**Project Name:** Optimal Path Finder

**Github Link:** https://github.com/projectsforstudents2022/Optimal\_Path\_Finder.git

**Why was this project created?**

A supporting system called an artificial intelligence-driven assistant system is used to locate a path inside a strictly defined area and can also be utilised for automation. Numerous practical problems, including chip design, network reliability, transit planning, and cluster analysis, can be resolved using connected graphs.

**What problem is it solving?**

The Dijkstra's Algorithm can be used to solve optimal path problems by finding the shortest routes. The implementation's minimal cost is the major goal. Finding a path between two vertices on a given graph such that the sum of the weights on its individual edges is reduced. Due to the numerous applicability of this issue in graph theory, artificial intelligence, computer networks, and the design of transportation systems, it has received much research over the years.

**Entire explanation of project**

* **PROPOSED APPROACH**

Let's call the node from which we are starting the initial node. The Dijkstra algorithm will assign some initial distance values and attempt to incrementally increase them. The current intersection will serve as the starting point and have a distance to it of zero for the initial iteration. The present intersection will, for following iterations, be the nearest unexplored intersection to the beginning point; it will be simple to locate. Update the distance from the present intersection to every unvisited intersection that is directly connected to it. This is accomplished by adding the distance between an intersection that hasn't been visited and its current value, and relabeling the intersection if the difference is greater than the intersection's present value.

If the path across the present intersection is shorter than the previously known paths, the intersection is effectively renamed. If you label or relabel the intersection, mark the route with an arrow pointing there in pencil and cross out all others to make it easier to identify the quickest path. Mark the current intersection as visited and choose the unvisited intersection with the lowest distance or lowest label as the current intersection once you have updated the distances to each neighbouring intersection. Nodes that have been marked as visited are identified by the shortest route from the starting point to them and won't be gone back to.

Algorithm for creating next word prediction model :

function dijkstra(G, S)

for each vertex V in G

distance[V] <- infinite

previous[V] <- NULL

If V != S, add V to Priority Queue Q

distance[S] <- 0

while Q IS NOT EMPTY

U <- Extract MIN from Q

for each unvisited neighbor V of U

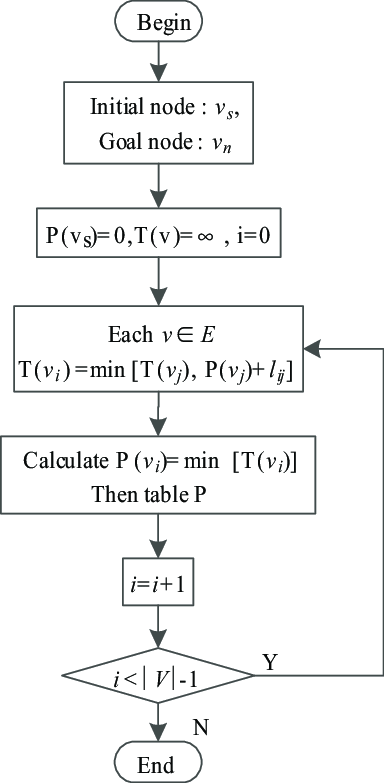
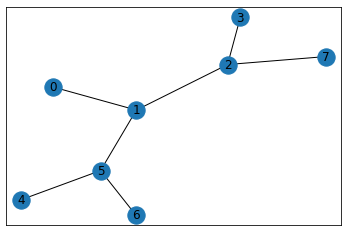
tempDistance <- distance[U] + edge\_weight(U, V)

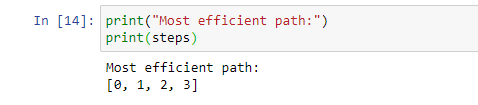
if tempDistance < distance[V]

distance[V] <- tempDistance

previous[V] <- U

return distance[], previous[]

* **DATA FLOW DIAGRAM**
* **RESULT**



* **CONCLUSION**

For the challenge of finding the shortest path in transportation networks, we have suggested a workable algorithm. Based on the provided nodes and the distance between the two nodes, the suggested approach can restrict the search in a sub-graph. The calculation for the shortest path has been streamlined as a result. In comparison to previous efforts, experimental findings on a real-world road network show promising characteristics of the proposed approach.