PARKING

MANAGER

UTILITY

**PROBLEM STATEMENT**

Parking Manager Utility is aimed to be a utility that can automate the managing of records of incoming and outgoing vehicles in the parking house.

The vehicles are categorised into 3 types : 2 wheeler(scooters, bicycles),4 wheeler light(cars),4 wheeler heavy(trucks, buses). The vehicles are registered on their arrival. Vehicles are not allowed to park for more than 8 hours. The vehicles can be parked normally or under the CCTV for security. There is a price chart available containing the per hour price for parking. One could view the details of the vehicles parked in the parking lot. The bill for the vehicle is generated according to the number of hours of stay.

Thus this utility enables drivers to reserve a parking space in the parking lot.

**SUBMODULES**

**1.Arrival of a vehicle**

In this module the vehicle details are saved and a parking space is allotted to the vehicle.

**2.Total no of vehicles arrived**

In this module the details of all the vehicles parked in the parking lot can be viewed

**3.Total parking charges of the vehicle**

In this module the bill of the vehicles are generated according to its number of hours of stay. The bill can be viewed by typing the vehicle number.

**4.Departure of the vehicle**

In this module the details of the vehicle is deleted from the file as soon as the car departs from the parking lot

**5.View price chart**

This module displays the parking price per hour for the vehicles for different categories.

**CODE**

**//Vehicle.h**

#ifndef VEHICLE\_H\_

#define VEHICLE\_H\_

class vehicle

{public:

char type[30];

int vno;

float count;

char dname[30];

char cctv\_location;

char\* entry;

char\* exit;

int input();

void output();

void bill();

void entry\_time();

void exit\_time();

};

#endif

#ifndef two\_WHEELER\_H\_

#define two\_WHEELER\_H\_

class two\_wheeler:public vehicle

{public:

float p;

two\_wheeler()

{

p=20;

}

void cal\_two();

};

#endif

#ifndef four\_WHEELER\_LIGHT\_H\_

#define four\_WHEELER\_LIGHT\_H\_

class four\_wheeler\_light:public vehicle

{public:

float price;

four\_wheeler\_light()

{

price=30;

}

void cal\_fourL();

};

#endif

#ifndef four\_WHEELER\_HEAVY\_H\_

#define four\_WHEELER\_HEAVY\_H\_

class four\_wheeler\_heavy:public vehicle

{public:

float cost;

four\_wheeler\_heavy()

{

cost=35;

}

void cal\_fourH();

};

#endif

**//Parking.cpp**

#include "vehicle.h"

#include<iostream>

using namespace std;

#include<conio.h>

#include<string>

#include<process.h>

#include<fstream>

#include <iomanip>

#include<ctime>

vehicle a;

int n;

char x;

void vehicle:: entry\_time()

{

time\_t e = time(0);

entry = ctime(&e);

cout<<"\n\n\n\t\tThe entry time of the vehicle is: "<<entry;

}

void vehicle:: exit\_time()

{

time\_t e = time(0);

exit = ctime(&e);

cout<<"\n\n\n\t\tThe exit time of the vehicle is: "<<exit;

}

void line()

{

for(int i=1;i<41;i++)

cout<<"--";

cout<<"\n";

}

int vehicle:: input()

{

system("CLS");

cout<<"\n\n\tEnter the type of vehicle (2\_wheeler,4\_wheeler\_light,4\_wheeler\_heavy) : \n";

cout<<"\t\t";

cin>>type;

string s1=a.type;

if((s1.compare("2\_wheeler"))==0||(s1.compare("4\_wheeler\_light"))==0||(s1.compare("4\_wheeler\_heavy"))==0)

{

cout<<"\n\n\tEnter the name of driver : ";

cin>>dname;

cout<<"\n\n\tEnter the vehicle no : ";

cin>>vno;

cout<<"\n\n\tEnter the no of hours of stay : ";

cin>>count;

cout<<"\n\n\tDo You Want to Park in CCTV location? Charges are double.(y/n) : ";

cin>>cctv\_location;

if(count<=8)

{

cout<<"\n\n\tParking Available! You can park your vehicle. ";

cout<<"\n\n\t\tvehicle got parked!!";

return 1;

}

else

cout<<"\n\n\tSORRY!Too many hours of stay(more than 8 hours)!Parking not possible!!";

return 0;

}

else{

cout<<"\n\n\t\t\tIncorrect type of the vehicle!";

return 0;

}

}

void two\_wheeler:: cal\_two()

{

if(a.cctv\_location=='y')

{

cout<<"\n\n\n\t\tThe total expenses during parking ";

cout<<(a.count)<<" \* "<<2\*p<<" = "<<2\*(a.count)\*p<<" rupees";

}

else

{

cout<<"\n\n\n\t\tThe total expenses during parking ";

cout<<(a.count)<<" \* "<<p<<" = "<<(a.count)\*p<<" rupees";

}

}

void four\_wheeler\_light:: cal\_fourL()

{

if(a.cctv\_location=='y')

{

cout<<"\n\n\n\t\tThe total expenses during parking ";

cout<<(a.count)<<" \* "<<2\*price<<" = "<<2\*(a.count)\*price<<" rupees";

}

else

{

cout<<"\n\n\n\t\tThe total expenses during parking ";

cout<<(a.count)<<" \* "<<price<<" = "<<(a.count)\*price<<" rupees";

}

}

void four\_wheeler\_heavy:: cal\_fourH()

{

if(a.cctv\_location=='y')

{

cout<<"\n\n\n\t\tThe total expenses during parking ";

cout<<(a.count)<<" \* "<<2\*cost<<" = "<<2\*(a.count)\*cost<<" rupees";

}

else

{

cout<<"\n\n\n\t\tThe total expenses during parking ";

cout<<(a.count)<<" \* "<<cost<<" = "<<(a.count)\*cost<<" rupees";

}

}

void vehicle:: output()

{

cout<<"\n\n\t\tThe Vehicle Name : "<<type<<endl;

cout<<"\n\n\t\tThe Driver Name : "<<dname<<endl;

cout<<"\n\n\t\tThe Vehicle No : "<<vno<<endl;

cout<<"\n\n\t\tThe Hours Of Stay : " <<count;

}

void vehicle:: bill()

{

cout<<"\n";

cout<<"\n\t\t\t\t\tBILL\n";

line();

a.output();

cout<<"\n";

line();

}

void delete\_record()

{

int n;

system("CLS");

cout<<"\n\n\t\tEnter the Vehicle number you want to get depart : ";

cin>>n;

ifstream inFile;

inFile.open("parking3.dat", ios::binary);

ofstream outFile;

outFile.open("temp.dat", ios::out | ios::binary);

while(inFile.read((char\*)&a, sizeof(a)))

{

if(a.vno!= n)

{

outFile.write((char\*)&a, sizeof(a));

cout<<"\n\n\tNo Record Found\n";

return;

}

else

{

a.exit\_time();

cout<<"\n\n\t\t Parking Record Deleted";

}

}

inFile.close();

outFile.close();

remove("parking3.dat");

rename("temp.dat", "parking3.dat");

}

int main()

{

two\_wheeler two\_w;

four\_wheeler\_light four\_wl;

four\_wheeler\_heavy four\_wh;

int login();

login();

int choice,vno1,d;

while(1) {

system("CLS");

cout << "\n\n\t\t ====== PARKING MANAGER UTILITY ======";

cout <<"\n\n ";

cout<<" \n\t\t\t======================";

cout << "\n\n\t\t\t 1. Arrival of a Vehicle";

cout << "\n\n\t\t\t 2. Total no of Vehicles Arrived";

cout << "\n\n\t\t\t 3. Total parking charges of the Vehicle";

cout << "\n\n\t\t\t 4. Departure of the Vehicle";

cout << "\n\n\t\t\t 5. View Price Chart";

cout << "\n\n\t\t\t 6. Exit Program";

cout<<" \n\t\t\t======================";

cout << "\n\n";

cout << "\t\t\t Select Your Choice ::";

cin>>choice;

switch(choice)

