VELLORE INSTITUTE OF TECHNOLOGY

Department of Computer Science Engineering

Course code: CSE-3001

“Data Structures and Algorithm”

B.TECH – 3rd Semester

Academic year 2019-2020

Project Based on

“Car Sale System”

Slot – G2 slot

FINAL REVIEW

Submitted By

“ Mourya Vardhan Reddy ”

DECLARATION BY THE CANDIDATE

I hereby declare that the report titled "**Car Sale System**" submitted by me to VIT Chennai is a record of bonafide work undertaken by me under the supervision of **Dr. M. Jayasudha, Vellore Institute of Technology, Chennai.**

Signature of the Candidate

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. M. Jayasudha,** School of Computer Science and Engineering for her consistent encouragement and valuable guidance offered to us throughout the course of the project work.

We are extremely grateful to **Dean,** School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology, Chennai, for extending the facilities of the School towards our project and for his unstinting support.

We express our thanks to our **Head of the Department** for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the courses.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

BONAFIDE CERTIFICATE

Certified that this project report entitled "**Car Sale System"** is a bona-fide work of **G. Mourya Vardhan Reddy (19BCE1533), carried** out the "J"-Project work under my supervision and guidance for CSE2003 – Data Structures and Algorithms

Dr. M. Jayasudha

TABLE OF CONTENTS

Ch. No	**Chapter**	**Page Number**
1	Abstract	**6**
2	Introduction	6
3	Literature Survey / Requirements	7
4	Proposed System / Module(s) description	7-16
5	Results and Discussion	23
6	Reference	23
7	Conclusion	23

ABSTRACT

The main aim of this project is to create an application that is helpful while selling cars.

In the existing System it is difficult to maintain the car information individually and to supply for the customers who are eager to buy them. Customer has to face difficulty in order to know the information of car like manufacturing year, car model and other valuable information in a single domain. Our main idea is to develop a system where we can have all the required information for the user in order to effectively interest him in the process of buying a car.

INTRODUCTION

- This project's aims to develop an online car shopping/selling for customers with the goal so that it is very easy to shop your loved cars from an extensive number of retailers and showrooms available. With the help of this you can explore multiple cars from your home. Here is no compelling reason to go to the crowed showrooms or to unknown sellers. You simply require a PC or a laptop and one important payment sending option to shop online.
- To get to this car sales system all the customers will need to have an email and password to login and proceed your shopping. The login credentials for a car sales system are under high security and nobody will have the capacity to crack it easily.
- Upon successful login the customers can purchase a car. Not just Indian cars, you can also purchase from outside nations by few clicks on your mouse.
- It is simple. You will enter your car specifications in the efficient search module and you can see all the available car models from every range, depending on brand new or used cars. Most of the information is available to the user such a car model, age of the car, quality, fuel efficiency, etc. The user nee a computer and a payment making options like net banking, credit card, debit card or PayPal to make the payment after viewing the car and maybe even having a look at the car after having a chat with the owner.

LITERATURE SURVEY

- We took this project taking the idea or the approach of the various things or tasks during this pandemic situation. Like all the things like classes of school and colleges are online , then there is work from home, then shopping from home which was quite popular before also but now necessities like groceries , medicines and other things are also quite bought popular, so we thought that why not think a little more big and about the future when this pandemic is over and we can travel like before with even more enthusiasm. And when travelling comes to mind the first thing that strikes is a perfect long drive, so we thought why not sell and buy cars online. Isn't this something spectacular? We know that there are already many websites like cars24 but we thought to also add the nearest approach for the seller and buyer with the help of google locations so that they don't have to travel long distances.

- So, we intended to give the buyer and seller a better option to get a filter of locations also so that they can keep in mind about that too when buying cars. We wanted to prove that not only cars can be bought or visited online but also like get the distance where the seller is. We have like hospital management system, database management systems, etc. so let's make car management system also. This will affect like in the future when we will have to travel like before then we may need our cars and vehicles to travel as some people will still have doubts about doing public travel. So, this system will help them to manage and get cars from nearby sellers quite easily.

PROPOSED SYSTEM MODULES:

Modules:

This application is divided into following modules

- Addition/Deletion of Cars and details
- Sort By popularity
- Sort By price
- Buying/selling facility
- Source Code

Module Description:

Addition/Deletion of Cars and details:

- System shall allow the administrator to login and add data related to cars and its images.
- He will also be able to delete old data related to cars.
- Admin has the permission to add, edit and delete data
- Shall be able to save list of cars

Viewing details:

- Search car based on matched keywords such as cost, year of manufacture etc.
- Search result page returns image of car, car name and information related to car

Sort by Popularity:

- This is a feature where user can see no of cars sold or bought by customers.
- So, they can know what is the bestselling car and decide accordingly.

Sort by Price:

- This a feature where user can no longer need to search for cars in their budget, they can simply sort it by their wish it can be either low to high and high to low.

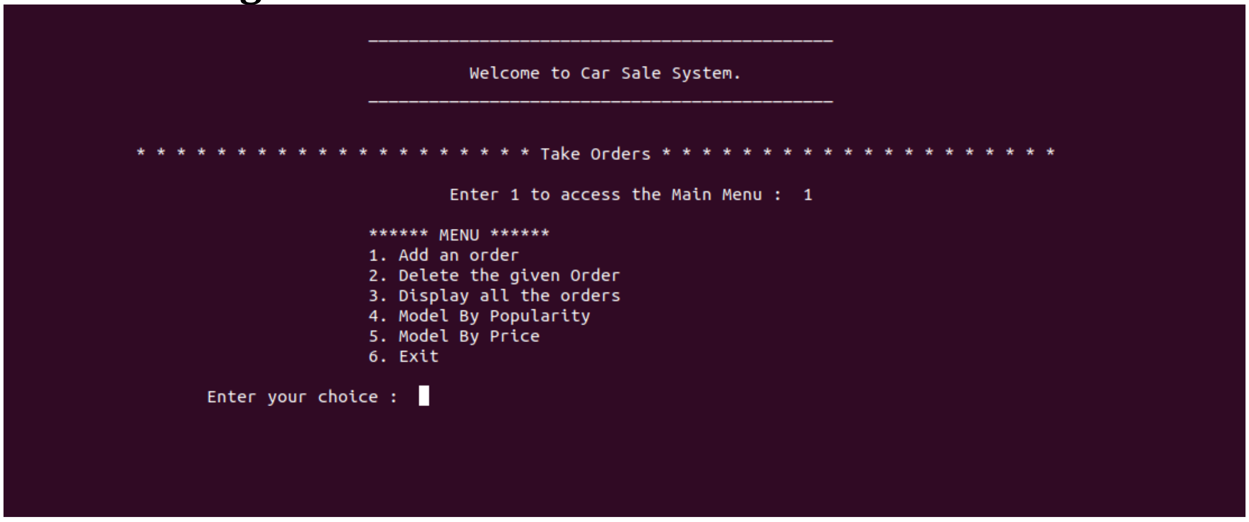
- As our project is an online one customer can have a hesitant free choice unlike the showrooms where they might be judged or people sometimes feel less of them compared to others.

