OOPS-PROJECT

CAR PARKING MANAGEMENT SYSTEM

TEAM MEMBERS DETAILS:

1.Eswar -S20210020271

2.Arjit -S20210020257

3.Mahan -S20210020260

OBJECTIVE:

The aim of implementing Parking Management System is to reduce time and increase efficiency of the current Parking Management System. In overpopulated cosmopolitan zones, parking strategies must be well implemented for management of vehicles. The system provides details of the vacant parking slots in the vicinity and reduces the traffic issues due to illegal parking in the vicinity. It is designed with an objective to meet the requirements of controlled parking that offers effortless parking tactics to the authorities. Our Parking Management System will ease people's task of finding safe parking spots in real time. The system helps an individual to pre-book the parking spot from the distant area, reducing traffic congestion and allowing a user to know the availability of parking space in advance.

INTRODUCTION:

Our parking management solution can significantly offer benefit to both the user and the parking lot owner. It has two sections-one to

allocate/give parking area by the lot owner and other to view/book a parking slot by the user. The parking slot owner can give the information regarding number of parking slots , price for a given time and get the information regarding the number of slots booked, car number, Arrival and departure time of a car and money received. The user can get the information regarding the free and can book them for a required time period. When calculating price for a given time, the user must specify whether the vehicle is a VIP or not. For VIP parking, there is a discounted rate, and for other customers, there is a standard rate.

TECHNOLOGY:

→Implementing the code using C++ (Object Oriented Programming).

→Using File system in C++ to store the data given by the user and owner.

→IDE Tool (Recommended: VS Code

OUTPUT:

OWNER:

→The owner can set the number of parking slots available.

→Owner can search, delete, update vehicle.

→Get the slot number of car by entering its number.

→Owner can view the payment status of any car by its number.

→Get the Details of all cars parked with payment details.

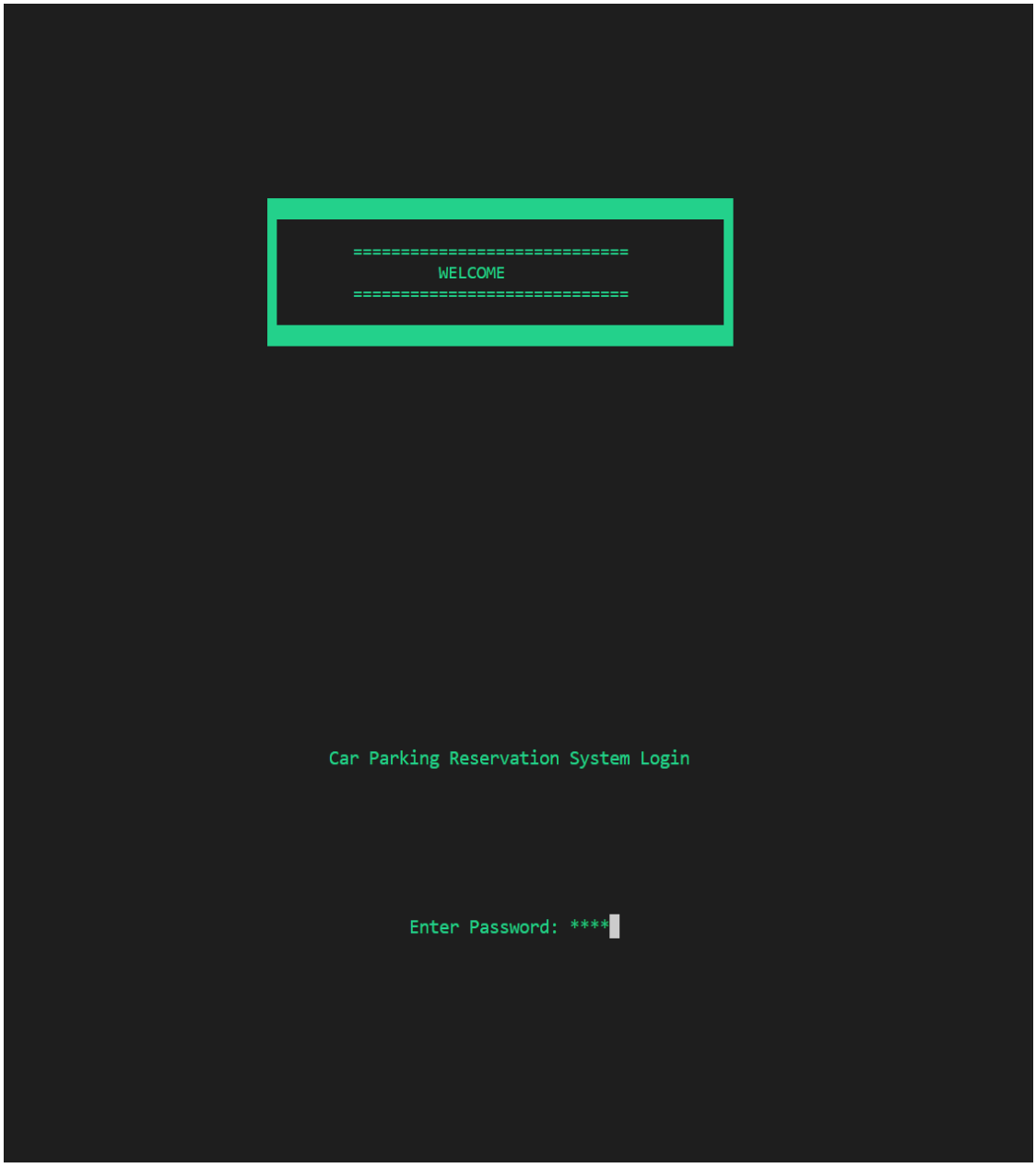
USER:

