A collage of a building

Description automatically generated with medium confidence

Course name: Computer programming

Course Code: CSE131

**Under the supervision of:**

[***Kyrelloss Nashaat Samy Bottros***](https://eng.asu.edu.eg/staff/k.nashaat.kn)

***Summer Semester – 2023***

**Team members: ID:**

**Ahmed Mohammed Elsayed Tabbash 20P1076**

**Khalid Mohammad Saleh 16P8236**

**Emad ahmed shehata 21p0038**

All the files required to run this code:

// this code runs all the requirements for modules 1&2&3&4&5 except the average rental time

// it won’t work without the txt.files

#include <iostream>

#include <conio.h>// it takes input from keyboard and displays it on screen

#include <fstream>//allowing you to read from and write to files

#include <windows.h>//used to access the Win32 API functions and it makes it easier for the user to use the in-built functionality.

#include <string.h>//contains some useful string functions that can be directly used in a program

#include <stdio.h>//contains the information needed to include input/output routines in our program. For instance, printf, scanf

#include <time.h>// contains time and date function declarations to provide standardized access to time/date manipulation and formatting

#include <cstdlib>//defines a collection of functions and macros to facilitate efficient, high-performing, standardized C++ code

#include <vector> //sequence containers representing arrays that can change their size during runtime.

#include <algorithm>//test whether first set is subset of another or not

using namespace std;

void mostrentedcar();

void allreservations();

void customerData();

int carCount();

int custCount();

int rate(int, int);

void newCustData();

void carData ();

void admin();

void user();

void menu();

void art();

void exitArt();

void rentedcarsperday();

void newCarData();

void delCar();

void displayCar();

void existingCust();

void load();

void load\_CHECK();

void gotoxy(int , int );

void time();

void Boarder();

int countUser();

void tNc();

void readUserPass();

void availCar();

int countAvail();

const string currentDateTime(); // we use const to tell the compiler to prevent the programmer from modifying it

const string Date();

void date();

void dispAvailCar();

void GotoXY(int , int );

void load\_EXIT();

void load\_UPDATE();

void newUserPass();

void password();

void resetAvail();

void showCarData();

void welcome();

struct car

{

char plate\_num[10];

char brand[20];

char model[20];

float capacity;

char colour[20];

float rate\_per\_hour;

float rate\_per\_half;

float rate\_per\_day;

char transmission[6];

};

struct customer

{

int id;

char name[100];

char phone[15];

char ic[15];

char address[1500];

};

struct pass

{

int ID;

char passWord[20];

};

pass userPass[1000];

car rent[1000];

customer cust[1000];

car avail[1000];

const string currentDateTime() // ensure that a string remains constant and prevent accidental modifications

{

time\_t now = time(0);

struct tm tstruct;

char buf[80];

tstruct = \*localtime(&now);

strftime(buf, sizeof(buf), "%d/%m/%Y TIME: %X", &tstruct);

return buf; // acts as a temporary placeholder

}

const string Date()

{

time\_t now = time(0);

struct tm tstruct;

char buf[80];

tstruct = \*localtime(&now);

strftime(buf, sizeof(buf), "%d/%m/%Y TIME: %X", &tstruct);

return buf; //`buf` is the value being returned and allows you to pass the value of a variable back to the calling function

}

void time()

{

int i=0;

while( i<5)

{

Boarder();

gotoxy(20,20); //use to print the statement at appropriate position on the screen

std::cout <<"\n\n\n\n\n\t |\t\t\t\tDATE: "<<currentDateTime() << std::endl;

Sleep(900);

system("cls");

i++;

}

menu();

}

void date()

{

cout <<"\n\t |\t\t\t\t\t\tDATE: "<<Date()<<endl;

}

void readUserPass()

{

ifstream ifs; //a program that reads data from files and displays it

ifs.open("UserPass.txt");

int i;

while(!ifs.eof())

{

ifs>>userPass[i].ID;

ifs.ignore();

ifs.getline(userPass[i].passWord,20); //allows you to handle input with spaces or multiple words as a single string and used to read a line of text from the input stream and store it in a string variable.

i++;

}

ifs.close(); //used to close an input file stream this means when you open a file for reading using an input file stream you need to close it once you are done

}

void password()

