**Final Report**

**CAB BOOKING SYSTEM**

**ALGORITHM DESIGN AND ANALYSIS**

****

**Submitted by: -**

**ANSH KAKUSTH (2K19/IT/024)**

**ABHINAV GUPTA (2K19/IT/002)**

**SUBMITTED TO:-**

**RAHUL GUPTA**

**ACKNOWLEDGEMENT**

I ANSH KAKUSTH (2K19/IT/024) would like to express my special thanks of gratitude to RAHUL GUPTA as well as to DTU who gave me this opportunity to do this project on the topic **Cab Booking System** which has also helped me in doing a lot of research and I came to know about so many new things I am really thankful to them. Secondly, I would like to thank my group partner ABHINAV GUPTA(2K19/IT/002) and my friends who helped me a lot in finishing this project within the limited time.

ANSH KAKUSTH ABHINAV GUPTA

2K19/IT/024 2K19/IT/002

**INTRODUCTION**

India- A country known for lowest car ownership across all emerging countries found a new answer to its mobility needs in 2013, when Taxi for sure launched its app based on-demand taxi program. The very same year, when Uber entered India and launched its services in Bangalore.

Till 2014, there were very few takers for on-demand taxis but a lot changed in 2015 when Ola rolled out a much improved and accurate app with broader coverage. Till 2014 and even early 2015, Ola`s ~20% bookings were made through laptops and desktops. India`s weak internet infrastructure and comparatively lower smartphone penetration was the prime reason behind this.

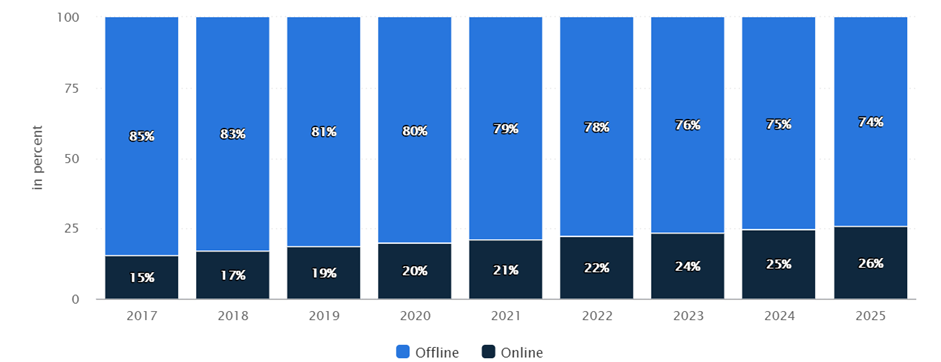
The smartphone penetration was 6% in India in 2013 and reached 28% by May 2019. In metro cities, the penetration reached 60%-65% by Q1-2015 and that is when and where the revolution of on-demand taxis began in India.

In [India](https://en.wikipedia.org/wiki/India), most taxicabs, especially those in [Delhi](https://en.wikipedia.org/wiki/Delhi) and [Mumbai](https://en.wikipedia.org/wiki/Mumbai), have distinctive black and yellow liveries with the bottom half painted black and upper half painted yellow. In [Kolkata](https://en.wikipedia.org/wiki/Kolkata), most taxis are painted yellow with a blue strip in the middle. Private companies operating taxis can have their own liveries but need to get them approved from the government. Now drivers don’t need to pay a commission. [[1]](https://en.wikipedia.org/wiki/Taxis_in_India#cite_note-1)

Taxis and all other commercial vehicles have a yellow number plate so charging taxes and toll in highways is easier for the officials. [Delhi](https://en.wikipedia.org/wiki/Delhi) is the only city in India with taxicabs running only on [Compressed Natural Gas](https://en.wikipedia.org/wiki/Compressed_Natural_Gas). To hail a taxicab, you normally just wait on the street or go to locations called taxi stands. Taxicabs are referred to as taxis in India and the word cab is rarely used.

Most of these cabs do not have an air conditioning system but there are numerous private taxi operators like "Cool Cab" are air-conditioning. The newer taxis are white, one of the many reasons why the expensive taxis have been dubbed White Taxis by the locals in cities. Taxis and all other commercial vehicles have a yellow number plate so charging taxes and toll in highways is easier for the officials.