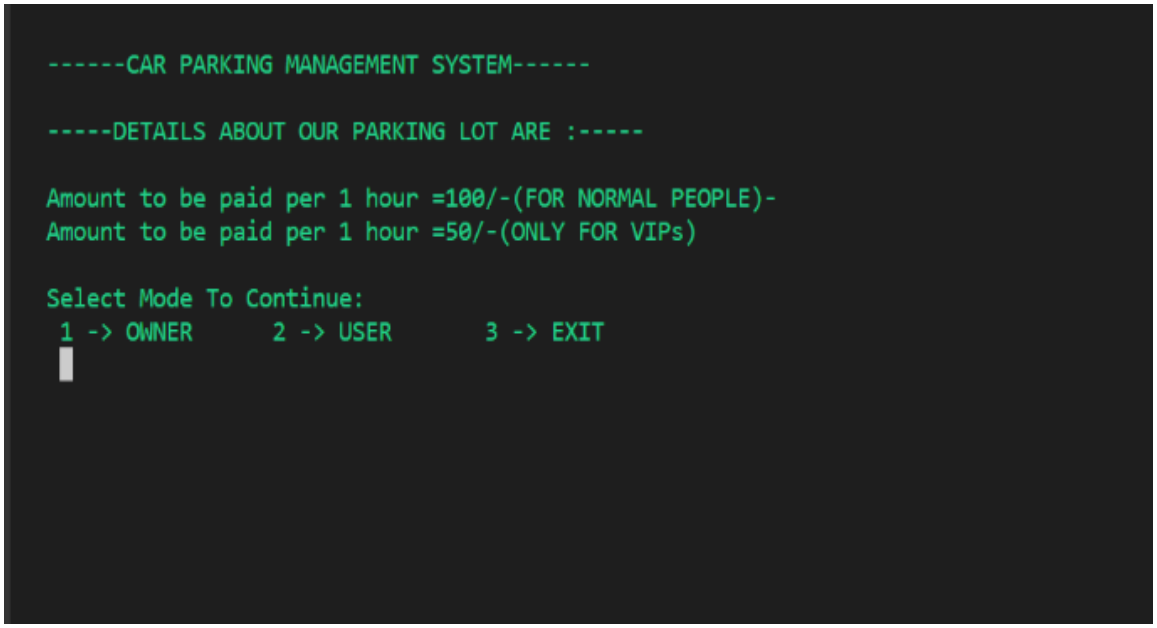
→Book a slot.

→Search his vehicle or update details.

→User also have an option to cancel his booking.

→Get payment.




------CAR PARKING MANAGEMENT SYSTEM------
-----DETAILS ABOUT OUR PARKING LOT ARE :-----
Amount to be paid per 1 hour =100/-(FOR NORMAL PEOPLE)-
Amount to be paid per 1 hour =50/-(ONLY FOR VIPs)
Select Mode To Continue:
1 -> OWNER
2 -> USER
3 -> EXIT


1. Set No Of Parking Slots Available
2. Show Vehicle Slot No
3. Search Vehicle
4. Delete Vehicle
5. Update Vehicle
6. List All Vehicle
7. Check Earned Money
8. Check Payment Status
9. Exit
Enter Your Choice:

```
----MENU FOR THE DETAILS----

1.Add new vehicle.
2.Display all vehicles.
3.Search your vehicle.
4.Delete a vehicle.
5.Update details of the user.
6.Payment Of Fee.
7.Exit.
Please enter your option :
```

