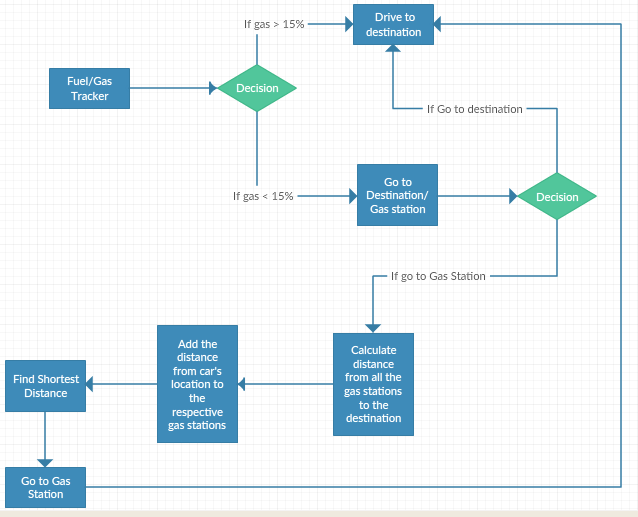
Requirements:

1. System keeps track of the available gas in the tank.
2. A map of where the gas stations are located is known.
3. If the gas in the tank goes below 15% the vehicle checks if the destination can be reached with the amount of gas left in the tank. (decision made by the user to go to the destination or fill gas).
4. If the user chooses to fill the gas, the system checks if there is any gas station on route to the destination. If there is, it chooses that if it can get there with the amount of gas left.
5. If there are no gas stations on route, it searches for all the gas stations in the area and calculates the distance from the vehicle to each one of them.
6. The system then checks for the minimum distance from the individual gas station to the destination. The shortest route is the selected.

Domain Analysis:

Our main goal is to tackle the problem of shortage of gas in the tank. The system will alert the user if the gas in the tank falls below the threshold and allow the user to choose if it wants to go to the destination or the gas station. For this to happen, the exact location coordinates of the gas stations in the map has to be known. A fuel tracker that constantly checks if the current fuel in the tank. We constantly compare the current fuel with the fuel in the tank, and if it goes below 15% the user gets to choose if he/she would like to go to the destination directly or to the nearest gas station. This option will be displayed on the display screen in the cat vehicle. If the user chooses to go to the destination without refueling the tank, it displays the maximum point up to which it can go with the left over gas. If the user chooses to go to the nearest gas station, our system gets the location coordinates of the nearest gas stations. Then, it checks if there are any gas stations on route to the destination. If there are any gas stations in the same route and if it can be reached with the amount of gas left in the tank, it chooses that one. If there are not any, then the total distance from the current position of the vehicle, to the destination via all the gas stations are calculated. The shortest distance among all of them is then chosen. The algorithm for path optimization which is currently very popular is the A\* algorithm.



Algorithms:

For our system, we have decided to use the popular A\* algorithm to find the shortest path from the current location of the car to the destination via the gas station.

**A\*** is a computer algorithm that is widely used in pathfinding and graph traversal, the process of plotting an efficiently directed path between multiple points, called nodes. It enjoys widespread use due to its performance and accuracy. However, in practical travel-routing systems, it is generally outperformed by algorithms which can pre-process the graph to attain better performance, although other work has found A\* to be superior to other approaches It is an extension of Edsger Dijkstra's 1959 algorithm. A\* achieves better performance by using heuristics to guide its search.

A\* is a best-first search, meaning that it solves problems by searching among all possible paths to the goal for the one that incurs the smallest cost (least distance travelled, shortest time, etc.), and among these paths it first considers the ones that *appear* to lead most quickly to the solution. It is formulated in terms of weighted graphs: starting from a specific node of a graph, it constructs a tree of paths starting from that node, expanding paths one step at a time, until one of its paths ends at the predetermined goal node.

At each iteration of its main loop, A\* needs to determine which of its partial paths to expand into one or more longer paths. It does so on an estimate of the cost (total weight) still to go to the goal node. Specifically, A\* selects the path that minimizes

F(n) = g(n) + h(n)

f ( n ) = g ( n ) + h ( n ) {\displaystyle f(n)=g(n)+h(n)} where *n* is the last node on the path, *g*(*n*) is the cost of the path from the start node to *n*, and *h*(*n*) is a heuristic that estimates the cost of the cheapest path from *n* to the goal. The heuristic is problem-specific.For our project we have decided to take the heuristic as the straight line distance to the goal.

Pseudo Code:

**OPEN:** consists on nodes that have been visited but not expanded (meaning that successors have not beenexplored yet). This is the list of pending tasks.

**CLOSE:** consists on nodes that have been visited andexpanded (successors have been explored already andincluded in the open list, if this was the case)

|  |  |
| --- | --- |
|  |  |