{

case 1 :

{

char f;

cout<<"\n\n\t\t\t==Do You Want to continue==y/n";

cin>>f;

if(f=='y')

{

ofstream f1("parking3.dat",ios::binary|ios::out|ios::app);

int ch=a.input();

if(ch!=0)

{

a.entry\_time();

f1.write((char\*)&a,sizeof(a));

}

}

break;

}

case 2 :

{

system("CLS");

cout << "\n\n\t\t=== NO VEHICLE PARKED ===";

cout << "\n";

ifstream infile("parking3.dat",ios::binary|ios::in);

int d=1;

while (infile.read((char \*)&a,sizeof (a)))

{

system("CLS");

cout << "\n";

cout<<"\n\n\tThe Vehicle position in parking lot : "<<d;

d++;

a.output();

cout<<"\n";

cout<<"\n\n\n";

system("PAUSE");

cout<<endl;

}

break;

}

case 4:

{

delete\_record();

break;

}

case 3:

{

int n;

system("CLS");

cout<<"\n\n\t\tEnter the vehicle no for which you want to see the bill : ";

cin>>n;

system("CLS");

ifstream infile("parking3.dat",ios::binary|ios::in);

while (infile.read((char \*)&a,sizeof (a)))

{

if(a.vno==n)

{

string s1=a.type;

cout << "\n";

if((s1.compare("2\_wheeler"))==0)

{

a.bill();

two\_w.cal\_two();

}

else if((s1.compare("4\_wheeler\_light"))==0)

{

a.bill();

four\_wl.cal\_fourL();

}

else if((s1.compare("4\_wheeler\_heavy"))==0)

{

a.bill();

four\_wh.cal\_fourH();

}

else

cout<<"\n\n\t\t\tIncorrect type of the vehicle!";

cout<<"\n";

cout<<endl;

}

else

cout<<"\n\n\t\t\tWrong vehicle no!";

}

break;

}

case 5:

{

system("CLS");

cout<<"\n\n\n";

cout<<setw(50)<<"PRICE CHART\n\n";

line();

cout<<setw(30)<<"Vehicle Type"<<setw(21)<<"CCTV LOCATION"<<setw(23)<<"PRICE\n";

line();

cout<<setw(30)<<"2 Wheeler"<<setw(20)<<"Yes"<<setw(22)<<2\*two\_w.p<<"Rs\n";

cout<<setw(30)<<"2 Wheeler"<<setw(20)<<"No"<<setw(22)<<two\_w.p<<"Rs\n";

cout<<setw(30)<<"4 Wheeler Light"<<setw(20)<<"Yes"<<setw(22)<<2\*four\_wl.price<<"Rs\n";

cout<<setw(30)<<"4 Wheeler Light"<<setw(20)<<"No"<<setw(22)<<four\_wl.price<<"Rs\n";

cout<<setw(30)<<"4 Wheeler Heavy"<<setw(20)<<"Yes"<<setw(22)<<2\*four\_wh.cost<<"Rs\n";

cout<<setw(30)<<"4 Wheeler Heavy"<<setw(20)<<"No"<<setw(22)<<four\_wh.cost<<"Rs\n";

line();

break;

}

case 6:

{

system("CLS");

cout<<"\n\n\n\t\t\tTHANK YOU\n";

exit(0);

break;

}

default :

{

cout<<"\n\n\t\t Invalid input";

cout<<"\n\n\t\tPress Enter to continue";

}

}

getch();

}}

int login(){

string pass ="";

char ch;

cout <<"\n\n\n\n\n\n\n\t\t\tParking Manager Utility Login";

cout << "\n\n\n\n\n\n\n\t\t\t\t\t\tEnter Password: ";

ch = \_getch();

while(ch != 13){

pass.push\_back(ch);

cout << '\*';

ch = \_getch();

}

if(pass == "pass"){

cout << "\n\n\n\n\t\t\t Access Granted! Welcome To Our System \n\n\n";

system("PAUSE");

}else{

cout << "\n\n\n\n\t\t\t\t\tAccess Aborted...Please Try Again!!\n";

system("PAUSE");

system("CLS");

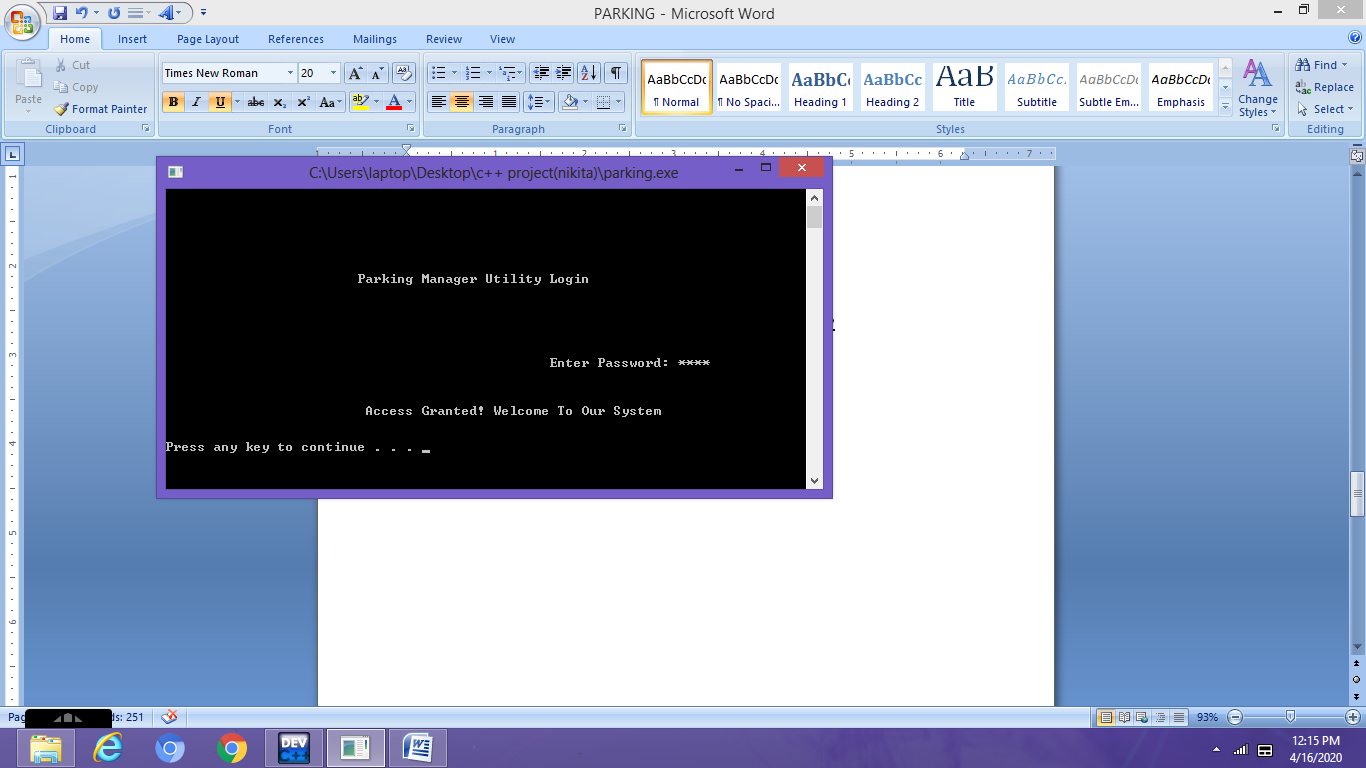
login();