CODE:

```
//                                                          Team-10

#include<conio.h>
#include<string.h>
#include<iostream>
#include <iomanip>
#include<fstream>
#include<ctime>
#include <map>
#include <math.h>
#include <vector>
#include <set>
#include <queue>
#include  <sstream>

using namespace std;

#define ll long long int
const int mod = 1e9 + 7;


class ParkingLot {
    int MAX_SIZE = 0;
```

```
    int CURRENT_CARS = 0;


public:
    class Car{
    public:
        int regNo;

        Car(const int &regNo);
    };

    bool isFull(){
        if(CURRENT_CARS==MAX_SIZE)
            return true;
        else
            return false;
    }

    bool isEmpty(){
        if(CURRENT_CARS==0 and MAX_SIZE!=0)
            return true;
        else{
            return false;
        }
    }

public:
    vector<int> availableSlots;
    map<int,int> slotToRegNo;

    map<int,int> carToSlot;


    void createParkingLot(int lotCount);

    void park(int regNo);

    int getSlotNumberFromRegNo(int regNo);

    void leave(int slotNo);
};


ParkingLot::Car::Car(const int &regNo) {
    this->regNo = regNo;


}
void ParkingLot::createParkingLot(int lotCount) {
```

```
    try {
        if(lotCount<0)
            throw lotCount;
        this->MAX_SIZE = lotCount;
    } catch (int lot) {
        cout<<"Invalid lot count"<<endl;
    }

    availableSlots.assign(MAX_SIZE+1,0);
    cout<<"Created parking lot with "<<lotCount<<" slots"<<endl;
}

void ParkingLot::park(int regNo) {
    if (this->MAX_SIZE == 0) {
        cout<<"Sorry, parking lot is not created"<<endl;
    } else if (isFull()) {
        cout<<"Sorry, parking lot is full"<<endl;
    } else {
          int vac ;
        for(int i=1;i<=MAX_SIZE;i++){
            if(availableSlots[i]==0){
                vac = i;
                break;
            }
        }
        availableSlots[vac] = 1;
        carToSlot[regNo] = vac;
        slotToRegNo[vac] = regNo;

        CURRENT_CARS++;
        cout<<"Car with vehicle registration number \""<<regNo<<"\" has been
parked at slot number "<<vac<<endl;
    }
}



int ParkingLot::getSlotNumberFromRegNo(int regNo) {
    if (this->MAX_SIZE == 0) {
        cout<<"Sorry, parking lot is not created"<<endl<<endl;
    }else if(carToSlot[regNo]){
        cout<<carToSlot[regNo]<<endl;
    }
    else {
        cout<<"Not found"<<endl;
        cout<<endl;
    }
    return carToSlot[regNo];
}
```

```
void ParkingLot::leave(int slotNo) {
    if (this->MAX_SIZE == 0) {
        cout<<"Sorry, parking lot is not created"<<endl;
    }
    else if(isEmpty()) {
        cout<<"Parking lot is empty"<<endl;
    }
    else if(availableSlots[slotNo]==1){
        availableSlots[slotNo] = 0;
        int regNo = slotToRegNo[slotNo];

        carToSlot.erase(regNo);

        CURRENT_CARS--;
        slotToRegNo.erase(slotNo);
        cout<<"Slot number "<<slotNo<<" vacated, the car with vehicle
registration number \""<<regNo<<"\" left the space."<<endl;
    }
    else{
        cout<<"Slot number "<<slotNo<<" is already empty"<<endl;
    }
}



class EarnedMoney{
    private:
        int carType;
        int total;
    public:
        EarnedMoney(){
            this->carType=0;

            this->total=0;
        }
        void setEarnedMoney(int carType,int total){
            this->carType=carType;

            this->total=total;
        }
        void showEarnedMoney(){
            cout<<"\n-----------------------------------------------------------
-----\n";
            cout<<setw(10)<<"Car|"<<setw(10)<<"Total|";
            cout<<"\n-----------------------------------------------------------
-----\n";
            cout<<setw(9)<<this->carType<<"|"<<setw(9)<<this->total<<"|";
```

```
            cout<<"\n-------------------------------------------------------------
-----\n";
        }
        void getAllEarnedMoney();
        friend void addTwoMoney(EarnedMoney,EarnedMoney&);
};
void EarnedMoney:: getAllEarnedMoney(){
    ifstream fin;
    fin.open("EarnedMoney.txt",ios_base::in|ios_base::binary);
    if(!fin){
        cout<<"\nFile Not Found";
    }
    else{
        fin.read((char*)this,sizeof(*this));
        //this->showEarnedMoney();
    }
}

void addTwoMoney(EarnedMoney oldMoney,EarnedMoney &newMoney){
    newMoney.carType+=oldMoney.carType;

    newMoney.total+=oldMoney.total;
}
class VehicleType{
    private:
        char vehicleTypeName[50];
    public:
        VehicleType(){
            strcpy(vehicleTypeName,"Vehicle");
        }
        void setVehivleType(EarnedMoney &earnedMoney,int no){

                strcpy(this->vehicleTypeName,"Car");
                earnedMoney.setEarnedMoney(no,no);
        }
        void showVehicleType(){cout<<vehicleTypeName<<"  ";}
};


class Vehicle:public VehicleType , public ParkingLot{
    public:
          int vehicleNo;
          int pay_status ;
          char type;

    public:

        char shortDescription[60];
        char description[100];
```

```
        char createdBy[50];
        char lastUpdatedBy[50];
        char createdDateTime[50];
        char lastUpdatedDateTime[50];
        EarnedMoney earnedMoney;
    public:
            Vehicle(){
            this->vehicleNo=0;
            strcpy(this->shortDescription,"default");
            strcpy(this->description,"default");
            strcpy(this->createdBy,"GRP-10: OWNER");
            strcpy(this->lastUpdatedBy,"GRP-10: OWNER");
            time_t tt;
            time(&tt);
            strcpy(this->createdDateTime,asctime(localtime(&tt)));
            strcpy(this->lastUpdatedDateTime,asctime(localtime(&tt)));
        }
        void setVehicle(){
                        cout<<"\nEnter The Vehicle Short Description: ";
                        cin.getline(shortDescription,50);
                        cout<<"\nEnter The Vehicle Description: ";
                        cin.getline(description,100);
                        strcpy(createdBy,"GRP-10: OWNER");
                        strcpy(lastUpdatedBy,"GRP-10: OWNER");
                        cout << "Are you a VIP?(Y or N): ";

                        cin >> type;
                        if(type == 'Y' || type == 'y')
                        {
                           setVehivleType(earnedMoney,50);
                        }
                        else if (type == 'n' || type == 'N')
                        {
                           setVehivleType(earnedMoney,100);

                        }
                       //date and time
                        time_t tt;
                        time (&tt);
                        strcpy(createdDateTime,asctime(localtime(&tt)));
                        strcpy(lastUpdatedDateTime,asctime(localtime(&tt)));

                }


        void showVehicle(){
            cout<<vehicleNo<<"  "<<shortDescription<<"  ";
            cout<<description<<"  ";
            showVehicleType();
```

```
            cout<<createdBy<<"  "<<lastUpdatedBy<<"  ";
            cout<<createdDateTime<<lastUpdatedDateTime<<endl;
            //earnedMoney.showEarnedMoney();
        }

        void addVehicle();
        void getAllVehicleList();
        void searchVehicle(int vno);
        void deleteVehicle(int vno);
        void updateVehicle(int vno);
};
void Vehicle:: addVehicle(){
    ofstream fout;
    fout.open("VehicleData.txt",ios_base::app|ios_base::binary);
    getch();
    fout.write((char*)this,sizeof(*this));
    fout.close();

    EarnedMoney em;
    em.getAllEarnedMoney();
    addTwoMoney(em,this->earnedMoney);
    fout.open("EarnedMoney.txt",ios_base::out|ios_base::binary);
    fout.write((char*)(&this->earnedMoney),sizeof(this->earnedMoney));
    fout.close();
    cout<<"\nRecord Added Successfully\n";
}
void Vehicle:: getAllVehicleList(){
    ifstream fin;
    int nor=0;
    fin.open("VehicleData.txt",ios_base::in|ios_base::binary);
    if(!fin){
        cout<<"\nFile Not Found";
    }
    else{
        fin.read((char*)this,sizeof(*this));
        while(!fin.eof()){
            this->showVehicle();
            nor++;
            fin.read((char*)this,sizeof(*this));
        }
        fin.close();
        if(nor==0){
            cout<<"\nFile Has No Record:";
        }
    }
}
void Vehicle:: searchVehicle(int vno){
    ifstream fin;
```

```
    int nor=0;
    fin.open("VehicleData.txt",ios_base::in|ios_base::binary);
    if(!fin){
        cout<<"\nFile Not Found";
    }
    else{
        fin.read((char*)this,sizeof(*this));
        while(!fin.eof()){
            if(this->vehicleNo==vno){
                this->showVehicle();
                nor=1;
                break;
            }
            fin.read((char*)this,sizeof(*this));
        }
        fin.close();
        if(nor==0){
            cout<<"\nRecord Not Found:";
        }
    }
}
void Vehicle:: deleteVehicle(int vno){
    ifstream fin;
    ofstream fout;
    int flag=0;
    fin.open("VehicleData.txt",ios_base::in|ios_base::binary);
    if(!fin){
        cout<<"\nFile Not Found";
    }
    else{
        fin.read((char*)this,sizeof(*this));
        fout.open("TempVehicleData.txt",ios_base::out|ios_base::binary);
        while(!fin.eof()){
            if(this->vehicleNo==vno){
                flag=1;
            }
            else{
                fout.write((char*)this,sizeof(*this));
            }
            fin.read((char*)this,sizeof(*this));
        }
        fin.close();
        fout.close();
        if(flag==0){
            cout<<"\nRecord Not Found: Hence Can Not delete";
            remove("TempVehicleData.txt");
        }
        else{
```

```
            cout<<"\nRecord Deleted:";
