**MINOR-1 PROJECT**

**SYNOPSIS on**

Travel Route Chat-bot

Submitted By:

|  |  |  |
| --- | --- | --- |
| **Name** | **Roll No** | **Branch** |
| Dishant Saini | R2142220485 | BTech CSE AIML |
| Palak Jain | R2142220583 | BTech CSE AIML |
| Sahil | R2142220622 | BTech CSE AIML |
| Akshat Rajput | R2142220209 | BTech CSE AIML |

**Under the guidance of**

Dr. Bhavana Kaushik

(Assistant professor at School of Computer Science)

A picture containing text, clipart

Description automatically generated

**School of Computer Science**

**University of Petroleum & Energy Studies, Dehradun**

**Dehradun-248007**

**Aug-Nov 2024**

**Approved By**

**Project Guide Cluster Head**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S NO.** | **TOPIC** | **PAGE NO.** |
| 1 | Introduction | 1 |
| 2 | Problem Statement | 2 |
| 3 | Literature Review | 3 |
| 4 | Objective | 4 |
| 5 | Methodology | 5 |
| 6 | Timeline | 6 |
| 7 | System Requirement | 7 |
| 8 | References | 8 |

# Introduction

One of the most complex issues that people face in their daily lives is deciding which direction to take at a certain moment. For example, people tend to aimlessly roam around in university campuses trying to understand their surroundings, though not always does the answer lie in an ideal way. The shortest path problem is the search for the least time taken, or the least cost incurred in traveling from one point to another in the network. In this case, we are interested in a “Chatbot” device programmed to assist the users to move within the campus by walking them from one point to the other. The navigation system on the other hand starts from the user’s location and recommends the shortest time to reach the destination.

In a graph, every edge is associated with a length and the question is to find the least length path among the edges connecting the two nodes maybe. In order to define a common margin, earn to this shortest path problem, we define distance in terms of Haversine formula, which in this case is used to measure the distance between two geographical coordinates in the spherical geography that we’re dealing with. This formula, therefore, becomes important in geographical navigation of the earth since the facts concerning the earth being round is a factor.

Different types of the shortest path problem appear, such as single-source, pair-wise, and all-pairs shortest paths. In pair-wise shortest paths, members consider two particular nodes and compute the shortest distance. In all-pairs shortest paths, the distance for all possible pairs of nodes is calculated. The single-source shortest path problem deals with the minimum distance to one particular node from all the other nodes in the graph. Instead of using traditional algorithms like Dijkstra’s, in this method, we use the Haversine formula while working on the graphs, wherein the graph is a two-dimensional representation of professional identification documents. When spatial distances are prioritized over edge weights in the optimization, this method comes in handy since it gives a very realistic measurement of actual distances.

# Problem Statement

Most of the advanced navigation systems are unable to provide precise routes as well as information of building within a small region such as a hospital, university campus or shopping mall. As people get more and more connected to technology, they lose their human touch. People feel more convenient to look for the problem themselves than to ask someone for help. This technological era requires a reliable and precise guidance system. It should be able to navigate the user no matter where they are in the world. The guidance system needs to be easy to use. Students spend most of their time travelling between different buildings on a college campus since the size of the campus can vary from 30 acres to 200 acres. New students have a hard time searching their way inside the campus.

To find the optimal path within the campus travel guide chatbot is required.

# Literature Review

|  |  |  |
| --- | --- | --- |
| Reference​ | Methods​ | Year of Research​ |
| [1] | Application of Dijkstra's algorithm for railway route optimization in transportation systems​ | 2014​ |
| [2]​ | Dijkstra's algorithm combined with node combination for shortest pathfinding in Geographical Information Systems (GIS)​ | 2018​ |
| [3] ​ | Integration of Dijkstra’s algorithm with the Haversine formula for landmark-based shortest path detection using real-world coordinates​ | 2013​ |
| [4]​ | Development of a web-based chatbot management system for creating chatbots without programming knowledge​ | 2021​ |
| [5]​ | Review and evaluation of various shortest path algorithms, including Dijkstra’s algorithm, A\*, and others​ | 2013​ |

# Objectives

The program aims to develop a chatbot that simplifies campus navigation by providing users with the shortest and most efficient routes. Using Dijkstra’s algorithm, the chatbot ensures accurate and efficient route calculations. Designed for students, staff, and visitors, it enhances user experience with a responsive interface that offers real-time, clear directions. The scalable design allows adaptation to other campuses, and its seamless integration with web technologies ensures accessibility across various devices, making campus navigation quicker and more convenient.

* To build a responsive chatbot interface that provides real- time, accurate direction for campus navigation.
* To develop an algorithm framework using Dijkstra’s algorithm for efficient and precise route calculations.
* To create a user-friendly system designed specifically for students, staff and visitors to reduce time spent navigating the campus.
* To ensure scalability of the program for potential adaptation to other campuses or similar environments.