Buying/selling car

- Customer can search the cars of their choice and can buy them or customers can sell their car buy entering details and price.

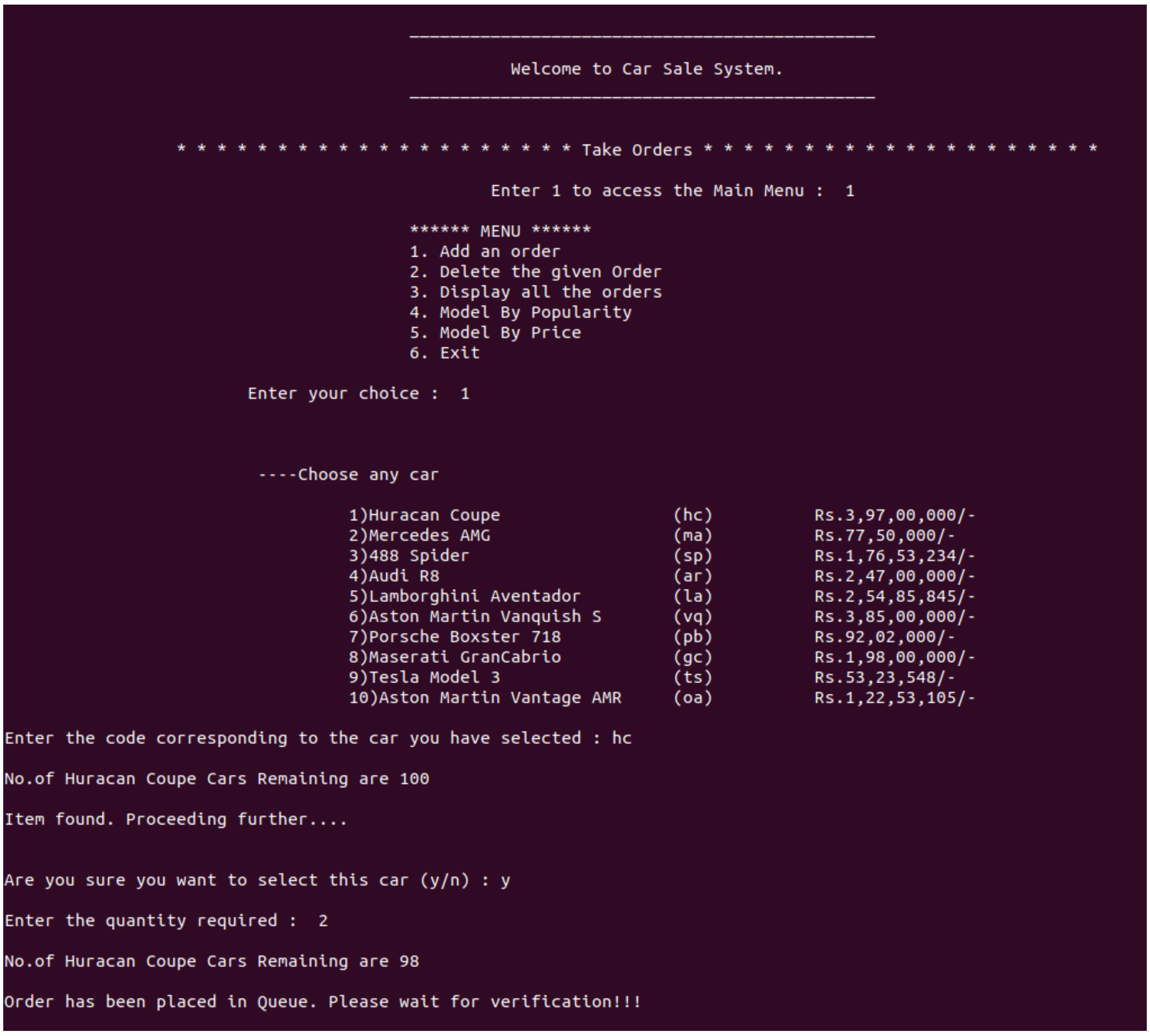
<u>Implementation:(command line)</u>

Welcome Page:



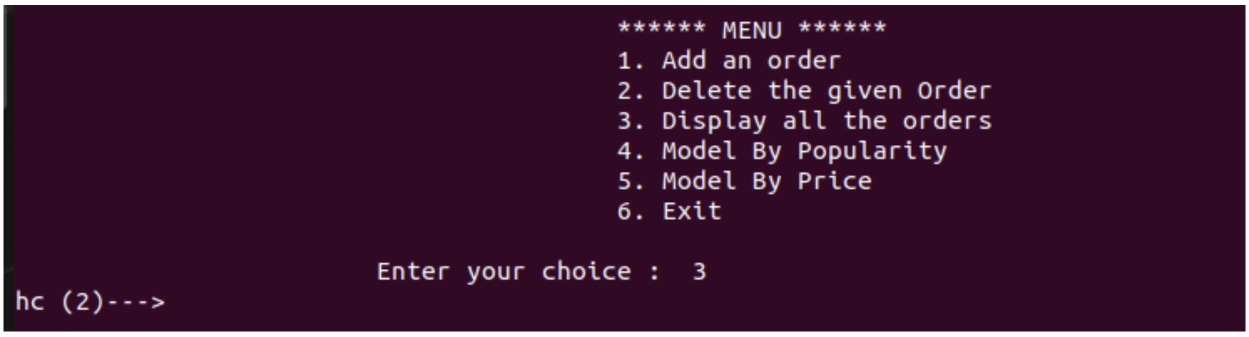
```
                              _____________________________________________

                                        Welcome to Car Sale System.
                              _____________________________________________

          * * * * * * * * * * * * * * * * * * * * Take Orders * * * * * * * * * * * * * * * * * * * *

                                    Enter 1 to access the Main Menu :  1

                              ****** MENU ******
                              1. Add an order
                              2. Delete the given Order
                              3. Display all the orders
                              4. Model By Popularity
                              5. Model By Price
                              6. Exit

               Enter your choice :  
```

Add an order:



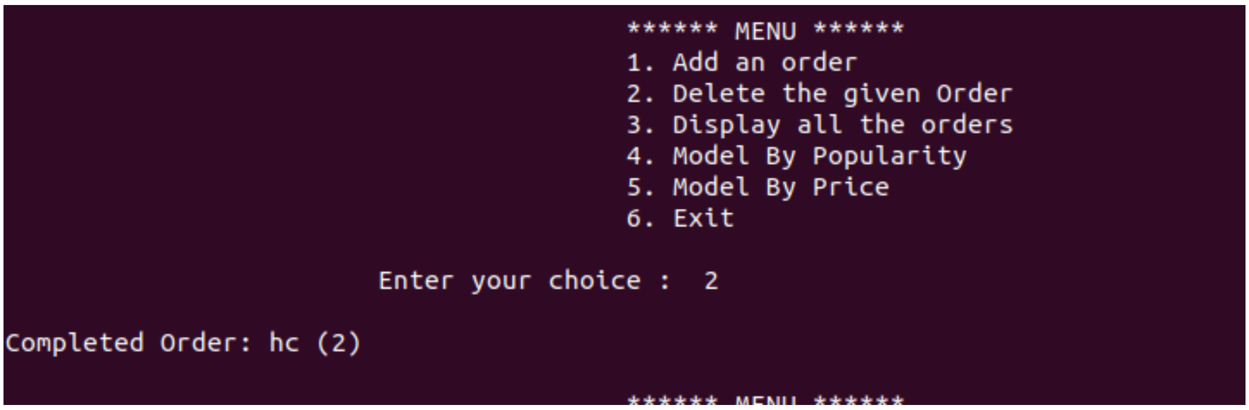
```
                                        ______________________________________________

                                                  Welcome to Car Sale System.
                                        ______________________________________________

                 * * * * * * * * * * * * * * * * * * * * Take Orders * * * * * * * * * * * * * * * * * * * *

                                                Enter 1 to access the Main Menu :  1

                                        ****** MENU ******
                                        1. Add an order
                                        2. Delete the given Order
                                        3. Display all the orders
                                        4. Model By Popularity
                                        5. Model By Price
                                        6. Exit

                        Enter your choice :  1



                         ----Choose any car

                                  1)Huracan Coupe                (hc)          Rs.3,97,00,000/-
                                  2)Mercedes AMG                 (ma)          Rs.77,50,000/-
                                  3)488 Spider                   (sp)          Rs.1,76,53,234/-
                                  4)Audi R8                      (ar)          Rs.2,47,00,000/-
                                  5)Lamborghini Aventador        (la)          Rs.2,54,85,845/-
                                  6)Aston Martin Vanquish S      (vq)          Rs.3,85,00,000/-
                                  7)Porsche Boxster 718          (pb)          Rs.92,02,000/-
                                  8)Maserati GranCabrio          (gc)          Rs.1,98,00,000/-
                                  9)Tesla Model 3                (ts)          Rs.53,23,548/-
                                  10)Aston Martin Vantage AMR    (oa)          Rs.1,22,53,105/-

Enter the code corresponding to the car you have selected : hc

No.of Huracan Coupe Cars Remaining are 100

Item found. Proceeding further....


Are you sure you want to select this car (y/n) : y

Enter the quantity required :  2

No.of Huracan Coupe Cars Remaining are 98

Order has been placed in Queue. Please wait for verification!!!
```

Display Orders:



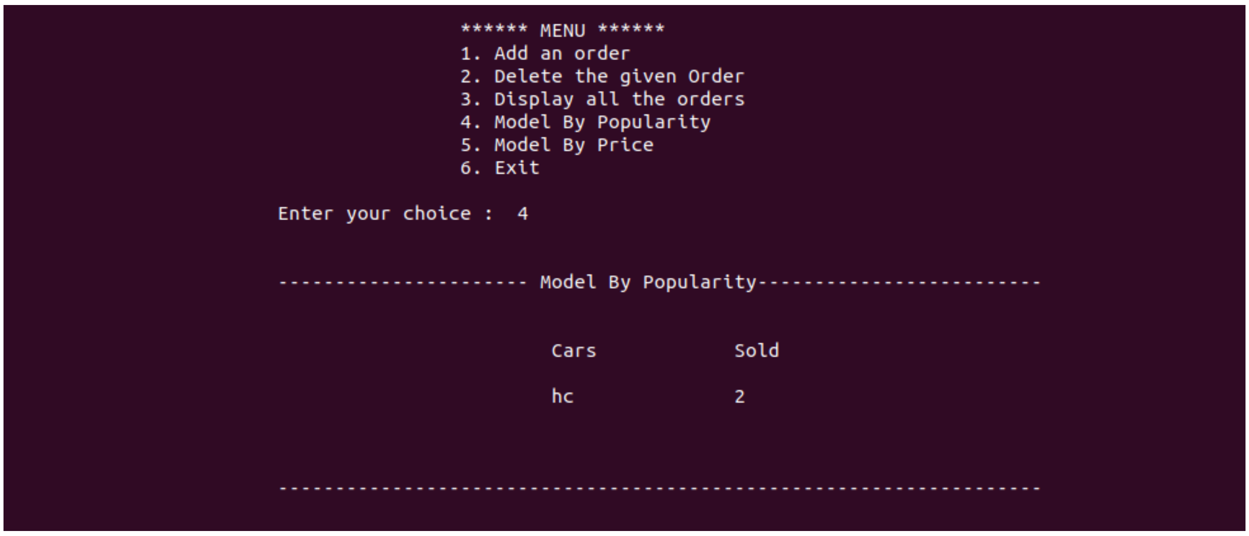
```
                                        ****** MENU ******
                                        1. Add an order
                                        2. Delete the given Order
                                        3. Display all the orders
                                        4. Model By Popularity
                                        5. Model By Price
                                        6. Exit

                           Enter your choice :  3
hc (2)--->
```

Delete Order:



```
                                        ****** MENU ******
                                        1. Add an order
                                        2. Delete the given Order
                                        3. Display all the orders
                                        4. Model By Popularity
                                        5. Model By Price
                                        6. Exit

                        Enter your choice :  2

Completed Order: hc (2)
```