            remove("VehicleData.txt");
            rename("TempVehicleData.txt","VehicleData.txt");
        }
    }
}
void Vehicle:: updateVehicle(int vno){
    fstream foutIn;
    int flag=0;
    foutIn.open("VehicleData.txt",ios_base::in|ios_base::out|ios_base::ate|ios
_base::binary);
    foutIn.seekg(0);
    if(!foutIn){
        cout<<"\nFile Not Found";
    }
    else{
        foutIn.read((char*)this,sizeof(*this));
        while(!foutIn.eof()){
            if(this->vehicleNo==vno){
                //this->setVehicle();
                cout<<"\nUpdating The Data:";


                cin.ignore();
                cout<<"\nEnter The Vehicle New Short Description: ";
                cin.getline(shortDescription,50);
                cout<<"\nEnter The Vehicle Description: ";
                cin.getline(description,100);
                strcpy(createdBy,"OWNER");
                strcpy(lastUpdatedBy,"ESWAR");
                //date and time
                time_t tt;
                time (&tt);
                strcpy(lastUpdatedDateTime,asctime(localtime(&tt)));

                foutIn.seekp(foutIn.tellp()-sizeof(*this));
                foutIn.write((char*)this,sizeof(*this));
                cout<<"\nRecord Updated Successfully\n";
                flag=1;
                break;
            }
            foutIn.read((char*)this,sizeof(*this));
        }
        foutIn.close();
        if(flag==0){
            cout<<"\nRecord Not Found:";
        }
    }
```

```
}


int login(){
   string pass ="";
   char ch;
   cout <<"\n\n\n\n\n\n\n\t\t\t\t\t\tCar Parking Reservation System Login";
   cout << "\n\n\n\n\n\n\n\t\t\t\t\t\tEnter Password: ";
   ch = _getch();
   while(ch != 13){//character 13 is enter
      pass.push_back(ch);
      cout << '*';
      ch = _getch();
   }
   if(pass == "pass"){
      cout << "\n\n\n\n\t\t\t\t\t  Access Granted! Welcome To Our System
\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();
   }
   return 0;
}




class item : public Vehicle
{
   public:
   int code = vehicleNo;
   char name[15];
   int Qty;
   void get_item(int No);
   void put_item(void);
   int amt_item(void);
   void put_item2(void);
   int amount;
   char vip;

    void show_record_function(int nn);

   time_t curtime;
```

```
   int get_code(void)
   {
      return code;
   }
   int update_Qty(int num)
   {
     Qty=num;
     return Qty;
   }
   int update_Hrs(int hrs)
   {
      return hrs*100;
   }
   void update_VIP(char v)
   {
      vip=v;
   }
   void update_time()
   {
     time(&curtime);
   }
    void update_amt(int q,item obj)
    {
        if(obj.vip == 'v')
        {
            amount = q*50;
        }
        else
        {
            amount = q*100;
        }
    }

};

class display
{
   public:
   display()
   {
      cout<<"\n------CAR PARKING MANAGEMENT SYSTEM------\n"<<endl;
      cout<<"-----DETAILS ABOUT OUR PARKING LOT ARE :-----\n"<<endl;
      cout<<"Amount to be paid per 1 hour =100/-(FOR NORMAL PEOPLE)-"<<endl;
      cout<<"Amount to be paid per 1 hour =50/-(ONLY FOR VIPs)"<<endl;
      cout<<"-----------------------------------------------------\n";

   }
```

```
};

void item:: get_item(int No)
{
   code = No;
   cout <<"\n Enter Name Of Car Owner: ";
   cin >> name ;

   cout<<"Enter no of hours :";
   cin>>Qty;

   amount=Qty*100;
   cout<<"Are you a VIP ?(y/n):";
   cin>>vip;
   time(&curtime);
}
void item:: put_item(void)
{
   if(vip=='Y' || vip=='y')
   {
   amount=Qty*50;
   }
   else
   {
        amount=Qty*100;

   }
    cout<<setw(6)<<code
    <<setw(15)<<name
    <<setw(6)<<Qty
    <<setw(6)<<vip
    <<setw(12)<<amount
    <<setw(30)<<ctime(&curtime)<<endl;
}


void item:: put_item2(void)
{
   if(vip=='Y' || vip=='y')
   {
   amount=Qty*50;
   }
   else
   {
        amount=Qty*100;

   }
    cout<<setw(12)<<amount<<endl;
}
```

```
item it;
fstream file;


class add_record : public item
{
  public:
  add_record(int no)
  {
  char ch='y';
  file.open("userdata.txt",ios::app | ios::binary);
  while(ch=='y' || ch=='Y')
  {
     it.get_item(no);
     file.write((char*)&it,sizeof(it));
     cout<<" Enter N to continue: " ;
     cin>>ch;
  }
  file.close();
  }
  friend void get_item(int no);
};

class show_All : public item
{
   public:
   show_All()
   {
   file.open("userdata.txt",ios::in | ios::binary);
   if(!file)
   {
     cout<<"File not found.";
     exit(0);
   }
   else
   {
     file.read((char*)&it,sizeof(it));
     while(!file.eof())
     {
        it.put_item();
        file.read((char*)&it,sizeof(it));
     }
   }
     file.close();
     }
     friend void put_item();
```

```
};



class show_All2 : public item
{
   public:
   show_All2()
   {
   file.open("userdata.txt",ios::in | ios::binary);
   if(!file)
   {
     cout<<"File not found.";
     exit(0);
   }
   else
   {