Depending on the city/state, taxis can either be hailed or are hired from taxi-stands. In cities such as [Bangalore](https://en.wikipedia.org/wiki/Bangalore), taxis need to be hired from taxi stands, whereas in cities like [Kolkata](https://en.wikipedia.org/wiki/Kolkata) and [Mumbai](https://en.wikipedia.org/wiki/Mumbai), taxis can be hailed on the street. There are additional surcharges for luggage, late-night rides and toll taxes are to be paid by the passenger. Thanks to the booming economy but due to disparities in income many types of taxis have come up.

****

**Objective:**

To make a program that will help users to book a cab of their own convenience.

**Implementation of Program**

In this program the user will first sign up and will give his/her details (Name, mobile number, mail etc.).

Then the user will be asked to enter location and his/her desired destination.

Program will provide users with a number of cabs based on their distance and type and will also display their cost.

* Program will also show the minimum path or distance from the user's location to desired destination.
* In end mode of payment and reviews\ratings will be asked

from the user.

**Algorithm**

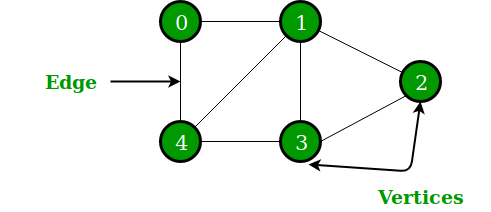
* Firstly, the project layout.
* Asking the user whether the user wanted to book a cab, if yes then we welcome him or else print thanks and exit.
* Then we display sign up.
* Then the program gets details of the user such as mobile no, email id.
* Then we ask the person 's location and its destination.
* Then we display cab charges according to cab size and luxury (like car brand) which are some distance away from the user.
* Then we ask whether the user wants to book a cab or not, if yes then further else exit.
* Then we print the best route.
* later we ask for payment mode whether he wants to pay cash or online payment.
* At the end we display thanks and ask for a review out of 5 stars.

**Graphs**

A Graph is a non-linear data structure consisting of nodes and edges. The nodes are sometimes also referred to as vertices and the edges are lines or arcs that connect any two nodes in the graph.

More formally a Graph can be defined as,

A Graph consists of a finite set of vertices (or nodes) and a set of edges which connect a pair of nodes.



Graphs are used to represent many real-life applications: Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network. Graphs are also used in social networks like LinkedIn, Facebook. For example, in Facebook, each person is represented with a vertex (or node). Each node is a structure and contains information like person id, name, gender, and locale.

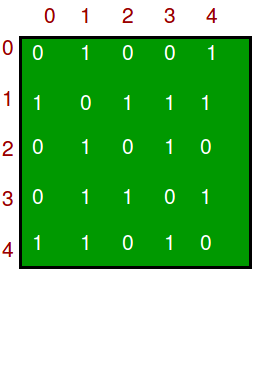
The following two are the most commonly used representations of a graph.

**1.** Adjacency Matrix

**2.** Adjacency List

**Adjacency Matrix:**

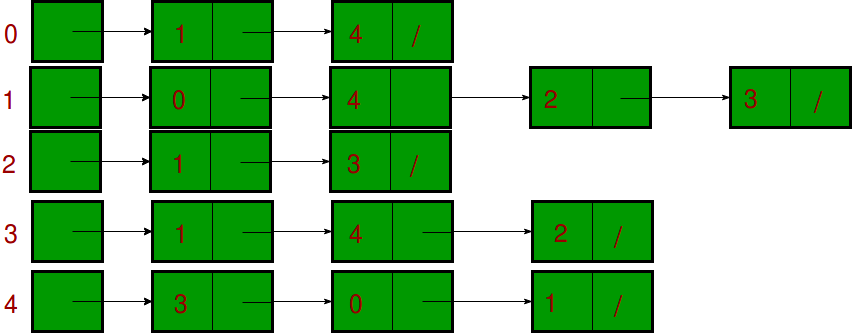
Adjacency Matrix is a 2D array of size V x V where V is the number of vertices in a graph. Let the 2D array be adj[][], a slot adj[i][j] = 1 indicates that there is an edge from vertex i to vertex j. Adjacency matrix for an undirected graph is always symmetric. Adjacency Matrix is also used to represent weighted graphs. If adj[i][j] = w, then there is an edge from vertex i to vertex j with weight w.