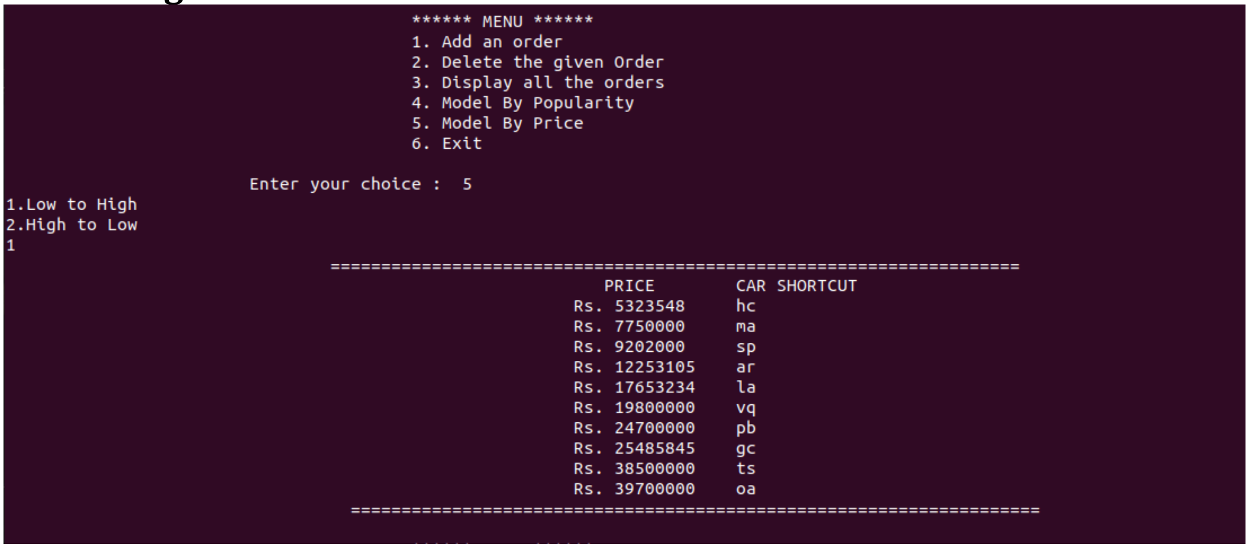
Sort By Popularity:



```
                                        ****** MENU ******
                                        1. Add an order
                                        2. Delete the given Order
                                        3. Display all the orders
                                        4. Model By Popularity
                                        5. Model By Price
                                        6. Exit

                        Enter your choice :  4


                        --------------------- Model By Popularity------------------------


                                        Cars            Sold

                                        hc              2



                        -----------------------------------------------------------------
```

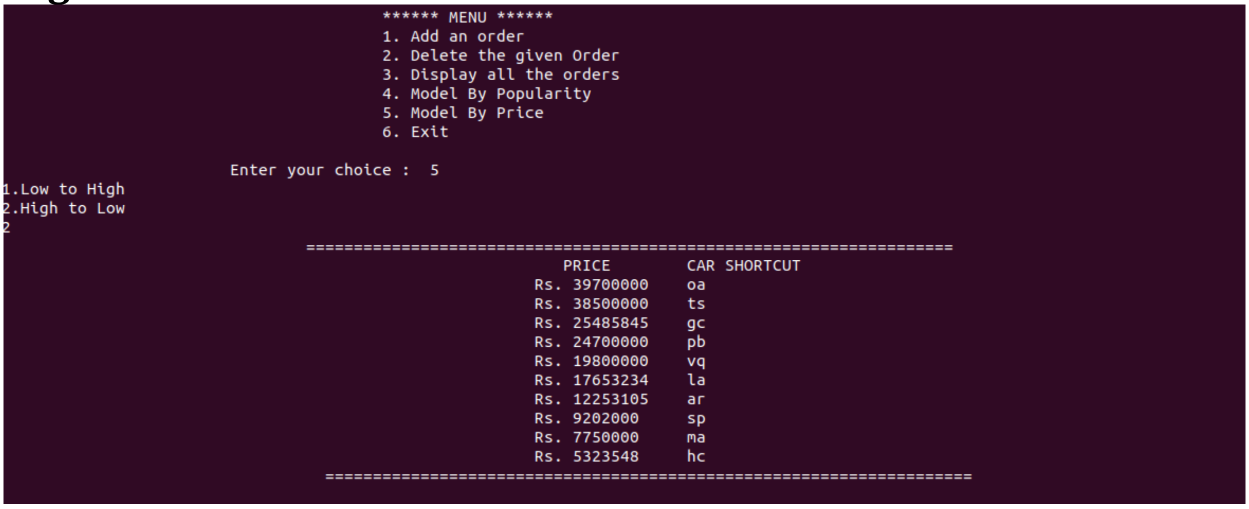
Sort by Price:

Low to High:



```
                                        ****** MENU ******
                                        1. Add an order
                                        2. Delete the given Order
                                        3. Display all the orders
                                        4. Model By Popularity
                                        5. Model By Price
                                        6. Exit

                        Enter your choice :  5
1.Low to High
2.High to Low
1
                              ===================================================================
                                                   PRICE          CAR SHORTCUT
                                                Rs. 5323548       hc
                                                Rs. 7750000       ma
                                                Rs. 9202000       sp
                                                Rs. 12253105      ar
                                                Rs. 17653234      la
                                                Rs. 19800000      vq
                                                Rs. 24700000      pb
                                                Rs. 25485845      gc
                                                Rs. 38500000      ts
                                                Rs. 39700000      oa
                                ===================================================================
```

High to Low:



```
                                   ****** MENU ******
                                   1. Add an order
                                   2. Delete the given Order
                                   3. Display all the orders
                                   4. Model By Popularity
                                   5. Model By Price
                                   6. Exit

                     Enter your choice :  5
1.Low to High
2.High to Low
2
                            ====================================================================
                                                        PRICE         CAR SHORTCUT
                                                     Rs. 39700000     oa
                                                     Rs. 38500000     ts
                                                     Rs. 25485845     gc
                                                     Rs. 24700000     pb
                                                     Rs. 19800000     vq
                                                     Rs. 17653234     la
                                                     Rs. 12253105     ar
                                                     Rs. 9202000      sp
                                                     Rs. 7750000      ma
                                                     Rs. 5323548      hc
                              ====================================================================
```

Implementation:()

- Homepage:
- Login Page:
- Register Page:
- Store Page:
- Searching for Car in Store:
- Filtering by Brand:
- Filtering by Body Type:
- Car Adding Functionality by Customer or Showroom

CONCLUSION:

We started with the thought of making a web application for the car's management and trading. We used Django for the front-end development as it is quite friendly with the internet applications. We used to search and sorting algorithms for the filters like sort by name, model, price, etc. We were going to use Dijkstra algorithm for finding the shortest distance but then we found out that Django is not supporting any API files and also, it's fluctuating with the database like sometimes it takes the complete data, sometimes part of it and then few times not at all. So, we then switched back to c language and command window for executing the approach. Here we don't have the good and sparking of the web application that we created before but here it's basically working properly. We have a menu that asks for the various options you want to try and other things similar to the website like sorting and searching. Overall, we wanted to do something else but eventually did something different learning quite a number of things in the journey. So, it was a quite substantial, prominent experience in doing this project of cars management system.