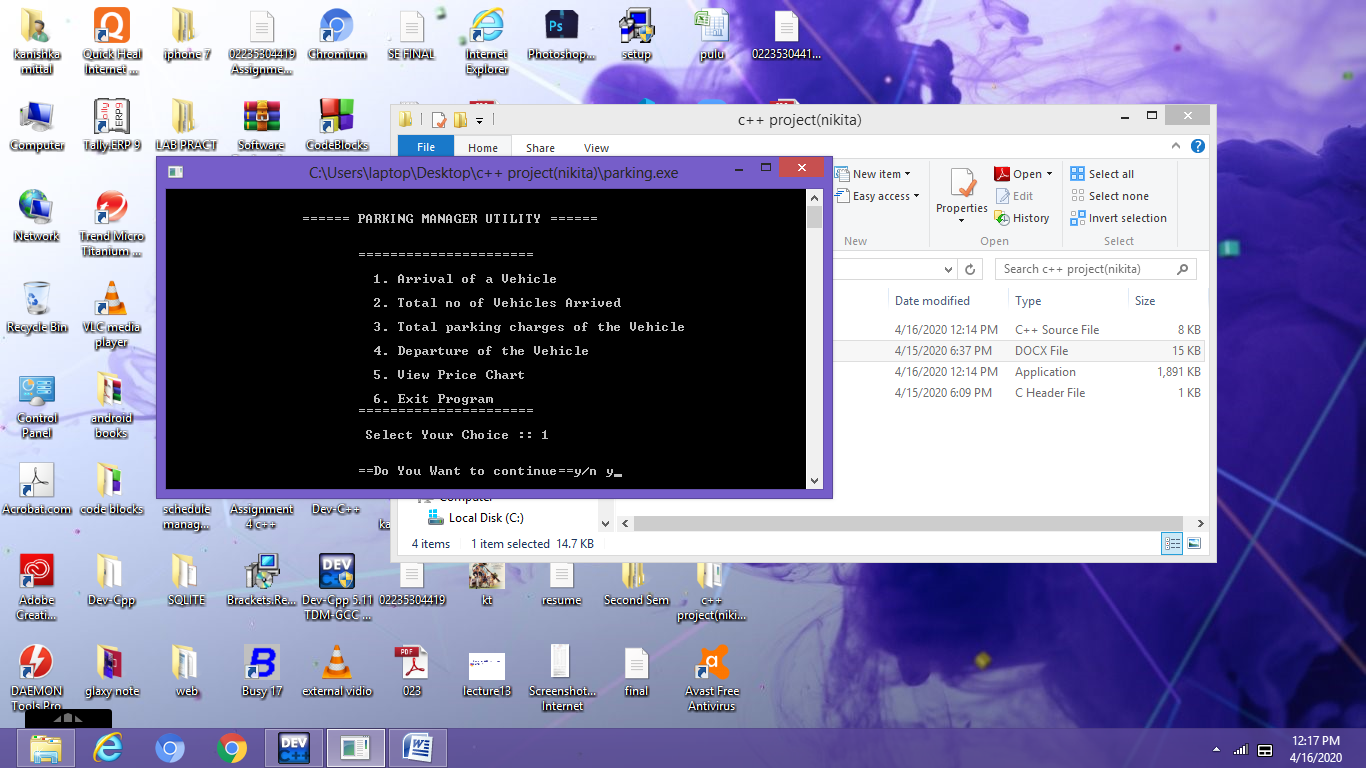
}

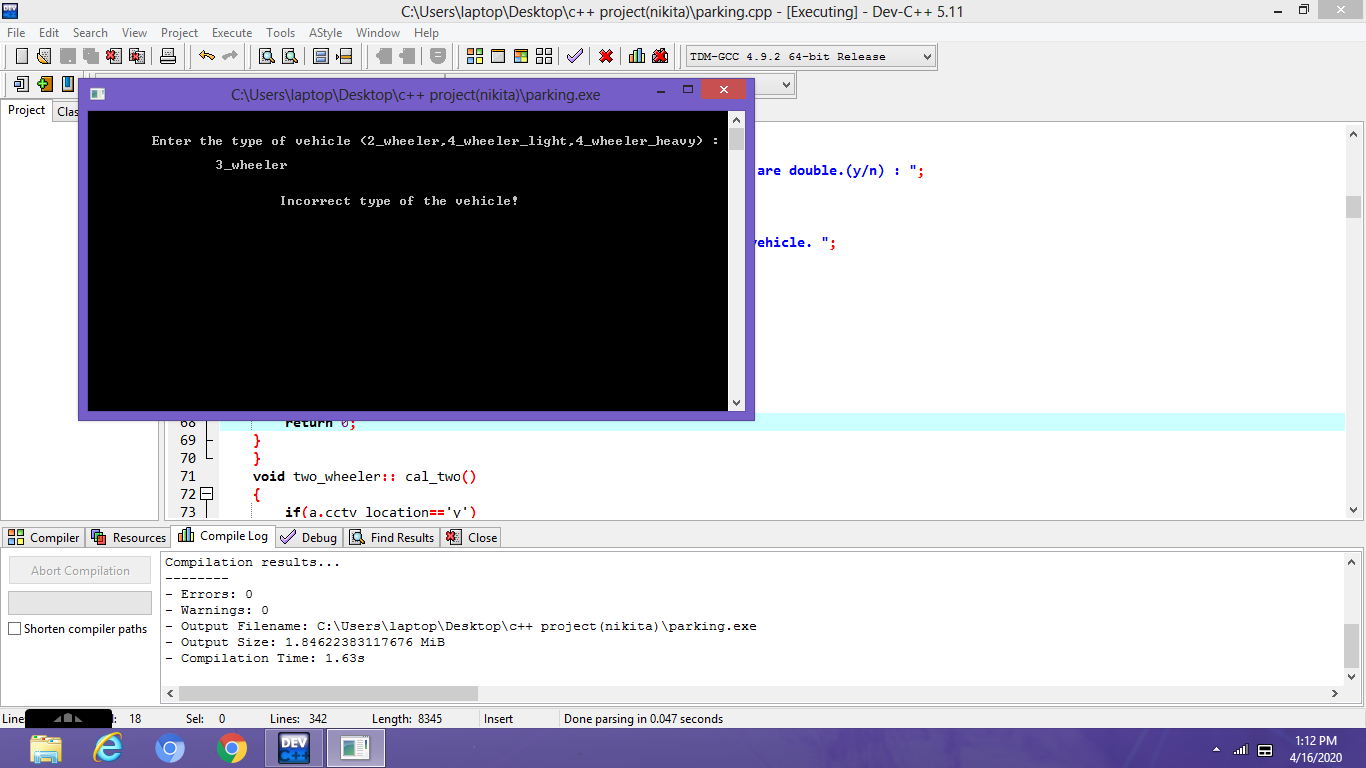
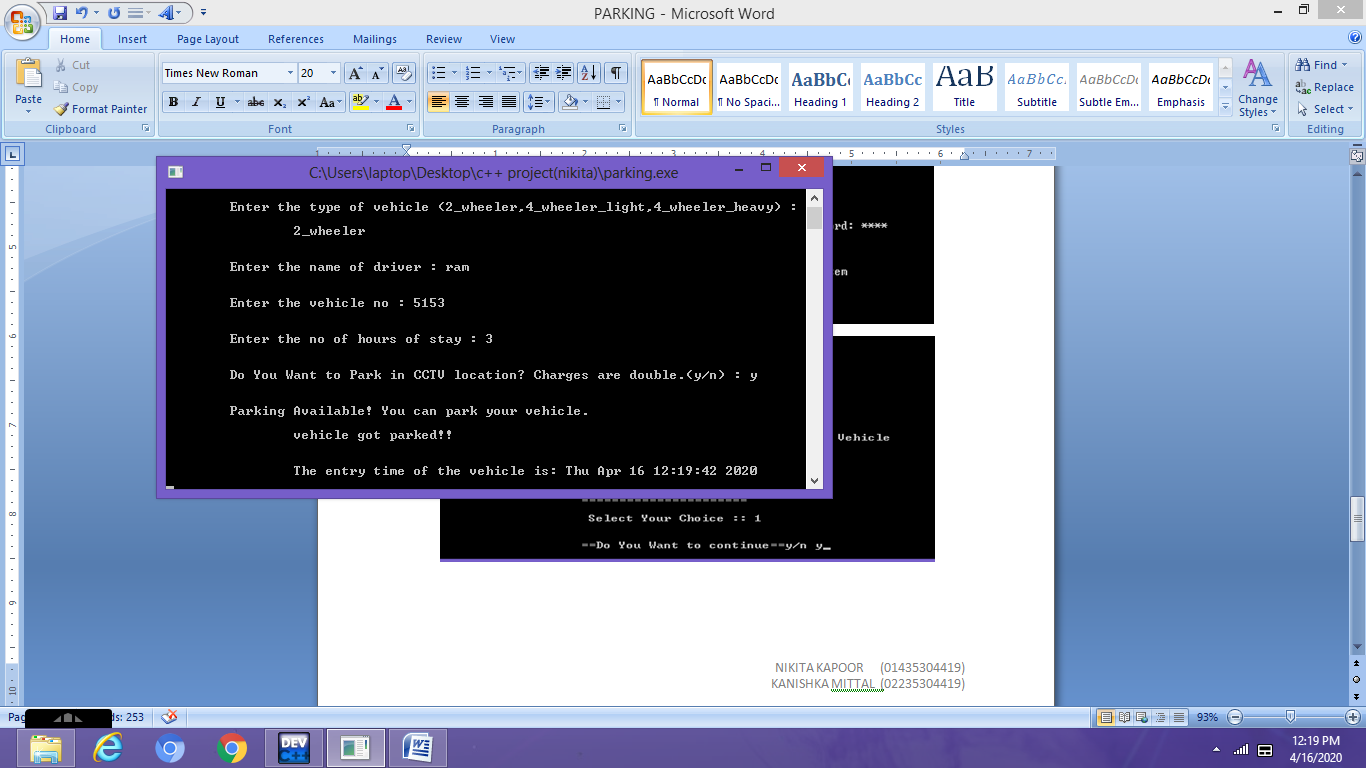
}