     file.read((char*)&it,sizeof(it));
     while(!file.eof())
     {
        it.put_item2();
        file.read((char*)&it,sizeof(it));
     }
   }
     file.close();
     }
     friend void put_item2();
};




class show_record : public item
{
   public:
   show_record()
   {
   int no,
   flag=0;

   file.open("userdata.txt",ios::in | ios::binary);
   if(!file)
   {
     cout<<"File not found.";
     exit(0);
   }
   else
   {
```

```
     cout<<"Enter car number to search: ";
     cin>>no;
     file.read((char*)&it,sizeof(it));
     while(!file.eof())
     {
       if(no==it.get_code())
       {
        flag=1;
        cout<<"-----------------------------------------------------------------
-------------\n";
        cout<<setw(6)<<"Num"<<setw(15)<<"Name"<<setw(6)<<"Hrs"<<setw(6)<<"VIP"
<<setw(14)<<"Amount"<<setw(15)<<"Time"<<endl;
        cout<<"-----------------------------------------------------------------
-------------\n";
        it.put_item();
        cout<<"-----------------------------------------------------------------
-------------\n";
        break;
      }
       file.read((char*)&it,sizeof(it));
     }
     if(flag==0)
     {
       cout<<"Item not found.";
     }
   }
   file.close();
   }
   friend void put_item();
};

class delete_record : public item
{
  public:
  delete_record(int no)
  {
    ofstream file2;
   file2.open("new.txt",ios::out | ios::binary);

   file.open("userdata.txt",ios::in | ios::binary);
   if(!file)
   {
     cout<<"File not found.";
     exit(0);
   }
   else
   {
     file.read((char*)&it,sizeof(it));
```

```
     while(!file.eof())
     {
        if(no != it.get_code())
        {
           file2.write((char*)&it,sizeof(it));
        }
        file.read((char*)&it,sizeof(it));
     }
   }
   file2.close();
   file.close();

   remove("userdata.txt");
   rename("new.txt","userdata.txt");
   }

};

class modify_record : public item
{
    public:
    int no,num;
    public:
    void modify_record_car(int n)
    {
      no = n ;
    cout<<"Enter no of hrs you want to park :";
    cin>>num;
    cout<<"Are you a VIP? (Enter Y/N):";
    cin>>vip;


   file.open("userdata.txt",ios::in | ios::out | ios::binary);

   if(!file)
   {
     cout<<"File not found.";
     exit(0);
   }
   while(file.read((char*)&it,sizeof(it)))
   {
     if(it.get_code() == no)
     {
       it.update_Qty(num);
       it.update_Hrs(num);
       it.update_VIP(vip);
       it.update_time();
```

```
       int pos=sizeof(it);
       file.seekp(-pos, ios::cur);
       file.write((char*)&it,sizeof(it));
     }
   }
    file.close();
    }

};


class payment :  public item
{
    public :
        int bill = amount ;

    public:
       void bill_payment(int no,item obj)
       {
            int flag=0;
            file.open("userdata.txt",ios::in | ios::binary);
            if(!file)
           {
              cout<<"File not found.";
              exit(0);
            }
           else{
           file.read((char*)&it,sizeof(it));
           while(!file.eof())
           {
              flag = 1;
              if(no==it.get_code())
               {
               int amt , balance;


               cout << "\nEnter The Amount To Pay: ";
               cin >> amt ;
               int ac_amt = obj.amount ;
               cout << "ACtual Amount:" << ac_amt ;

               cout << "\nYour payment is successfull.";

               if (amt < ac_amt)
               {
                cout << "\nYou haven't paid the requied amount. Please pay the
balance. ";
                balance = ac_amt - amt;
                cout << "\nBalance amount = "<<balance ;
```

```
            pay_status = 0;
          }
           else if(amt == ac_amt)
           {
              cout << "\nTHANK YOU.";
              pay_status = 1;
           }
          else
           {
            cout << "\nYou Recieved Cash Back For Your Payment.";
            balance = amt-ac_amt;
            cout <<"CashBack Amount : "<< balance ;
            pay_status = 1;
           }
            }
           file.read((char*)&it,sizeof(it));
      }
           if(flag==0)
            {
               cout<<"Item not found.";
             }
      }

        file.close();

   }

  void payment_status(int no)
  {
         int flag=0;
       file.open("userdata.txt",ios::in | ios::binary);
       if(!file)
      {
         cout<<"File not found.";
         exit(0);
       }
      else{
      file.read((char*)&it,sizeof(it));
      while(!file.eof())
      {
         flag = 1;
       if(no==it.get_code())
          {
               if(pay_status == 1)
               {
                   cout << "\nPAID.";
               }
               else if(pay_status == 0)
```

```
                    {
                        cout << "\nNot Paid.";
                    }