{

Boarder();

countUser(); //used to keep track of count - how many times a piece of code is executed

string password;

char c;

gotoxy(40,20); //can be useful for controlling the output or creating interactive text-based user interfaces

cout << "\n\n\n\n\t |\t\t\tPASSWORD: ";

while(c != '\r') //Loop until 'Enter' is pressed

{

c = getch(); //hold the output screen for some time until the user passes a key from the keyboard to exit the console screen

if(c == 0)

{

switch(getch())

{

default:

break;

};

}

else if(c == '\b') //If the 'Backspace' key is pressed

{

if(password.size() != 0) //If the password string contains data, erase last character

{

cout << "\b \b";

password.erase(password.size() - 1, 1);

}

continue;

}

else if(c <= '9' && c >= '0' || c >= 'a' && c <= 'z' || c >= 'A' && c <= 'Z') //If user enters 1-9, a-z, or A-Z, add it to the password and display an asterisk

{

password += c;

cout << "\*";

}

else

continue;

}

for(int i=0; i<countUser();i++)

{

if(password.compare(userPass[i].passWord)==0)

{

system("cls"); //terminal/command prompt commands used to clear the screen

load\_CHECK();

admin();

}

}

if(password == "a")

{

system("cls");

load\_CHECK(); //checks that a routine (function or class member) is called during the execution of the surrounding test case

admin();

}

else

cout<<"\n\t |\t\t\tWrong Password.. Call Administrator";

cout<<"\n\t |\t\t\tReturning to Main Menu.."<<endl;

cout<<"\n\t |\t\t\t";

Sleep(1000);

cout<<"\n\t |\t\t\tPress Enter to Continue.";

cout<<"\n\t |\t\t\t";

getch();

menu();

}

void gotoxy(int x, int y) //use to print the statement at appropriate position on the screen

{

COORD coord; //structure to hold screen COORDinates X and Y

coord.X = x;

coord.Y = y;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coord); //provides a mechanism for retrieving the standard input ( STDIN ), standard output ( STDOUT ), and standard error ( STDERR ) handles associated with a process

}

void load() //used to make the loading screen slower

{

Boarder();

int timer =5+ rand()%20;

int row,col,r,c,q;

gotoxy(65,34);

cout<<"BOOTING UP...";

gotoxy(60,36);

for(r=1;r<=timer;r++)

{

for(q=0;q<=100000000;q++);//to display the character slowly

printf("%c",177);

}

Sleep(100);

system("cls"); //terminal/command prompt commands used to clear the screen

}

void load\_CHECK() //checks that a routine (function or class member) is called during the execution of the surrounding test case

{

Boarder(); //used to make the log in screen slower

int timer = rand()%5+1;

int row,col,r,c,q;

gotoxy(65,34);

printf("LOG IN...");

gotoxy(60,36);

for(r=1;r<=timer;r++)

{

for(q=0;q<=100000000;q++);//to display the character slowly

printf("%c",177);

}

Sleep(100);

system("cls");

}

void load\_UPDATE()

{

Boarder();

int row,col,r,c,q;

int timer = rand()%25+1;

gotoxy(65,34);

printf("UPDATING DATABASE...");

gotoxy(60,36);

for(r=1;r<=timer;r++)

{

for(q=0;q<=100000000;q++);//to display the character slowly

printf("%c",177);

}

Sleep(100);

system("cls");

}

void load\_EXIT()

{

Boarder();

int row,col,r,c,q;

int timer = rand()%5+1;

gotoxy(65,34);

printf("LOGGING OFF...");

gotoxy(60,36);

for(r=1;r<=timer;r++)

{

for(q=0;q<=100000000;q++);//to display the character slowly

printf("%c",177);

}

Sleep(100);

system("cls");

}

void GotoXY(int x, int y)

{

HANDLE a;

COORD b;

fflush(stdout);

b.X = x;

b.Y = y;

a = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleCursorPosition(a,b);

}

void Boarder()

{

int i;

system("cls");

for(i=10;i<140;i++)

{

GotoXY(i,10);

cout<<"=";

GotoXY(i,58);

cout<<"=";

}

for(i=10;i<58;i++)

{

GotoXY(10,i);

cout<<"|";

GotoXY(140,i);

cout<<"|";

}

}

void art() // the art displayed in the begining and the ending of the output

{

Boarder();

gotoxy(0, 20);

ifstream ifs ("art.txt");

string Lines = "";

if (ifs)