Adjacency Matrix for above graph

**Adjacency List:**

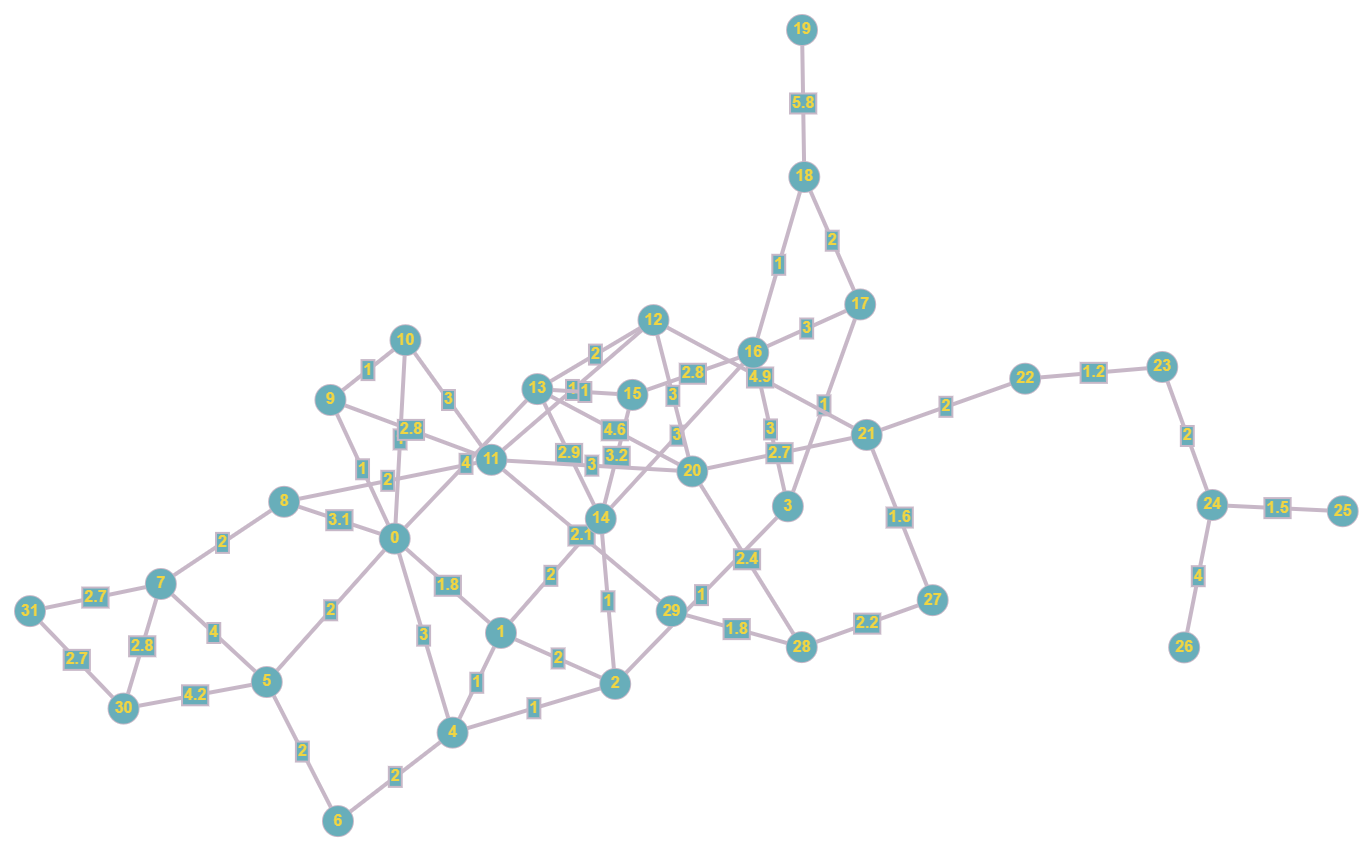
An array of lists is used. The size of the array is equal to the number of vertices. Let the array be an array[]. An entry array[i] represents the list of vertices adjacent to the **i**th vertex. This representation can also be used to represent a weighted graph. The weights of edges can be represented as lists of pairs.