# Methodology

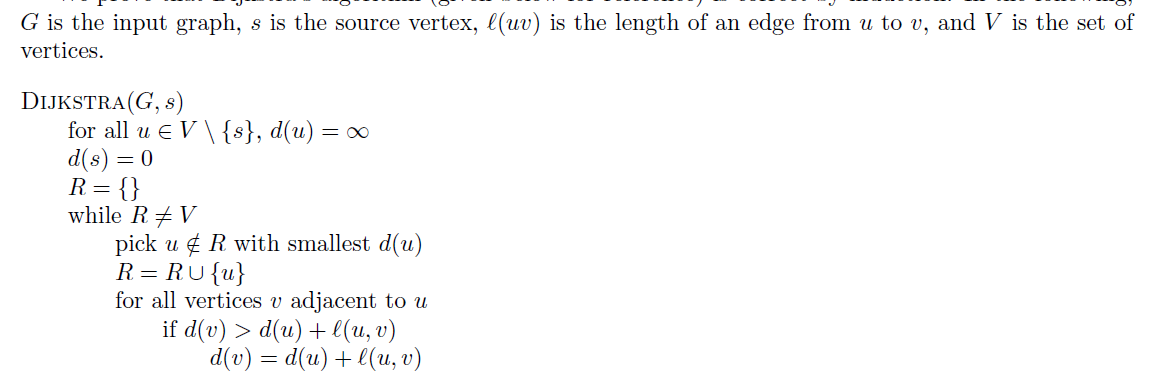
Dijkstra’s algorithm is for minimum spanning tree. In Dijkstra’s we generate a*SPT (shortest path tree)* with given source as root. We maintain two sets, one set contains vertices included in shortest path tree, and other set includes vertices not yet included in shortest path tree. At every step of the algorithm, we find a vertex which is in the other set (set of not yet included) and has a minimum distance from the source.

The Dijkstra’s algorithm maintains two sets of vertices which are:

*L*:Labeled vertices (shortest path is known)

*C*: Candidate vertices (shortest path not yet known)

One vertex is moved from *C* to *L* in each iteration



The steps to be followed for completing the proposed project are as follows:

1. **Identification of Key Blocks and Coordinates in Campus:** key areas (blocks) are identified on campus and their geographic coordinates (latitude and longitude) are collected. These positions are the nodes of the graph used in the Dijkstra algorithm. Nodes represent blocks such as classrooms, libraries, hotels, or other important landmarks.
2. **Distance and Direction Calculation Between Nodes:** Distances to neighboring nodes are determined using geographic information (Haversine formula for real-world coordinates) or manually calculated distances Furthermore, distances to neighboring nodes are determined using directional data (e.g. "left", "right", "straight"). (blocks) between the connected nodes. This data is a weighted graph where each edge represents the distance and direction between two blocks.
3. **Finding the Shortest Path and Providing Step-by-Step Directions:** Using Dijkstra’s Algorithm, the shortest distance between the user’s start block and destination block is calculated. The program will not only provide the user with the shortest route but also provide the user with step-by-step navigation instructions (e.g. turn left, go straight) and intermediate distances between the two, and help them navigate their environment better.

# System Requirement

**Software Requirements**

Operating System : Windows 11/10 (32-bit or 64-bit)/ Linux

Software : Text Editor/Turbo C/Dev C++

Compiler : GCC

**Hardware Requirements**

Processor : Dual Core 2.7 GHz or better

RAM : 2GB or higher

# **Timeline**

|  |  |  |
| --- | --- | --- |
| Task | Timeframe | Description |
| Project Start | 20-21 August | Official start of the project |
| Requirements Analysis (T1) | Last Week of August | Analyzing requirements for the project |
| Estimate Distance (T2) | Last Week of August | Estimating distances between campus infrastructure |
| Chatbot Design and Front-End Development (T3) | Mid-Semester | Designing the chatbot interface using HTML and CSS |
| Algorithm Implementation/Coding (T4) | Last Week of September | Developing algorithms and integrating it into the chatbot system |
| Front-End & Algorithm Integration (T5) | Last Week of October | Integrating the Chatbot Interface with Dijkstra’s algorithms |
| Final Testing (T6) | Mid-November | Testing the integrated chatbot system |
| Project End | End of November | Project conclusion |

# References

1. Pandey, Pramod, and Sunanda Dixit. "Railway Route Optimization System Using Dijkstra Method." *International Journal on Recent and Innovation Trends in Computing and Communication (IJRITCC) Volume* 2 (2014): 435-440.
2. Fitro, Achmad, et al. "Shortest path finding in geographical information systems using node combination and dijkstra algorithm." *SHORTEST PATH FINDING IN GEOGRAPHICAL INFORMATION SYSTEMS USING NODE COMBINATION AND DIJKSTRA ALGORITHM* 9.2 (2018): 755-760.
3. Ingole, P. V., and Mangesh K. Nichat. "Landmark based shortest path detection by using Dijkestra Algorithm and Haversine Formula." *International Journal of Engineering Research and Applications (IJERA)* 3.3 (2013): 162-165.
4. Hasyim, M. W., and S. Pramono. "Web-based telegram chatbot management system: create chatbot without programming language requirements." *IOP Conference Series: Materials Science and Engineering*. Vol. 1096. No. 1. IOP Publishing, 2021.
5. Magzhan, Kairanbay, and Hajar Mat Jani. "A review and evaluations of shortest path algorithms." *Int. J. Sci. Technol. Res* 2.6 (2013): 99-104.