                }

                file.read((char*)&it,sizeof(it));
            }
                if(flag==0)
                 {
                    cout<<"Item not found.";
                  }
            }
              file.close();
        }


};

void item:: show_record_function(int no)
   {

   int flag=0;

   file.open("userdata.txt",ios::in | ios::binary);
   if(!file)
   {
     cout<<"File not found.";
     exit(0);
   }
   else
   {
    cout<<"Your bill is displayed as follows :"<<endl;
     file.read((char*)&it,sizeof(it));
     while(!file.eof())
     {
       if(no==it.get_code())
       {
        flag=1;
        cout<<"-------------------------------------------------------------------
-------------\n";
        cout<<setw(6)<<"Num"<<setw(15)<<"Name"<<setw(6)<<"Hrs"<<setw(6)<<"VIP"
<<setw(14)<<"Amount"<<setw(15)<<"Time"<<endl;
        cout<<"-------------------------------------------------------------------
-------------\n";
        it.put_item();
        cout<<"-------------------------------------------------------------------
-------------\n";
        break;
```

```
      }
       file.read((char*)&it,sizeof(it));
     }
     if(flag==0)
     {
       cout<<"Item not found.";
     }
   }
   file.close();
     //friend void put_item();

   }




int menu(){
    int ch;
    cout<<"\n\t\t\t\t\t\t\t**WELCOME OWNER**";
    cout<<"\n\n1. Set No Of Parking Slots Available";
    cout<<"\n2. Show Vehicle Slot No";
    cout<<"\n3. Search Vehicle";
    cout<<"\n4. Delete Vehicle";
    cout<<"\n5. Update Vehicle";
    cout<<"\n6. List All Vehicle";
    cout<<"\n7. Check Earned Money";
    cout<<"\n8. Check Payment Status";
    cout<<"\n9. Exit";
    cout<<"\nEnter Your Choice: ";
    cin>>ch;
    return(ch);
}



int main(){

    //loading animation
    int count = 0;
    cout << "\n\n\n\n\n\n\n\n\n\n\n";
    cout << "\t\t";
    cout << "\t\t\t\t\tLoading. ";
    for (count; count < 12; ++count)
    {
        system("color 0B");
        cout << ". ";
        for (int j = 0; j <= 100000000.7; j++)
            ;
```

```
    }

    system("cls");
    system("color 0A");

    //introduction
    cout << "\n\n\n\n\n\n\n\n\n";
    cout <<
"\n\t\t\t\t\t\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\
xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xd
b\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\n";
    cout <<
"\t\t\t\t\t\xdb                                                \xdb\n";
    cout <<
"\t\t\t\t\t\xdb        ============================            \xdb\n";
    cout <<
"\t\t\t\t\t\xdb                WELCOME                         \xdb\n";
    cout <<
"\t\t\t\t\t\xdb        ============================            \xdb\n";
    cout <<
"\t\t\t\t\t\xdb                                                \xdb\n";
    cout <<
"\t\t\t\t\t\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xd
b\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\
xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\xdb\n\n\n";
    getch();
    system("cls");

    login();

    system("cls");
    int mode;


    Vehicle vehicle;
    EarnedMoney earnedMoney;
    int vno,q;

    ParkingLot parkingLot;

    display obj;

    payment pa ;

    modify_record mo;

   int option;
```

```
while(1)
{

   cout << "\n1 -> OWNER  \n2 -> USER \n3 -> EXIT \n ";
   cout << "\nSelect Mode To Continue: ";
   cin >> mode ;
 switch (mode)
 {
   case 1:
 {
      int breakInfiniteloop=0 ;


 while(1){
     system("cls");
     switch(menu()){
         case 1:
              int lotcount;
              cout<<"\nEnter no of parking slots available:";
              cin>>lotcount;
              parkingLot.createParkingLot(lotcount);
              getch();
             break;


         case 2:
              cout << "\nEnter Vehicle No To Find Slot No:";
              int regno;
              cin>>regno;
              parkingLot.getSlotNumberFromRegNo(regno);
              getch();
             break;

         case 3:
             cout<<"\nEnter The Vehicle Number To Search: ";
             cin>>vno;
             vehicle.searchVehicle(vno);
             getch();
             break;
         case 4:
             cout<<"\nEnter The Vehicle Number To Delete: ";
             cin>>vno;
             if(parkingLot.isEmpty())
             {
                 cout << "\nParking Slot Is Already Empty.";
             }
             else{
             int slotNo = parkingLot.getSlotNumberFromRegNo(vno);
```

```
            parkingLot.leave(slotNo);
            vehicle.deleteVehicle(vno);
            }
            getch();

            break;
        case 5:
            cout<<"\nEnter The Vehicle Number To Update: ";
            cin>>vno;
            vehicle.updateVehicle(vno);

            mo.modify_record_car(vno);
            q= it.update_Qty(mo.num);