REFERENCES:

- https://www.agileventures.org/projects/car-sale-system
- https://nevonprojects.com/car-sales-system-mini-project/
- https://www.cardekho.com/
- https://projectsgeek.com/2016/02/car-sales-system-project-in-java.html
- http://www.proactivesoft.com/pro_CarSales.html

APPENDIX:

Sorting Algorithm:(used in both and command line)

A Sorting Algorithm is used to rearrange a given array or list elements according to a comparison operator on the elements. The comparison operator is used to decide the new order of element in the respective data structure.

```
void swap(int* a, int* b)
{
    int t = *a;
    *a = *b;
    *b = t;
}
void swap2(char *str1, char *str2)
{
  char *temp = str1;
  str1 = str2;
  str2 = temp;
}
int partition (int arr[], int low, int high, char arr2[][2])
{
    int pivot = arr[high];    // pivot
    int i = (low - 1);  // Index of smaller element

    for (int j = low; j <= high- 1; j++)
    {
        // If current element is smaller than the pivot
        if (arr[j] < pivot)
        {
            i++;    // increment index of smaller element
            swap(&arr[i], &arr[j]);
            swap2(&arr2[i][2], &arr2[j][2]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    swap2(&arr2[i+1][2], &arr2[high][2]);
    return (i + 1);
}

/* The main function that implements QuickSort
 arr[] --> Array to be sorted,
  low  --> Starting index,
  high  --> Ending index */
void quickSort(int arr[], int low, int high, char arr2[][2])
{
    if (low < high)
    {
        /* pi is partitioning index, arr[p] is now
           at right place */
        int pi = partition(arr, low, high,arr2);
        // Separately sort elements before
        // partition and after partition
        quickSort(arr, low, pi - 1,arr2);
        quickSort(arr, pi + 1, high,arr2);
    }
}
```

Search Algorithm:()

The searching algorithms are used to search or find one or more than one element from a dataset. These type of algorithms are used to find elements from a specific data structures.
Searching may be sequential or not. If the data in the dataset are random, then we need to use sequential searching. Otherwise we can use other different techniques to reduce the complexity.

```
def binarySearch(arr, x,arr1):
    l = 0
    r = len(arr)
    while (l <= r):
        m = l + ((r - l) // 2)

        res = (x == arr[m])

        # Check if x is present at mid
        if (res == 0):
            return m - 1

        # If x greater, ignore left half
        if (res > 0):
            l = m + 1

        # If x is smaller, ignore right half
        else:
            r = m - 1

        arr1 = arr[m]
    return ''
```

Dijkstra's algorithm:( )

The algorithm we are going to use to determine the shortest path is called " Dijkstra's algorithm." Dijkstra's algorithm is an iterative algorithm that provides us with the shortest path from one particular starting node to all other nodes in the graph. ... When a vertex is first created dist is set to a very large number.

```
from collections import defaultdict
import sys
import math

def printingpath(src, dest, path):
        i = dest
        a = []
        while(i != src):
            a.append(i)
            i = path[i]
        a.reverse()
        a.insert(0, src)
        return a

class Heap():

    def __init__(self):
        self.array = []
        self.size = 0
        self.pos = []

    def newMinHeapNode(self, v, dist):
        minHeapNode = [v, dist]
        return minHeapNode

    # A utility function to swap two nodes
    # of min heap. Needed for min heapify
    def swapMinHeapNode(self,a, b):
        t = self.array[a]
        self.array[a] = self.array[b]
        self.array[b] = t

    # A standard function to heapify at given idx
    # This function also updates position of nodes
    # when they are swapped.Position is needed
    # for decreaseKey()
    def minHeapify(self, idx):
        smallest = idx
        left = 2*idx + 1
        right = 2*idx + 2

        if left < self.size and self.array[left]
[1] < self.array[smallest][1]:
            smallest = left
        if right < self.size and self.array[right]
[1] < self.array[smallest][1]:
            smallest = right

        # The nodes to be swapped in min
        # heap if idx is not smallest
        if smallest != idx:

            # Swap positions
            self.pos[ self.array[smallest][0] ] = idx
            self.pos[ self.array[idx][0] ] = smallest

            # Swap nodes
            self.swapMinHeapNode(smallest, idx)

            self.minHeapify(smallest)

    #for checking the root node
    def peek(self):
        temp = self.array[0][0]
        return temp

    # Standard function to extract minimum
    # node from heap
    def extractMin(self):

        # Return NULL wif heap is empty
        if self.isEmpty() == True:
            return

        # Store the root node
        root = self.array[0]

        # Replace root node with last node
        lastNode = self.array[self.size - 1]
        self.array[0] = lastNode

        # Update position of last node
        self.pos[lastNode[0]] = 0
        self.pos[root[0]] = self.size - 1

        # Reduce heap size and heapify root
        self.size -= 1
        self.minHeapify(0)

        return root

    def isEmpty(self):
        return True if self.size == 0 else False

    def decreaseKey(self, v, dist):
        # Get the index of v in heap array

        i = self.pos[v]

        # Get the node and update its dist value
        self.array[math.ceil(i)][1] = dist

        # Travel up while the complete tree is
        # not hepified. This is a O(Logn) loop
        while i > 0 and self.array[math.ceil(i)]
[1] < self.array[math.ceil((i - 1) / 2)][1]:

            # Swap this node with its parent
            self.pos[ self.array[math.ceil(i)][0] ] = (i-1)/2
            self.pos[ self.array[math.ceil((i-1)/2)][0] ] = i
            self.swapMinHeapNode(math.ceil(i), math.ceil((i - 1)/2) )

            # move to parent index
            i = (i - 1) / 2;

    # A utility function to check if a given
    # vertex 'v' is in min heap or not
    def isInMinHeap(self, v):

        if self.pos[v] < self.size:
            return True
        return False


def printArr(dist, n):
    print("Vertex\tDistance from source")
    for i in range(n):
        print("%d\t\t%d" % (i,dist[i]))


class Graph():

    def __init__(self, V):
        self.V = V
        self.graph = defaultdict(list)

    # Adds an edge to an undirected graph
    def addEdge(self, src, dest, weight):

        # Add an edge from src to dest. A new node
        # is added to the adjacency list of src. The
        # node is added at the beginning. The first
        # element of the node has the destination
        # and the second elements has the weight
        newNode = [dest, weight]
        self.graph[src].insert(0, newNode)


    # The main function that calculates distances
    # of shortest paths from src to all vertices.
    # It is a O(ELogV) function
    def dijkstra(self, src, dest):

        V = self.V # Get the number of vertices in graph
        dist = [] # dist values used to pick minimum
                    # weight edge in cut
        path = [None] * V
        # minHeap represents set E
        minHeap = Heap()

        # Initialise min heap with all vertices.
        # dist value of all vertices
        for v in range(V):
            dist.append(sys.maxsize)
            minHeap.array.append( minHeap.newMinHeapNode(v, dist[v]) )

            minHeap.pos.append(v)

        # Make dist value of src vertex as 0 so
        # that it is extracted first
        dist[src] = 0
        minHeap.decreaseKey(src, dist[src])

        # Initially size of min heap is equal to V
        minHeap.size = V;

        # In the following loop, min heap contains all nodes
        # whose shortest distance is not yet finalised.
        while minHeap.isEmpty() == False:

            # Extract the vertex with minimum distance value
            newHeapNode = minHeap.extractMin()
            u = newHeapNode[0]

            # Traverse through all adjacent vertices of
            # u (the extracted vertex) and update their
            # distance values
            for pCrawl in self.graph[u]:

                v = pCrawl[0]

                # If shortest distance to v is not finalized
                # yet, and distance to v through u is less
                # than its previously calculated distance
                if minHeap.isInMinHeap(v) and dist[u] !
  = sys.maxsize and pCrawl[1] + dist[u] < dist[v]:
                        dist[v] = pCrawl[1] + dist[u]
                        path[v] = u
                        # update distance value
                        # in min heap also
                        minHeap.decreaseKey(v, dist[v])
        return [dist, path]
    def dijkstrafortwo(self, src1, src2, dest):
        dij1 = self.dijkstra(src1, dest)
        d1 = dij1[0][dest]
        diff1 = dij1[0][src2]
        path1 = dij1[1]
        path1_1 = printingpath(src1, dest, path1)
        dij2 = self.dijkstra(src2, dest)
        d2 = dij2[0][dest]
        naive = diff1 + d2
        diff2 = {}
        for nodes in range(len(path1_1)):
            diff2[path1_1[nodes]] = dij2[0][path1_1[nodes]]
        diff2_final = min(diff2.values())
        optimal = d1 + diff2_final
        path2 = dij2[1]
        path2_1 = printingpath(src2, dest, path2)
        a = dest
        if(naive <= optimal):
            path1_2 = printingpath(src1, src2, path1)
            return [path1_2, path2_1, False]
        else:
            for key in diff2:
                if(diff2_final == diff2[key]):
                    a = key
                    break
            path2_2 = printingpath(src2, a, path2)
            return [path2_2, path1_1, True]

# Driver program to test the above functions
# conn to database
# for every data in db
# graph.addedge(data1, data2, distance)
```