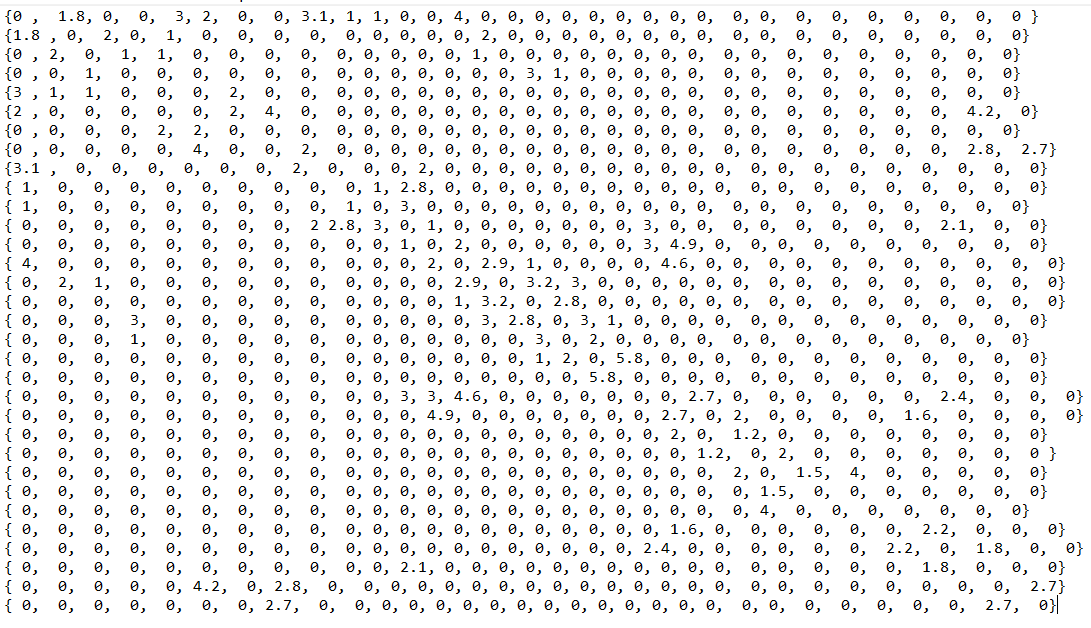
Adjacency List for above graph

We have used adjacency matrix in our project

Our Map:



Our Adjacency Matrix:



**Dijkstra Algorithm**

We have used the Dijkstra algorithm in our project to find the shortest path between two nodes for our graph.

Dijkstra's algorithm (or Dijkstra's Shortest Path First algorithm, SPF algorithm)[[4]](https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#cite_note-4) is an [algorithm](https://en.wikipedia.org/wiki/Algorithm) for finding the [shortest paths](https://en.wikipedia.org/wiki/Shortest_path_problem) between [nodes](https://en.wikipedia.org/wiki/Vertex_(graph_theory)) in a [graph](https://en.wikipedia.org/wiki/Graph_(abstract_data_type)), which may represent, for example, [road networks](https://en.wikipedia.org/wiki/Road_network). It was conceived by [computer scientist](https://en.wikipedia.org/wiki/Computer_scientist) [Edsger W. Dijkstra](https://en.wikipedia.org/wiki/Edsger_W._Dijkstra) in 1956 and published three years later.

The algorithm exists in many variants. Dijkstra's original algorithm found the shortest path between two given nodes, but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a [shortest-path tree](https://en.wikipedia.org/wiki/Shortest-path_tree).

For a given source node in the graph, the algorithm finds the shortest path between that node and every other.[[8]](https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#cite_note-mehlhorn-8):196–206 It can also be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined. For example, if the nodes of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road (for simplicity, ignore red lights, stop signs, toll roads and other obstructions), Dijkstra's algorithm can be used to find the shortest route between one city and all other cities.