            it.update_amt(q,it);

            getch();
            break;
        case 6:
            vehicle.getAllVehicleList();
            getch();
            break;
        case 7:
            earnedMoney.getAllEarnedMoney();
            earnedMoney.showEarnedMoney();
           getch();
             break;

        case 8:
            int nm;
            cout << "\nEnter Car Number To know Payment Status: ";
            cin >> nm ;
            pa.payment_status(nm);
            getch();
            break;
        case 9:
            breakInfiniteloop=1;
            break;

        default:
            cout<<"\nInvalid Choice:";
            getch();
    }
    if(breakInfiniteloop==1){
        break;
    }
}
   getchar();
```

```
      break;
  }
   case 2:
  {     int breakloop2 =0 ;

while(1)
{ //system("cls");
  cout<<"\n\t\t\t\t\t\t\t**WELCOME USER**";
  cout<<"\n\n----MENU FOR THE DETAILS----\n"<<endl;
  cout<<"1.Add new vehicle."<<endl;
  cout<<"2.Display all vehicles."<<endl;
  cout<<"3.Search your vehicle."<<endl;
  cout<<"4.Delete a vehicle."<<endl;
  cout<<"5.Update details of the user."<<endl;
  cout<<"6.Payment Of Fee. " << endl;
  cout<<"7.Exit."<<endl;



cout<<"Please enter your option : "<<endl;
cin>>option;

switch(option)
{
   case 1:
   {

     if(parkingLot.isFull())
             {
                 cout << "\nNo Free Slots Available.";
             }
     else{
             cout<<"\nEnter The Vehicle Number: ";
             cin>>vehicle.vehicleNo;

             cin.ignore();
             parkingLot.park(vehicle.vehicleNo);
             vehicle.setVehicle();
             vehicle.addVehicle();
             add_record obj(vehicle.vehicleNo);
             }


      cout<<"Press any key for Main Menu...";
      getchar();

      break;
   }
   case 2:
```

```
      {
        cout<<"-----------------------------------------------------------------
-------------\n";

        cout<<setw(6)<<"Num"<<setw(15)<<"Name"<<setw(6)<<"Hrs"<<setw(6)<<"VIP"
<<setw(14)<<"Amount"<<setw(15)<<"Time"<<endl;
        cout<<"-----------------------------------------------------------------
-------------\n";
        show_All obj;
        cout<<"-----------------------------------------------------------------
-------------\n";

        cout<<"Press any key for Main Menu...";
        getchar();
        break;
      }
      case 3:
      {
        show_record obj;
        cout<<"Press any key for Main Menu...";
        getchar();
        break;
      }
      case 4:
      {
        cout<<"\nEnter The Vehicle Number To Deleted: ";
        cin>>vno;
        if(parkingLot.isEmpty())
                {
                    cout << "\nParking Slot Is Already Empty.";
                }
                else{
                int slotNo = parkingLot.getSlotNumberFromRegNo(vno);

                parkingLot.leave(slotNo);
                vehicle.deleteVehicle(vno);
                 delete_record obj(vno);
                }
       cout<<"Press any key for Main Menu...";
        getchar();
        break;
      }
       case 5:
      {
         cout<<"\nEnter The Vehicle Number To Updated: ";
         cin>>vno;
         vehicle.updateVehicle(vno);
```

```
        mo.modify_record_car(vno);
        q= it.update_Qty(mo.num);
        it.update_amt(q,it);

        cout<<"Press any key for Main Menu...";
        getchar();
        break;
    }
    case 6:
    {
        int nn;
        cout << "\nEnter Vehicle Number To Pay Bill: ";
        cin >> nn;
        it.show_record_function(nn);
        pa.bill_payment(nn,it);
        cout<<"Press any key for Main Menu...";
        getchar();
        break;
    }
    case 7:
    {
        breakloop2 = 1;
        break;
    }
    default:
    {
      cout<<"Incorrect option entered...Please enter a vaild option";
      getchar();
      break;
    }
 }
   if(breakloop2==1){
         break;
      }
}
      getchar();
      break;
  }
  case 3:{
       exit(0) ;
  }
  default:
  {
      cout<<"Incorrect option entered...Please enter a vaild option";
      getchar();
      break;
  }
 }
```

```
    }
     return(0);
}
```

CONTRIBUTION:

BY ESWAR: Created class for owner and coded the functions for payment and saving the data into files. Created a switch case for owner to get desired data.

BY ARJIT: Created class for user and coded the functions used by user. Created a switch case for user to enter and get information.

BY MAHAN: Created the functions Search, Delete, Update and View information regarding slots. Storing and retrieving the entered details from files.

CONCLUSION: The main aim of this project is to reduce the time spent by car owners to search the free slot ang park their car. With this project, Both the car driver and the parking area owner get benefitted.

*** THANK YOU ***