{

while (ifs.good ()) //true if the stream is good, else false

{

string TempLine;

getline (ifs , TempLine); //allows accepting and reading single and multiple line strings from the input stream

TempLine += "\n";

Lines += TempLine;

}

cout << Lines;

date();

}

ifs.close ();

}

void welcome()

{

Boarder();

gotoxy(0, 30);

ifstream ifs ("welcome.txt");

string Lines = "";

if (ifs)

{

while (ifs.good ())

{

string TempLine;

getline (ifs , TempLine);////allows accepting and reading single and multiple line strings from the input stream

TempLine += "\n";

Lines += TempLine;

}

cout << Lines;

}

cout<<"\n\t |\t\t\t\t\t\t\t";

ifs.close ();

Sleep(1000);

}

void exitArt() // the art that will be displayed in the exit of the output

{

Boarder();

gotoxy(0, 30);

ifstream ifs ("exit art.txt");

string Lines = " ";

if (ifs)

{

while (ifs.good ())

{

string TempLine;

getline (ifs , TempLine);////allows accepting and reading single and multiple line strings from the input stream

TempLine += "\n";

Lines += TempLine;

}

cout << Lines;

}

cout<<"\n\t |\t\t\t\t\t\t\t";

ifs.close ();

Sleep(3000);

exit(0);

}

int rate(int hour, int j) // equation of the total money required

{

int hour\_24, hour\_12, hour\_1;

int total, total\_24, total\_12, total\_1;

hour\_24=hour/24;

hour\_12=(hour - hour\_24\*24)/12;

hour\_1=(hour - hour\_24\*24 - hour\_12\*12)/1;

total\_24 = hour\_24 \* rent[j].rate\_per\_day;

total\_12 = hour\_12 \* rent[j].rate\_per\_half;

total\_1 = hour\_1 \* rent[j].rate\_per\_hour;

total = total\_24 + total\_12 + total\_1;

return total;

}

void customerData()

{

ifstream ifs;

ifs.open("Customer.txt");

int custNum = 0;

while (!ifs.eof())

{

ifs >> cust[custNum].id;

ifs.ignore();

ifs.getline(cust[custNum].name,100,';');////allows accepting and reading single and multiple line strings from the input stream

ifs.getline(cust[custNum].phone,15,';');

ifs.getline(cust[custNum].ic,15,';');

ifs.getline(cust[custNum].address,1500);

custNum++;

}

ifs.close();

}

int carCount()

{

int numOfCar = 0;

ifstream ifs("car rental.txt");

string word;

while (getline(ifs, word))////allows accepting and reading single and multiple line strings from the input stream

{

if (!word.empty())

numOfCar++;

}

ifs.close();

return numOfCar;

}

int custCount()

{

int numOfCust = 0;

ifstream ifs("Customer.txt");

string word;

while (getline(ifs, word))////allows accepting and reading single and multiple line strings from the input stream

{

if (!word.empty())

numOfCust++;

}

ifs.close();

return numOfCust;

}

int countUser()

{

int numOfCust = 0;

ifstream ifs("UserPass.txt");

string word;

while (getline(ifs, word))////allows accepting and reading single and multiple line strings from the input stream

{

if (!word.empty())

numOfCust++;

}

ifs.close();

return numOfCust;

}

int countAvail()

{

int numOfCust = 0;

ifstream ifs("available.txt");

string word;

while (getline(ifs, word))////allows accepting and reading single and multiple line strings from the input stream

{

if (!word.empty())

numOfCust++;

}

ifs.close();

return numOfCust;

}

void displayCar()

{

cout << "\t | Plate Number\tBrand\t Model\tCapacity Colour Rate Per Hour Rate Per 12 Hour Rate Per 24 Hour Transmission" << endl;

cout << "\t | =======================================================================================================================" << endl;

for (int i = 0; i < carCount(); i++)

{

cout << "\t | " << rent[i].plate\_num << "\t\t" << rent[i].brand << "\t " << rent[i].model << "\t " << rent[i].capacity << "\t "

<< rent[i].colour << "\t " <<rent[i].rate\_per\_hour << "\t\t " << rent[i].rate\_per\_half << "\t\t " << rent[i].rate\_per\_day

<< "\t\t " << rent[i].transmission << endl;

}

}

void dispAvailCar()