The Dijkstra algorithm uses labels that are positive integers or real numbers, which are [totally ordered](https://en.wikipedia.org/wiki/Total_order). It can be generalized to use any labels that are [partially ordered](https://en.wikipedia.org/wiki/Partially_ordered_set), provided the subsequent labels (a subsequent label is produced when traversing an edge) are [monotonically](https://en.wikipedia.org/wiki/Monotonic_function) non-decreasing. This generalization is called the generic Dijkstra shortest-path algorithm

The algorithm uses a greedy approach in the sense that we find the next best solution hoping that the end result is the best solution for the whole problem.

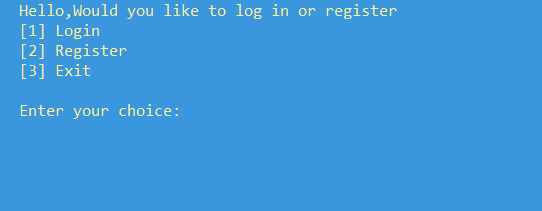
**CONCLUSIONS**

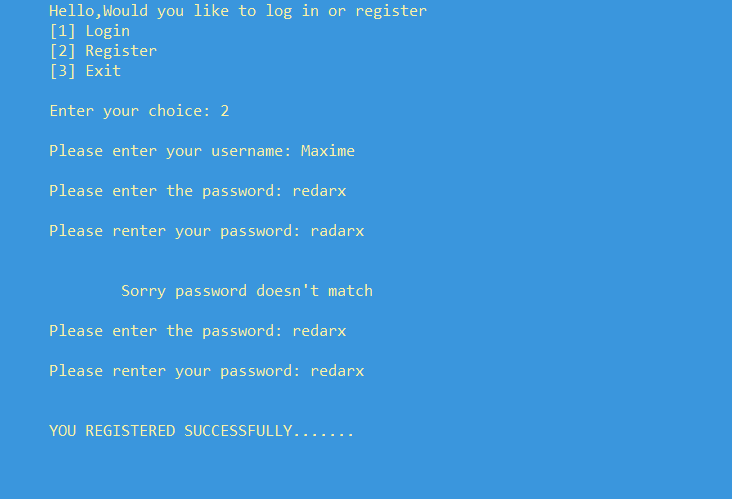
* We were successfully able to make a convenient taxi booking program for user
* In this project we studied and learnt about graphs and Dijkstra algorithm