SOURCE CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define max 20 // maximum number of orders can be given

int hc = 100, ma = 100, sp = 100, ar = 100, la = 100, vq = 100, pb = 100, gc = 100, ts = 100,
oa = 100; //maximum quantities of cars

// This is the declaration of Order list in queue using linked list

struct Node
{
  char data[2];
  int required;
  struct Node *next;
} *front = NULL, *rear = NULL;

// Priority Queue in which the model by popularity is stored

struct node
{
  int priority;
  char data[2];
  struct node *link;
} *Front = NULL;

void insert(char[]);
void deleted(int);
void display();
void Orders();
void modelByPopularity();
void modelByPrice();
int dataBase(char data[2], int);

void SortingPopularity(char[], int);
void displayPopularity();
void sortingPriority();

int main()
{
  int x, y;
  printf("\n");
  printf("\n");
  printf("\t \t \t \t \t\t\tLa Voiture\n");
  printf("\t\t\t\t\t___________________________________________");
  printf("\n");
  printf("\n\t\t\t\t\t  Welcome to Global Car Sales Company and Co.\n");
  printf("\t\t\t\t\t___________________________________________");
  int choice;
  printf("\n");
  printf("\n");
  printf("\n");
  printf("\t \t * * * * * * * * * * * * * * * * * * * * Take Orders * * * * * * * * * * * * * * * * * * *
*\n");
  printf("\n");
  printf("\t\t\t\t\t\tEnter 1 to access the Main Menu :  ");
  scanf("%d", &choice);
  switch (choice)
  {
  case 1:
    Orders();
    break;
  }
  return 0;
}

void Orders()
{
  int choice;
  char img[1];
  char ch;
```

```
 char value[2];
 //printf("\n:: Orders ::\n");
 while (1)
 {
  system("COLOR 07");

  printf("\n\t\t\t\t\t****** MENU ******\n");
  printf("\t\t\t\t\t1. Add an order\n\t\t\t\t\t2. Delete the given Order\n\t\t\t\t\t3.
Display all the orders\n");
  printf("\t\t\t\t\t4. Model By Popularity\n");
  printf("\t\t\t\t\t5. Model By Price\n");
  printf("\t\t\t\t\t6. Exit\n");
  printf("\n");
  printf("\t\t\tEnter your choice :  ");
  scanf("%d", &choice);
  switch (choice)
  {
  case 1:
  again:
   printf("\n\n\n \t\t\t ----Choose any car\n");
   printf("\n");

   // Cars List Represented here
   printf("\t\t\t\t  1)Huracan Coupe              (hc)        Rs.3,97,00,000/- \n");
   printf("\t\t\t\t  2)Mercedes AMG               (ma)        Rs.77,50,000/- \n");
   printf("\t\t\t\t  3)488 Spider                 (sp)        Rs.1,76,53,234/- \n");
   printf("\t\t\t\t  4)Audi R8                    (ar)        Rs.2,47,00,000/- \n");
   printf("\t\t\t\t  5)Lamborghini Aventador      (la)        Rs.2,54,85,845/- \n");
   printf("\t\t\t\t  6)Aston Martin Vanquish S    (vq)        Rs.3,85,00,000/- \n");
   printf("\t\t\t\t  7)Porsche Boxster 718        (pb)        Rs.92,02,000/- \n");
   printf("\t\t\t\t  8)Maserati GranCabrio        (gc)        Rs.1,98,00,000/- \n");
   printf("\t\t\t\t  9)Tesla Model 3              (ts)        Rs.53,23,548/- \n");
   printf("\t\t\t\t  10)Aston Martin Vantage AMR  (oa)        Rs.1,22,53,105/- \n");

   while (1)
   {
```

```
  printf("\nEnter the code corresponding to the car you have selected : ");
  scanf("%s", value);

  int rfdb = dataBase(value, 0); //rfdb is returning from database
  if (rfdb == 0)
  {
   printf("\nItem found. Proceeding further....\n\n");
   break;
  }
  else
  {
   system("COLOR FC");

   printf("Item not found in database. Please select from the given list of cars!\n");
  }
 }