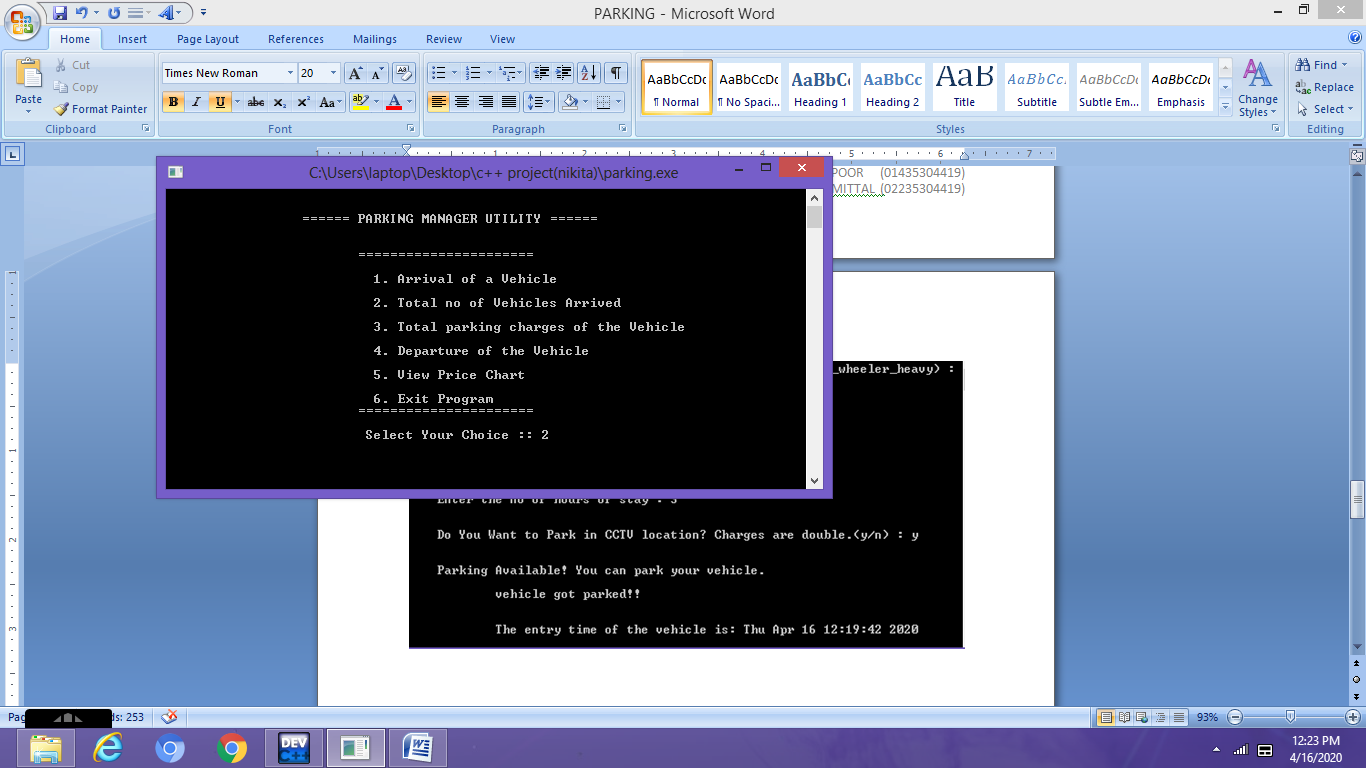
**TESTING SCREENSHOTS**

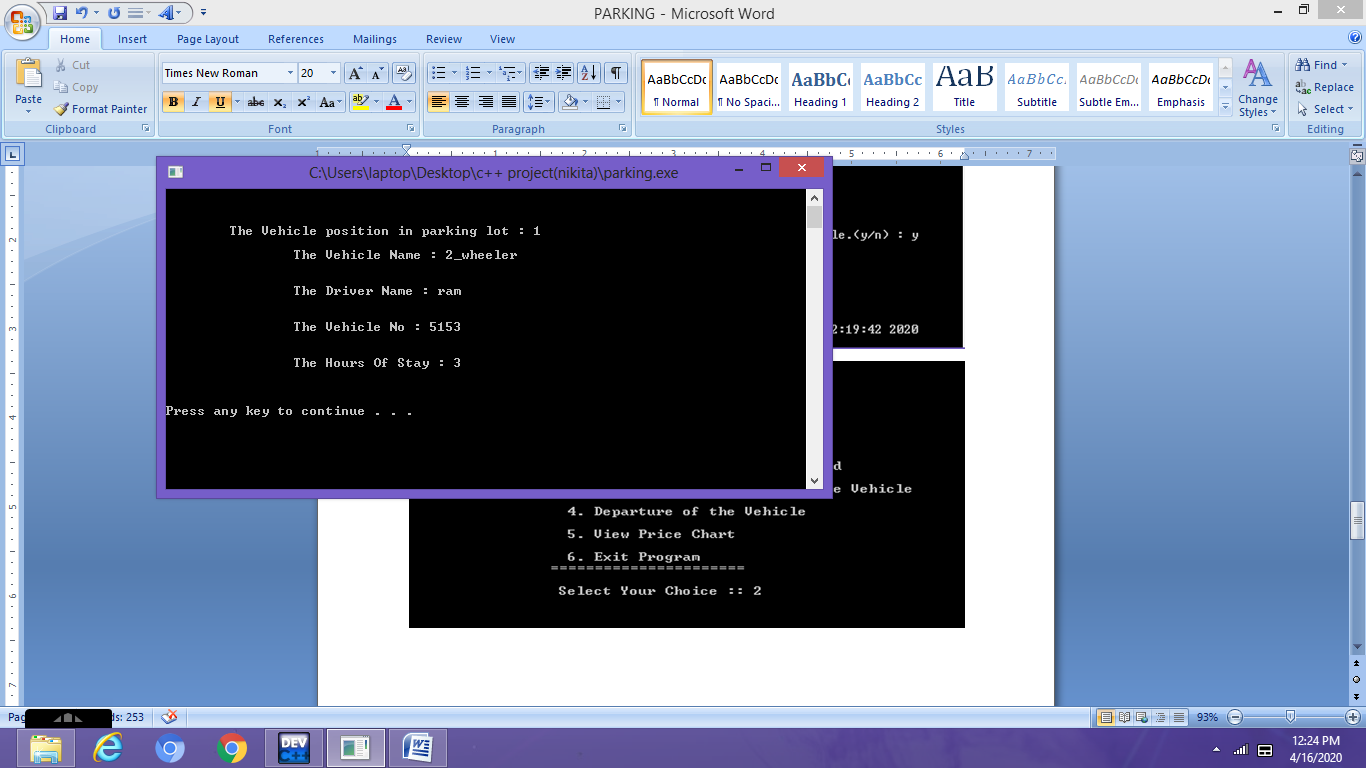
**Password:** pass

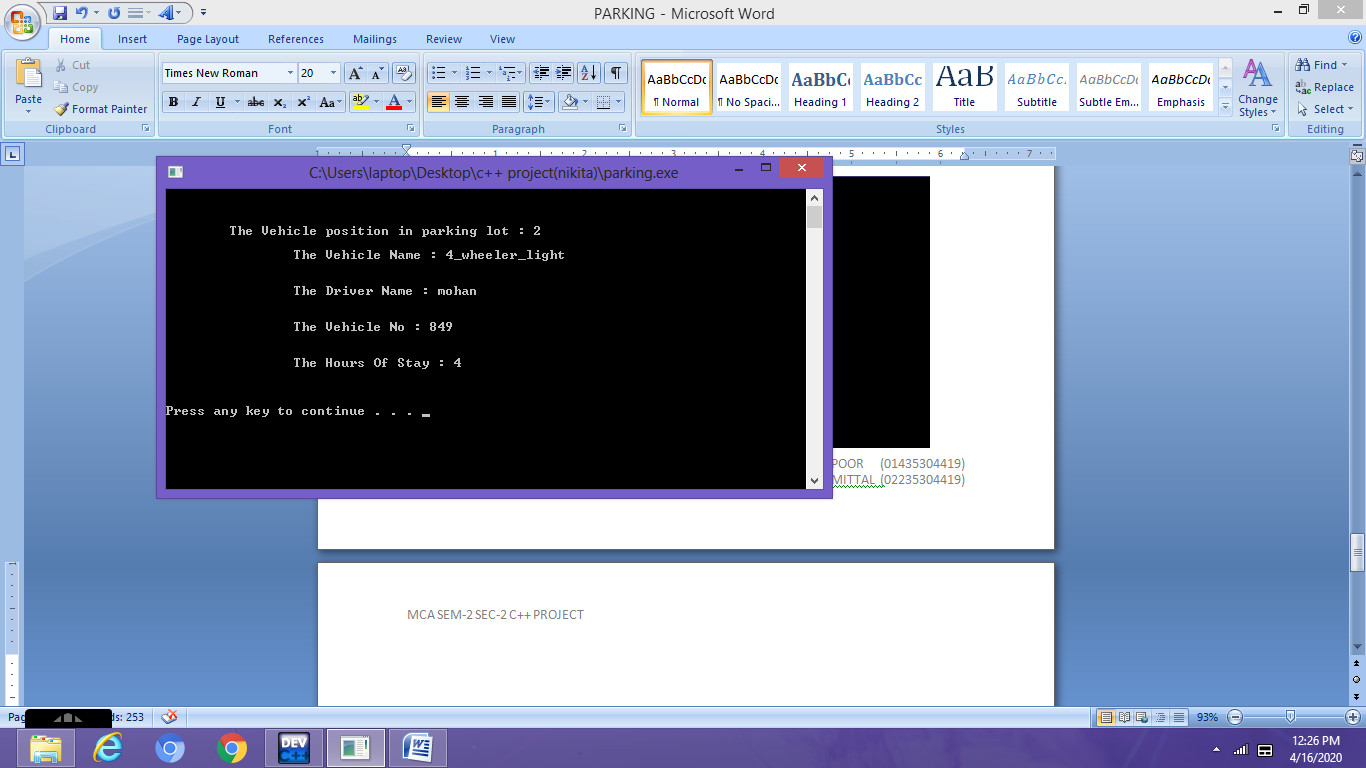


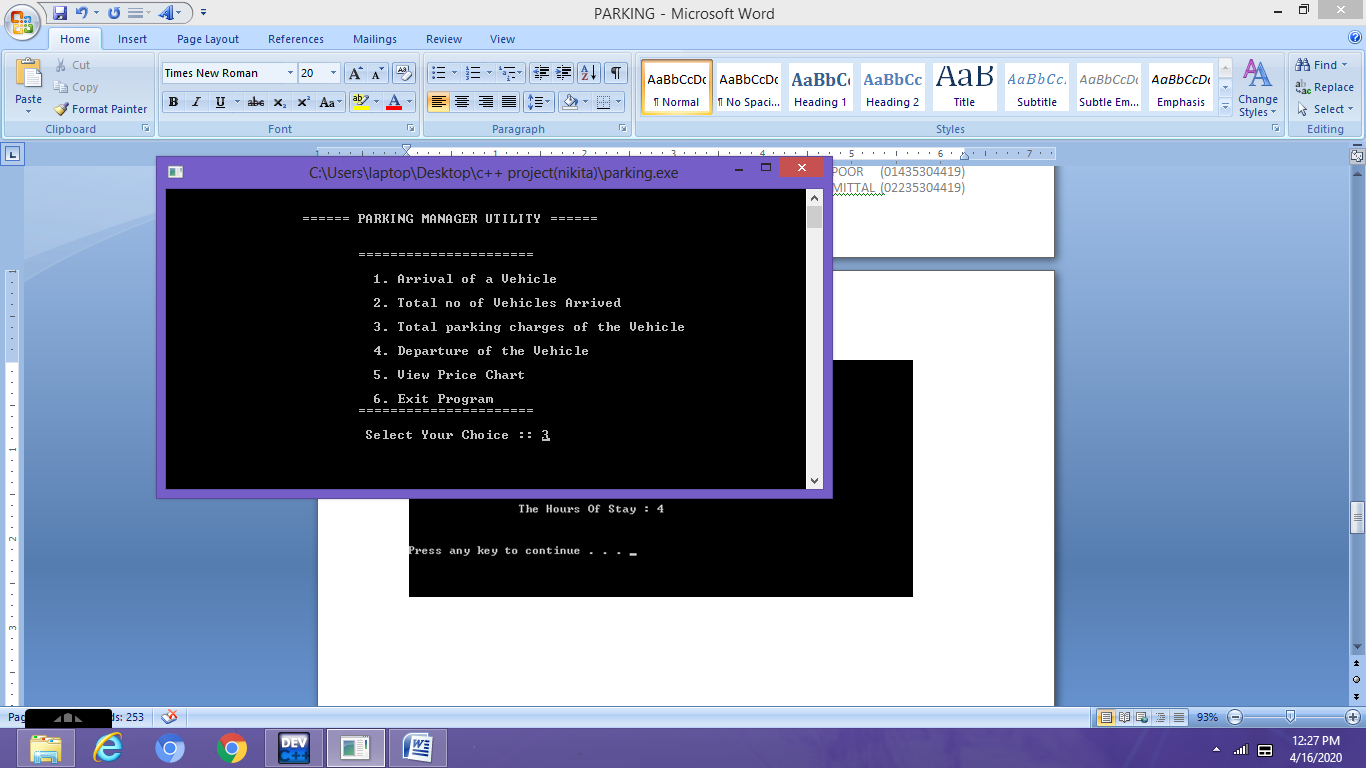


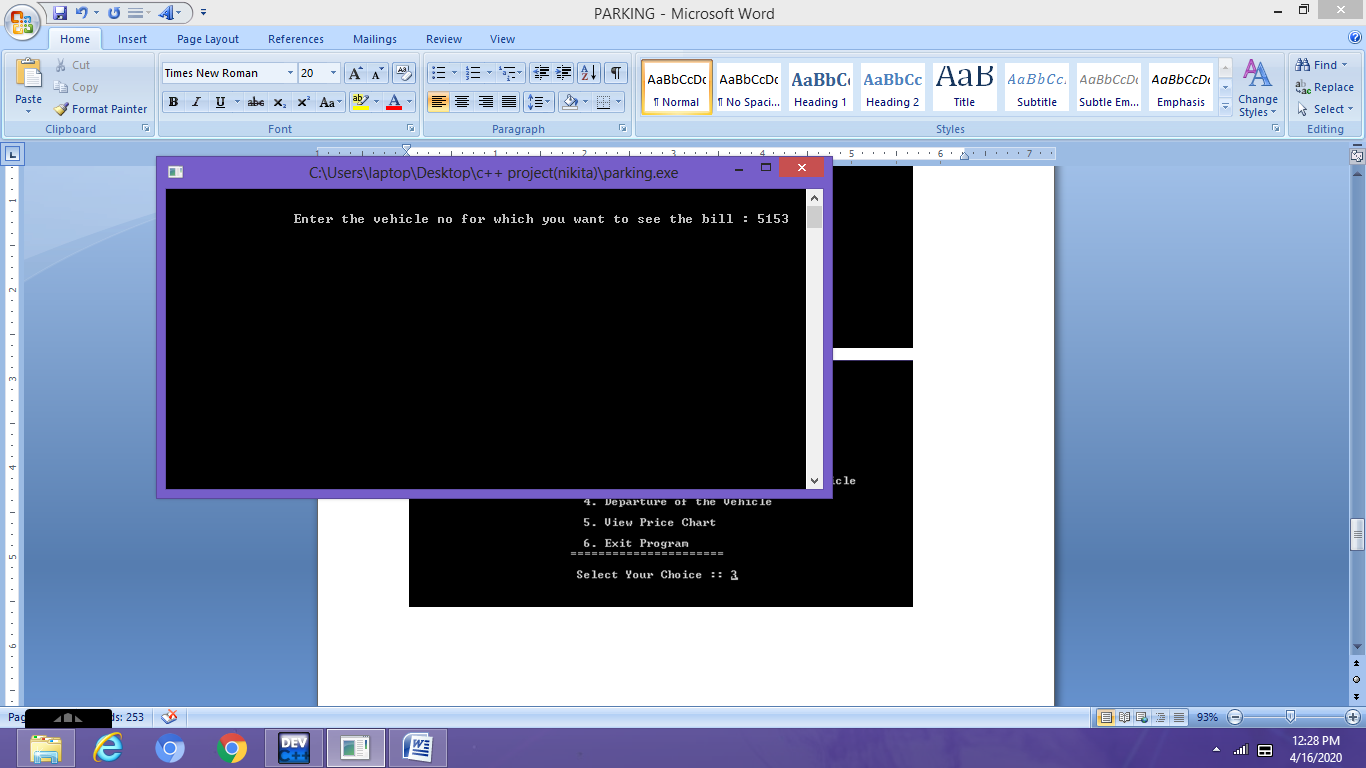