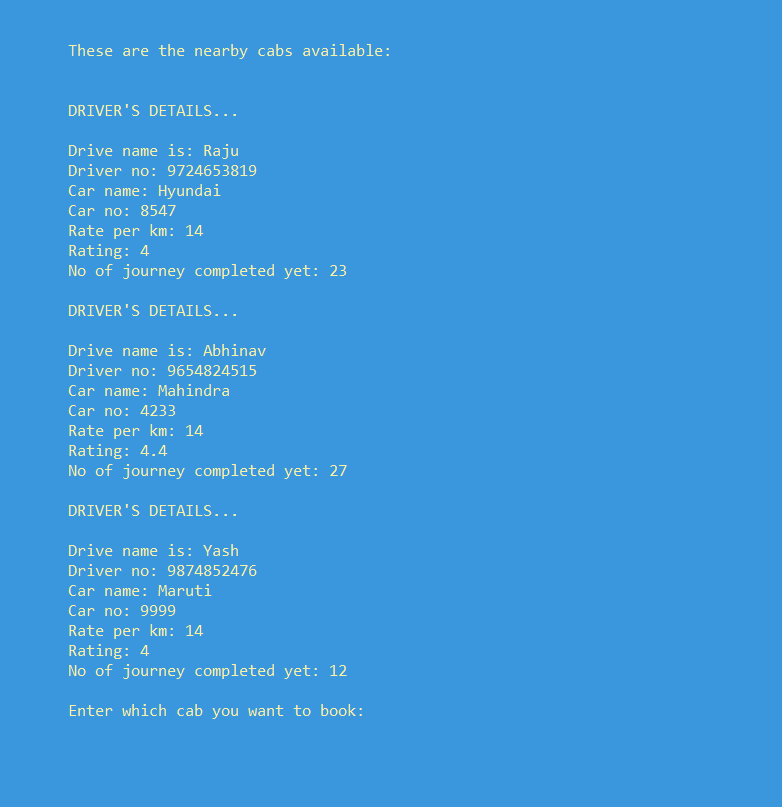
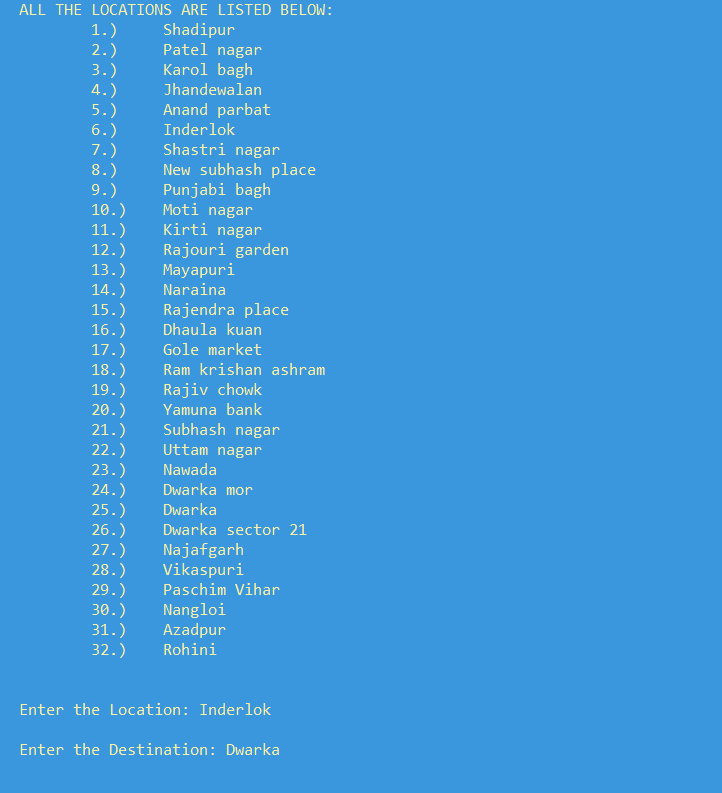
**References**

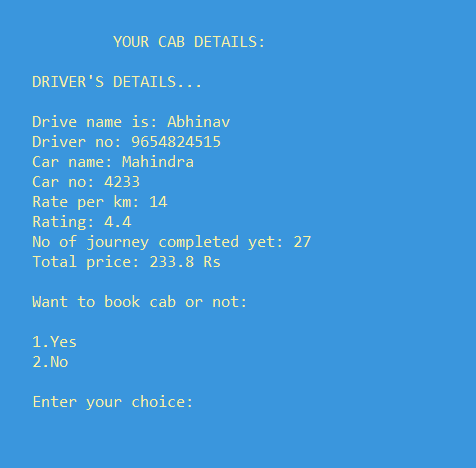
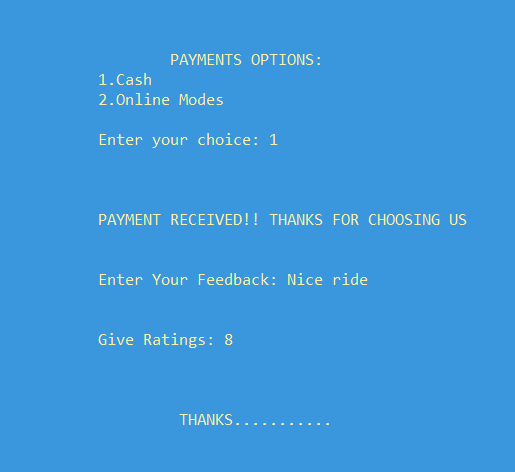
* <https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>
* <https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm>
* <https://www.geeksforgeeks.org/graph-data-structure-and-algorithms>
* Horowitz and Sahani, Fundamentals of Computer Algorithms, 2ND Edition ( PDFDrive )
* Weng, Gooi & Zailani, Suhaiza & Iranmanesh, Mohammad & Hyun, Sunghyup. (2017). Mobile taxi booking application service's continuance usage intention by users. Transportation Research Part D Transport and Environment. 57. 207-216.

**Annexure**

****

****

****

****