// Here the image (external file) is invoked using system() : command

system("COLOR 07");
printf("To get a glance of the Selected car, Enter the corresponding index :  ");
scanf("%s", img); //here img is the index of the selected car in the given list
system(strcat(img, ".jpg"));

printf("\nAre you sure you want to select this car (y/n) : ");
fflush(stdin);
scanf("%c", &ch);
fflush(stdin);
if (ch == 'y' || ch == 'Y')
{
  insert(value);
}
else if (ch == 'n' || ch == 'N')
{
  goto again;
}
```

```
     break;
    case 2:
     deleted(1);
     break;
    case 3:
     display();
     break;
    case 4:
     modelByPopularity();
     break;
    case 5:
     modelByPrice();
     break;
    case 6:
     exit(0);
    default:
     system("COLOR FC");

     printf("\nWrong selection!!! Please try again!!!\n");
    }
  }
}

void insert(char value[])
{

  struct Node *newNode;
  newNode = (struct Node *)malloc(sizeof(struct Node));
  strcpy(newNode->data, value);
  // newNode->data = value;  :: We cannot do this when we are copying a string into a node,
it shows error whereas we have to use strcpy
  newNode->next = NULL;
  if (front == NULL)
    front = rear = newNode;
  else
  {
    rear->next = newNode;
    rear = newNode;
  }
  printf("\nEnter the quantity required :  ");
  scanf("%d", &newNode->required);
  int rfdb = dataBase(newNode->data, newNode->required); //rfdb is returning from
database
  if (rfdb == 0)
  {
    SortingPopularity(newNode->data, newNode->required);
    printf("\nOrder has been placed in Queue. Please wait for verification!!!\n");
  }
  else if (rfdb == 2)
  {
    system("COLOR FC");
    printf("\nSorry Please go with another car\n");
    deleted(0);
  }
}

void deleted(int noItem)
{
  if (front == NULL)
    printf("\nOrder list is empty!!!\n");
  else if (noItem == 0)
  {
    struct Node *temp = front;
    front = front->next;
    printf("\nWrong Order: %s (%d)  has been deleted\n", temp->data, temp->required);
    free(temp);
  }
  else
  {
    struct Node *temp = front;
    front = front->next;
    printf("\nCompleted Order: %s (%d)\n", temp->data, temp->required);
    free(temp);
  }
}

void display()
{
  if (front == NULL)
    printf("\nOrder list is empty!!!\n");
  else
  {
    struct Node *temp = front;
    while (temp->next != NULL)
    {
      printf(" %s (%d)--->", temp->data, temp->required);
      temp = temp->next;
    }
    printf("%s (%d)--->NULL\n", temp->data, temp->required);
  }
}

int dataBase(char data[2], int req)
{
  char *carsList[10] = {"hc", "ma", "sp", "ar", "la", "vq", "pb", "gc", "ts", "oa"}; // we can store
string array in c in this way by creating a pointer array
  int i;
  if (req <= 100)
  {
    for (i = 0; i < 10; i++)
    {
      int j = strcmp(data, carsList[i]); // Checking whether the required car is present in the
database or not
      if (j == 0)
      {
        //Remaining cars which are there in Data base are shown from this to the users
        switch (data[0])
        {
```

```
case 'h':
 if (hc > 0)
 {
  hc = hc - req;
  printf("\nNo.of Huracan Coupe Cars Remaining are %d\n", hc);
 }
 else if (hc == 0)
 {
  printf("Huracan Coupe Cars have been completed. Please wait for 30 days\n");
 }
 break;

case 'm':
 if (ma > 0)
 {
  ma = ma - req;
  printf("\nNo.of Mercedes AMG Cars Remaining are %d\n", ma);
 }
 else if (ma == 0)
 {
  printf("Mercedes AMG Cars have been completed. Please wait for 30 days\n");
 }
 break;

case 's':
 if (sp > 0)
 {
  sp = sp - req;
  printf("\nNo.of 488 Spider Cars Remaining are %d\n", sp);
 }
 else if (sp == 0)
 {
  printf("488 Spider Cars have been completed. Please wait for 30 days\n");
 }
 break;
```

```
    case 'a':
     if (ar > 0)
     {
       ar = ar - req;
       printf("\nNo.of Audi R8 Cars Remaining are %d\n", ar);
     }
     else if (ar == 0)
     {
       printf("Audi R8 Cars have been completed. Please wait for 30 days\n");
     }
     break;

    case 'l':
     if (la > 0)
     {
       la = la - req;
       printf("\nNo.of Lamborghini Aventador Cars Remaining are %d\n", la);
     }
     else if (la == 0)
     {
       printf("Lamborghini Aventador Cars have been completed. Please wait for 30
days\n");
     }
     break;

    case 'v':
     if (vq > 0)
     {
       vq = vq - req;
       printf("\nNo.of Aston Martin Vanquish S Cars Remaining are %d\n", vq);
     }
     else if (vq == 0)
     {
       printf("Aston Martin Vanquish S Cars have been completed. Please wait for 30
days\n");
     }
```

```
 break;

case 'p':
 if (pb > 0)
 {
  pb = pb - req;
  printf("\nNo.of Porsche Boxster 718 Cars Remaining are %d\n", pb);
 }
 else if (pb == 0)
 {
  printf("Porsche Boxster 718 Cars have been completed. Please wait for 30 days\n");
 }
 break;

case 'g':
 if (gc > 0)
 {
  gc = gc - req;
  printf("\nNo.of Maserati GranCabrio Cars Remaining are %d\n", gc);
 }
 else if (gc == 0)
 {
  printf("Maserati GranCabrio Cars have been completed. Please wait for 30 days\n");
 }
 break;

case 't':
 if (ts > 0)
 {
  ts = ts - req;
  printf("\nNo.of Tesla Model 3 Cars Remaining are %d\n", ts);
 }
 else if (ts == 0)
 {
  printf("Tesla Model 3 Cars have been completed. Please wait for 30 days\n");
 }
```