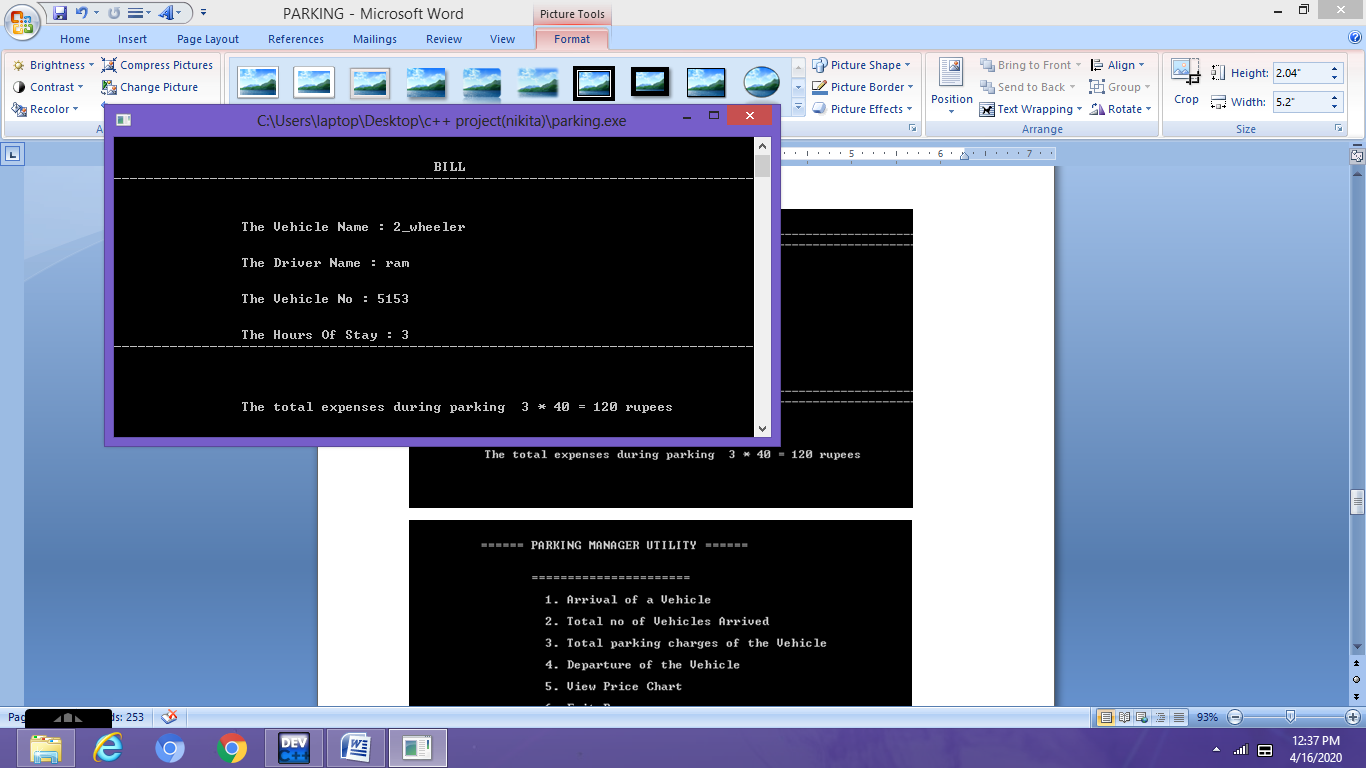


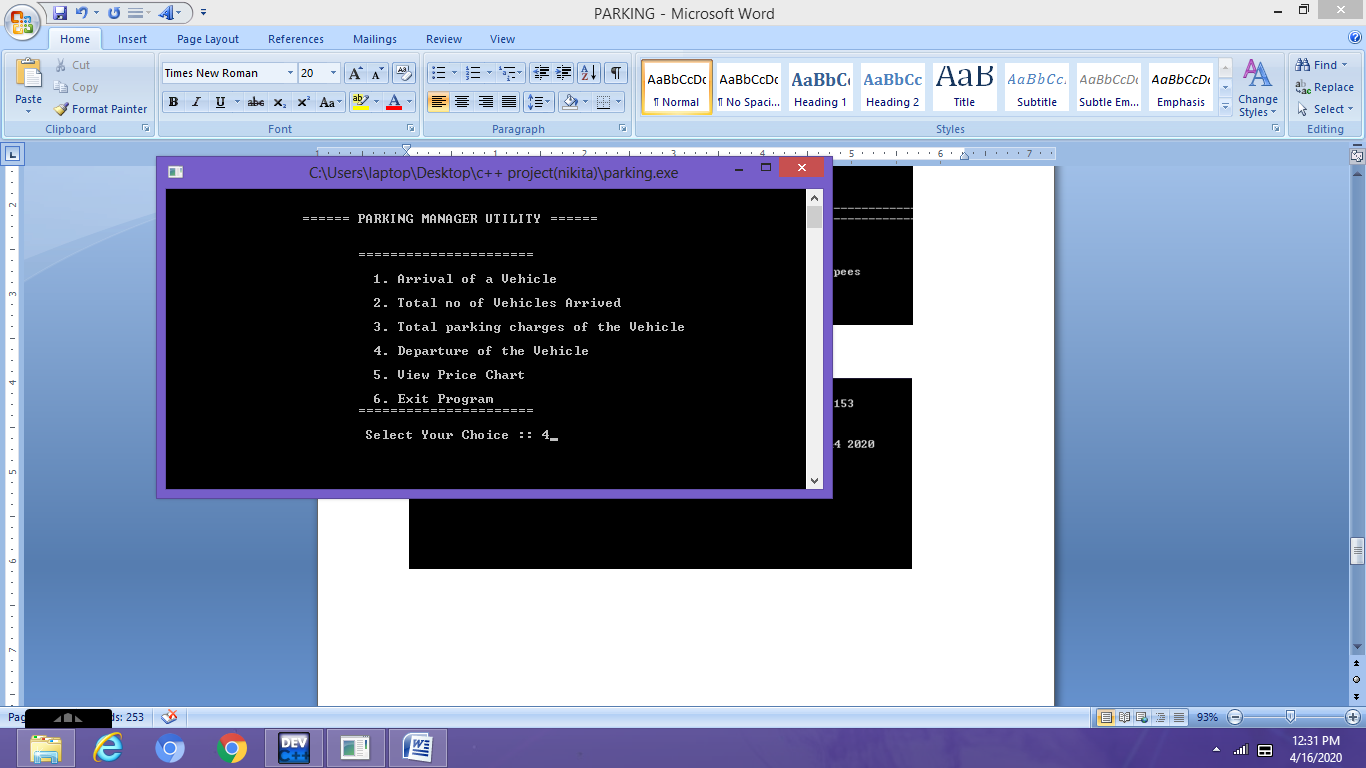


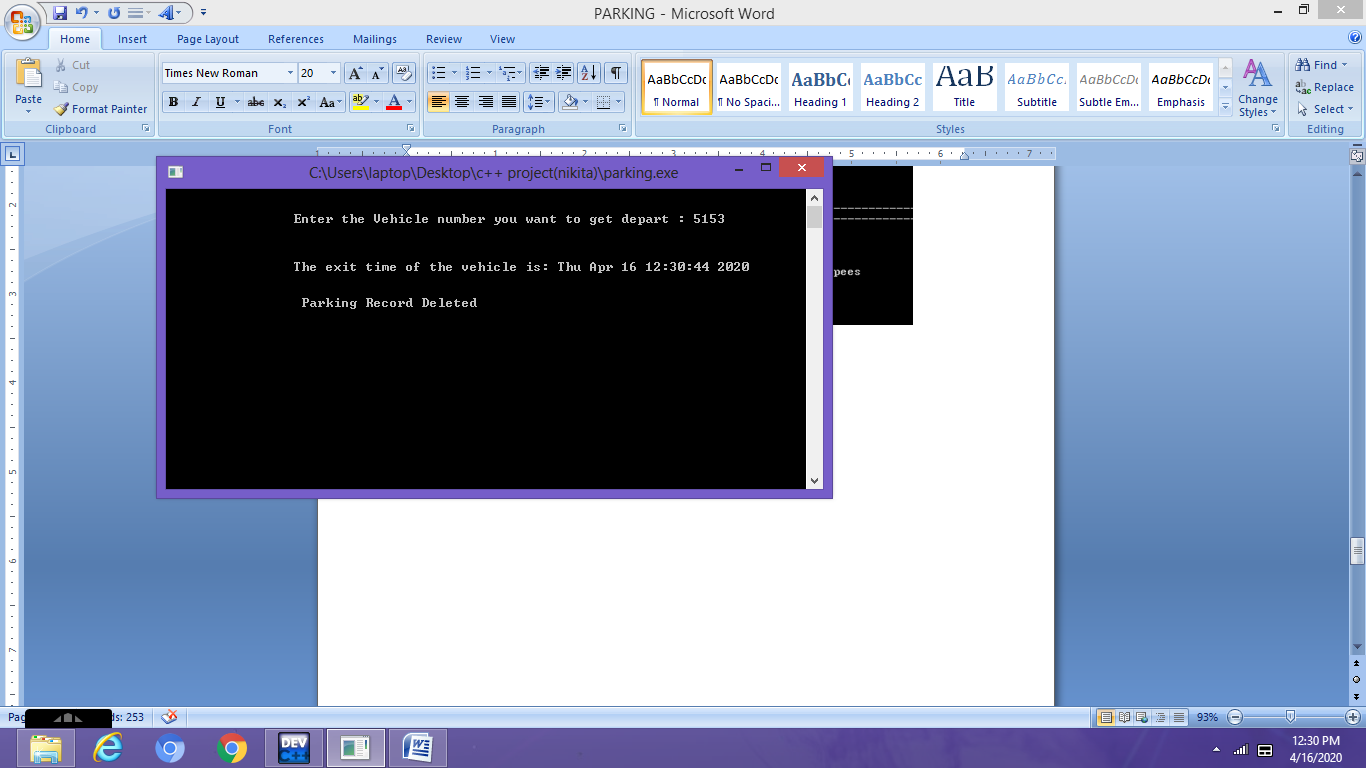


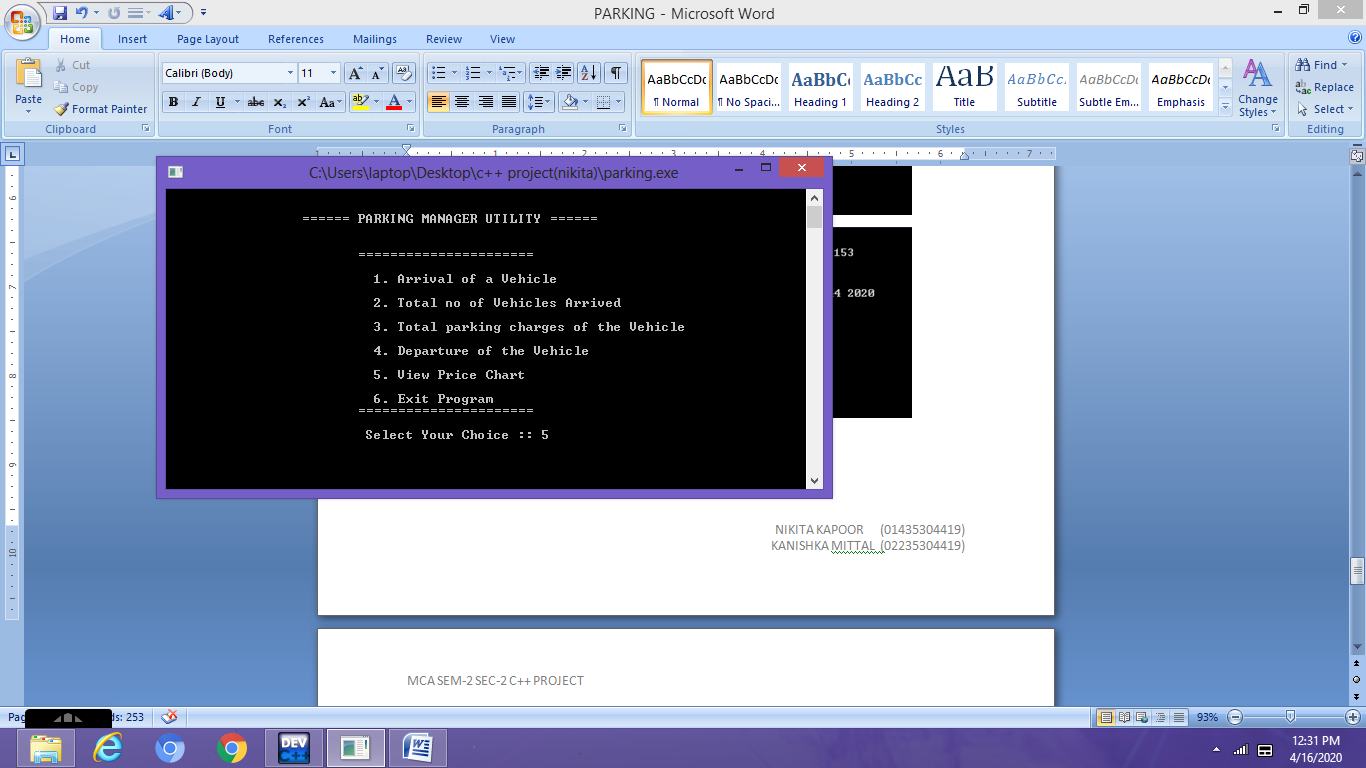


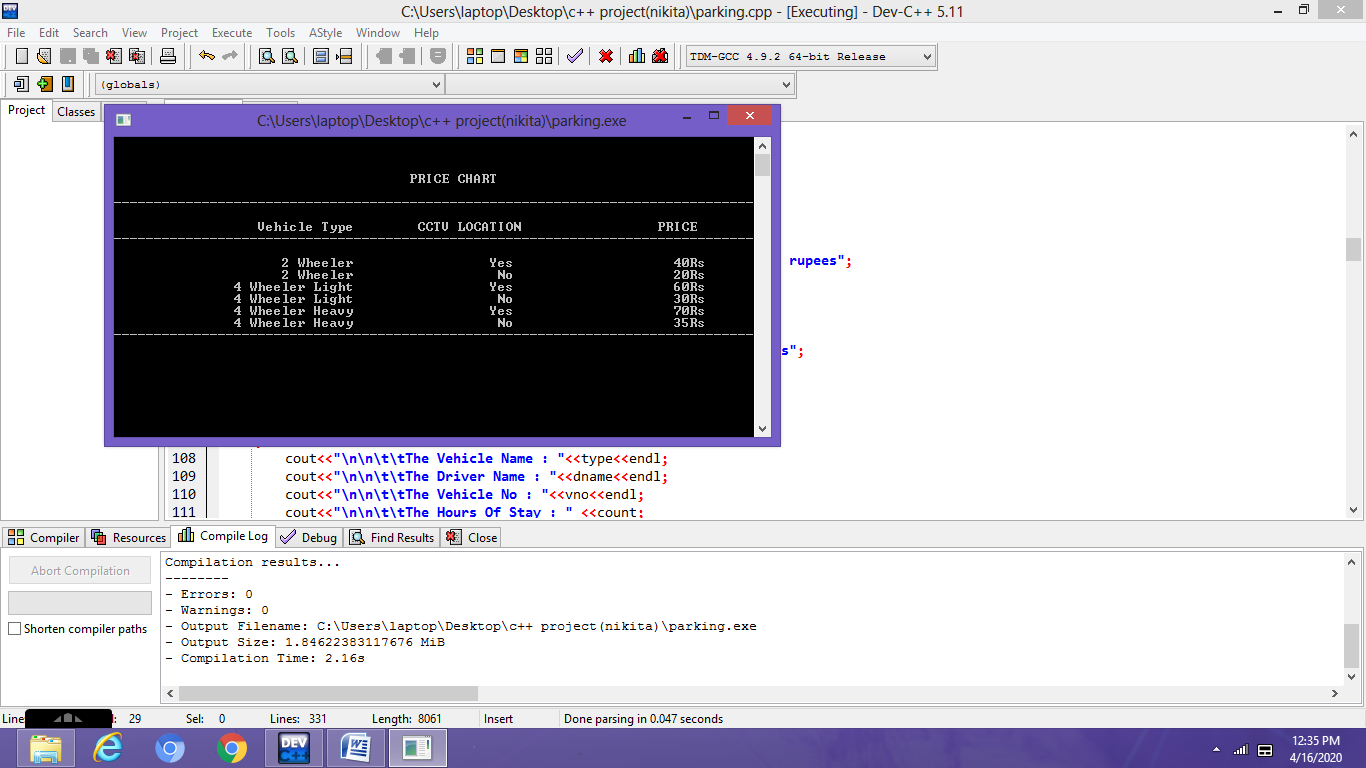


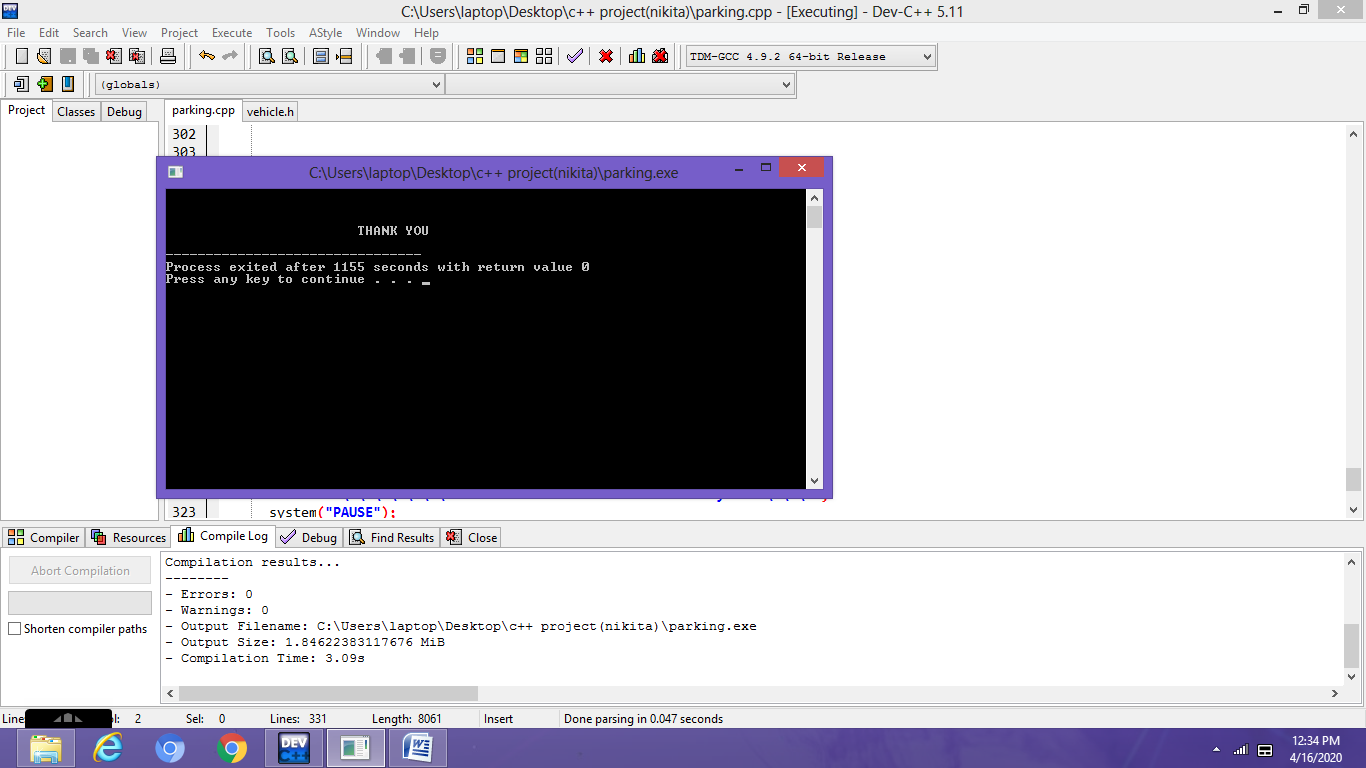












**CONCLUSION**

A successful implementation of this project would result in less traffic and chaos in crowded parking spaces like malls and business buildings where many people share a parking area.

As the Parking Manager Utility requires minimal manpower, there are minimum chances for human errors, increased security in addition to a swift and friendly car parking experience for drivers.

**FUTURE SCOPE**

Parking Manager Utility strategies would necessitate the installation of a parking guidance and information system.

The information on availability of parking space in each facility will be garnered based on the count of cars parked or from ticketing machines with the aid of sensors. The data would be sent to the central computer that will process and determine locations where parking spaces are available.