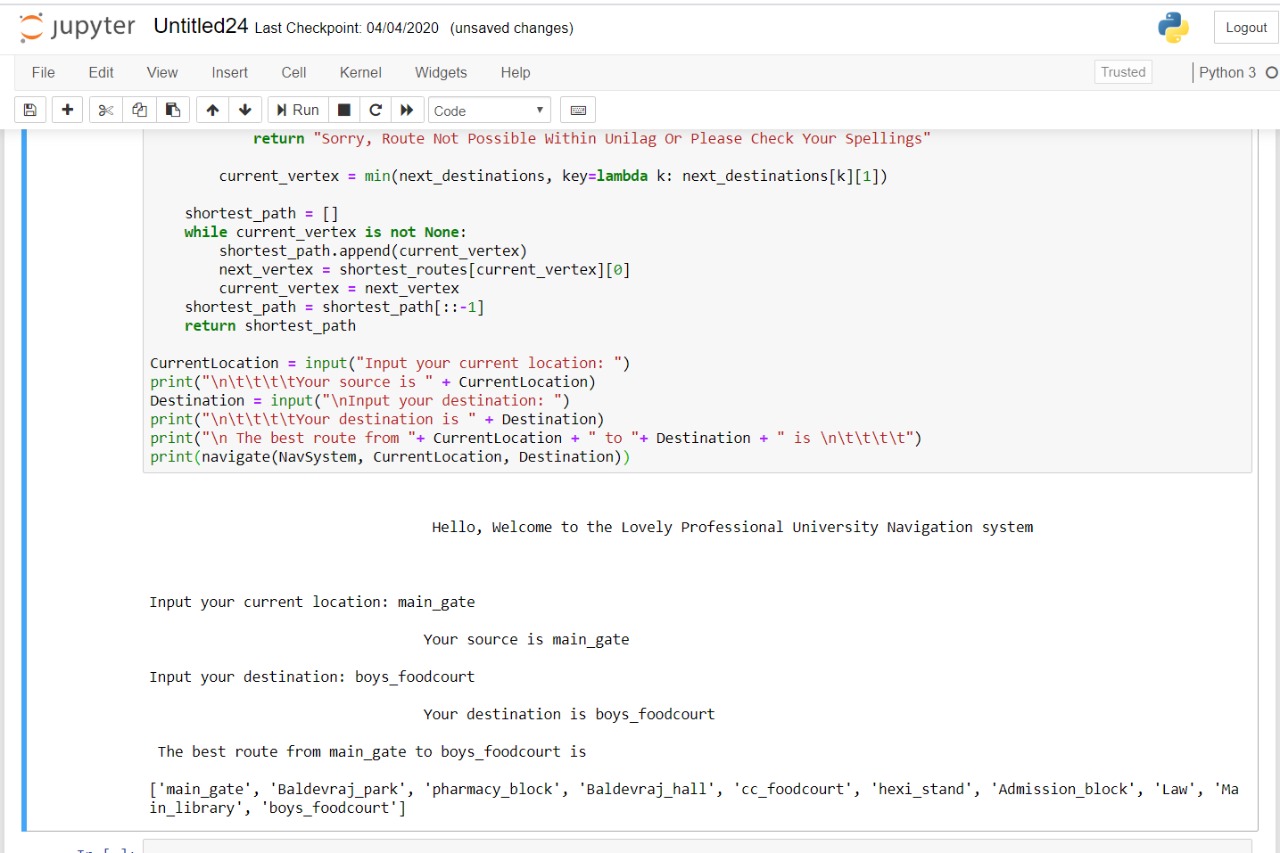
*Destination = input("\nInput your destination: ")*

*print("\n\t\t\t\tYour destination is " + Destination)*

*print("\n The best route from "+ CurrentLocation + " to "+ Destination + " is \n\t\t\t\t")*

*print(navigate(NavSystem, CurrentLocation, Destination))*

***Output:-Screenshot of the output when the source is main gate and destination is boys hostel food court***

******

**Literature Review & Result**

**Future Scope**

In the Present Scenario only we are merely depends on Google Maps to Travel from one Location to Another , the GMaps Algorithm updating and changing itself for every minute and hour depends upon the user inputs and the Conditions.

So In future means in the Upcoming Years Maps and Navigators will Play a key role in the life of Huma Beings .

This Project helped us to learn the Basics in Navigation System like GMaps(Package),API’s etc. It will definitely helps us in Future for further Purposes.

**Constarints**

**1.)**This program is limited to the locations based in LPU and cannot be used for any other location.

**2.)**There may be a need of optimized internet access to tap into the full resources of the program.

**CONCLUSION**

Numerous cognitive studies have indicated that the form and complexity of route navigation may be as important to human navigators as the overall length of route. Almost without exception, automated navigation systems rely on computing the solution to the shortest path problem, and not the problem of finding the “simplest” path.

This report addresses the issue of finding the “simplest” paths through a network, in terms of their ease of description. We propose a simplest path program that is computationally as efficient and provides an optimized solution

**References:Internet geeks for geeks**

|  |  |
| --- | --- |
| **Team Efforts:**  Name | Work Done |
| Roopak Parashar(5) | * Web Research * Prepared the detailed report * Extracted the Concepts to be used |
| Vaishali (3) | * Implemented the concepts * Wrote the code * Helped in finding material related to project |
| Manideep (9) | * Supported in all the works * Finding the shortest routes that are possible   Within the campus |