```
          break;

        case 'o':
          if (oa > 0)
          {
            oa = oa - req;
            printf("\nNo.of Aston Martin Vantage AMR Cars Remaining are %d\n", oa);
          }
          else if (oa == 0)
          {
            printf("Aston Martin Vantage AMR Cars have been completed. Please wait for 30
days\n");
          }
          break;
        }
        return 0;
      }
    }
    return 1;
  }
  else
  {
    printf("\nNo enough items in garage\n");
    return 2;
  }
}

void modelByPopularity()
{
  // Given car required quantities as parameter
  // arrange everything dynamically
  // Using Priority Queue
  displayPopularity();
}
void modelByPrice()
{
 int choice;
 printf("1.High to Low\n");
 printf("2.Low to High\n");
 scanf("%d", &choice);

 if (choice == 1)
 {
  printf("\t\t\t\t
==================================================================\
n");
  printf("\t\t\t\t  || 1)Huracan Coupe            (hc)          Rs.3,97,00,000/-  ||\n");
  printf("\t\t\t\t  || 2)Aston Martin Vanquish S     (vq)          Rs.3,85,00,000/-  ||\n");
  printf("\t\t\t\t  || 3)Lamborghini Aventador      (la)          Rs.2,54,85,845/-  ||\n");
  printf("\t\t\t\t  || 4)Audi R8                   (ar)          Rs.2,47,00,000/-  ||\n");
  printf("\t\t\t\t  || 5)Maserati GranCabrio        (gc)          Rs.1,98,00,000/-  ||\n");
  printf("\t\t\t\t  || 6)488 Spider                (sp)          Rs.1,76,53,234/-  ||\n");
  printf("\t\t\t\t  || 7)Aston Martin Vantage AMR    (oa)          Rs.1,22,53,105/-  ||\n");
  printf("\t\t\t\t  || 8)Porsche Boxster 718        (pb)          Rs.92,02,000/-    ||\n");
  printf("\t\t\t\t  || 9)Mercedes AMG             (ma)          Rs.77,50,000/-    ||\n");
  printf("\t\t\t\t  ||10)Tesla Model 3             (ts)          Rs.53,23,548/-    ||\n");
  printf("\t\t\t\t
=================================================================="
);
 }
 else if (choice == 2)
 {
  printf("\t\t\t\t
==================================================================\n
");
  printf("\t\t\t\t  || 1)Tesla Model 3             (ts)          Rs.53,23,548/-    ||\n");
  printf("\t\t\t\t  || 2)Mercedes AMG             (ma)          Rs.77,50,000/-    ||\n");
  printf("\t\t\t\t  || 3)Porsche Boxster 718        (pb)          Rs.92,02,000/-    ||\n");
  printf("\t\t\t\t  || 4)Aston Martin Vantage AMR    (oa)          Rs.1,22,53,105/-  ||\n");
  printf("\t\t\t\t  || 5)488 Spider                (sp)          Rs.1,76,53,234/-  ||\n");
  printf("\t\t\t\t  || 6)Maserati GranCabrio        (gc)          Rs.1,98,00,000/-  ||\n");
  printf("\t\t\t\t  || 7)Audi R8                   (ar)          Rs.2,47,00,000/-  ||\n");
```

```
   printf("\t\t\t\t  || 8)Lamborghini Aventador       (la)           Rs.2,54,85,845/-  ||\n");
   printf("\t\t\t\t  || 9)Aston Martin Vanquish S     (vq)           Rs.3,85,00,000/-  ||\n");
   printf("\t\t\t\t  ||10)Huracan Coupe              (hc)          Rs.3,97,00,000/-  ||\n");
   printf("\t\t\t\t
===========================================================================");
 }
}

void SortingPopularity(char added_item[2], int required)
{
  int flag = 0;
  struct node *tmp, *q;
  int item_priority;

  tmp = (struct node *)malloc(sizeof(struct node));

  item_priority = required;

  strcpy(tmp->data, added_item);
  tmp->priority = item_priority;
  tmp->link = NULL;

  if (Front == NULL)
  {
   Front = tmp;
  }

  else if (Front->priority < item_priority)
  {
   tmp->link = Front;
   Front = tmp;
  }
  else
  {
   q = Front;
   while (q->link != NULL)
  {
   int j = strcmp(q->data, added_item);
   if (j == 0)
   {
    q->priority = (q->priority) + (item_priority);
    printf("%d\n", q->priority);
    flag = 1;
    sortingPriority();
    break;
   }
   else
   {
    q = q->link;
   }
  }
  if (flag != 1)
  {
   while (q->link != NULL && (q->link)->priority >= item_priority)
   {
    q = q->link;
   }
   tmp->link = q->link;
   q->link = tmp;
   sortingPriority();
  }
 }
}

void displayPopularity()
{
 struct node *ptr;
 ptr = Front;
 if (Front == NULL)
 {
  printf("\n------------------------------------------------------------------------");
  printf("\n\nDatabase has just started. Please start ordering\n");
  printf("\n---------------------------------------------------------------------------");
 }
 else
 {
  printf("\n\n\t\t\t---------------------- Model By Popularity-------------------------\n\n\n");
  printf("\t\t\t\t\t\tCars\t\tSold\n\n");
  while (ptr != NULL)
  {
   printf("\t\t\t\t\t\t%s \t\t%d\n\n", ptr->data, ptr->priority);
   ptr = ptr->link;
  }
  printf("\n\n\t\t\t-------------------------------------------------------------------\n\n");
 }
}

void sortingPriority()
{

 struct node *i, *j;
 char tempo[2] = "";
 int num;
 i = Front;

 for (i = Front; i != NULL; i = i->link)
 {
  for (j = i->link; j != NULL; j = j->link)
  {
   if (i->priority < j->priority)
   {
    num = i->priority;
    i->priority = j->priority;
    j->priority = num;

    strcpy(tempo, i->data);
    strcpy(i->data, j->data);
    strcpy(j->data, tempo);
```

```
      }
    }
  }
}
```

RESULT:

In comparison to the existing systems the proposed system will be efficient in providing a market for various dealers to sell cars in a single platform and will also be give the customer a simple but efficient platform to communicate, comment, express his views on the various products offered by the dealer.

CONCLUSION:

We started with the thought of making a web application for the car's management and trading. We used Django for the front-end development as it is quite friendly with the internet applications. We used to search and sorting algorithms for the filters like sort by name, model, price, etc. We were going to use Dijkstra algorithm for finding the shortest distance but then we found out that Django is not supporting any API files and also, it's fluctuating with the database like sometimes it takes the complete data, sometimes part of it and then few times not at all. So, we then switched back to c language and command window for executing the approach. Here we don't have the good and sparking of the web application that we created before but here it's basically working properly. We have a menu that asks for the various options you want to try and other things similar to the website like sorting and searching. Overall, we wanted to do something else but eventually did something different learning quite a number of things in the journey. So, it was a quite substantial, prominent experience in doing this project of cars management system.

REFERENCES:

- https://www.agileventures.org/projects/car-sale-system
- https://nevonprojects.com/car-sales-system-mini-project/
- https://www.cardekho.com/
- https://projectsgeek.com/2016/02/car-sales-system-project-in-java.html
- http://www.proactivesoft.com/pro_CarSales.html