{

cout << "\t | Plate Number\tBrand\t Model\tCapacity Colour Rate Per Hour Rate Per 12 Hour Rate Per 24 Hour Transmission" << endl;

cout << "\t | =======================================================================================================================" << endl;

for (int i = 0; i < countAvail(); i++)

{

cout << "\t | " << avail[i].plate\_num << "\t\t" << avail[i].brand << "\t " << avail[i].model << "\t " << avail[i].capacity << "\t "

<< avail[i].colour << "\t " <<avail[i].rate\_per\_hour << "\t\t " << avail[i].rate\_per\_half << "\t\t " << avail[i].rate\_per\_day

<< "\t\t " << avail[i].transmission << endl;

}

}

void existingCust()

{

int tempCust, hour, userID;

char carSelect[10];

tempCust = custCount() - 1;

ofstream log;

log.open("Log.txt", fstream::app);

ofstream ofs; //It is used to create files and to write data to files

ofs.open("temp.txt");

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

cout << endl;

cout<<"\n\n";

art();

cout << endl;

cout << "\n\t |\t\t\t\tPlease enter user ID : ";

cin >> userID;

for (int i = 0 ; i < custCount() ; i++)

{

if (userID == cust[i].id)

{

cout<<"\n\t |\t\t\t\t\tWELCOME ";

cout << cust[i].name;

log <<"\nNAME: "<< cust[i].name;

cout<<"\t |\tID ";

cout << cust[i].id;

cout << "\n\n";

log<<"\nPHONE: "<<cust[i].phone;

log<<"\nIC: "<<cust[i].ic;

}

}

dispAvailCar();

cout<<"\n\n\t |\t\t\t\t\tPlease select car";

cout<<"\n\t |\t\t\t\t\tPlate Number : ";

cin >> ws;//tells the compiler to ignore buffer and also to discard all the whitespaces before the actual content of string or character array

cin.getline(carSelect,10);////allows accepting and reading single and multiple line strings from the input stream

int x=countAvail();

for(int i = 0; i < countAvail(); i++)

{

if (strcmp(carSelect,avail[i].plate\_num) != 0)

{

ofs << avail[i].plate\_num;

ofs << " ";

ofs << avail[i].brand;

ofs << " ";

ofs << avail[i].model;

ofs << " ";

ofs << avail[i].capacity;

ofs << " ";

ofs << avail[i].colour;

ofs << " ";

ofs << avail[i].rate\_per\_hour;

ofs << " ";

ofs << avail[i].rate\_per\_half;

ofs << " ";

ofs << avail[i].rate\_per\_day;

ofs << " ";

ofs << avail[i].transmission;

if(i != countAvail())

{

ofs<<endl;

}

}

}

ofs.close();

remove("available.txt"); //deletes a specified file

rename("temp.txt","available.txt"); //renames the file represented by the string pointed to by oldname to the string pointed to by newname

cout << "\t |\t\t\t\t\tHours of rent : ";

cin >> hour;

int j;

for(int i = 0; i < carCount(); i++)

{

if (strcmp(carSelect,rent[i].plate\_num) == 0) //used to compare two null-terminated strings and since it the return value=0 it indicates that both strings are equal

{

j = i;

log<<"\nCAR: "<<rent[i].plate\_num;

log<<"\nBRAND: "<<rent[i].brand;

log<<"\nMODEL: "<<rent[i].model;

log <<"\nHOUR: "<< hour;

log <<"\nPAYMENT: "<< rate(hour,j);

}

}

cout << "\t |\t\t\t\tPrice for " << hour << " hours of rent : RM ";

cout << rate(hour,j);

log <<"\nDATE: "<<currentDateTime();

log<<"\n==========================================================";

log.close();

Sleep(5000);

system("cls"); //terminal/command prompt commands used to clear the screen

menu();

}

void allreservations()

{

ifstream myfile;

myfile.open("Log.txt");

string myline;

if(myfile.is\_open())

{

while(myfile)

{

getline(myfile,myline);

cout<<myline<<"\n";

}

}

else

{

cout<<"couldnt open it ";

}

Sleep(6000);

system("cls");

}

void mostrentedcar()

{

string line, word;

int count = 0, maxCount = 0;//This statement declares a variable named `count` of type `int` and initializes it with the value & setting its initial value to be used in further calculations or comparisons in your program.

vector<string> words; //declares a vector object named `words` that can hold elements of type `string

//Opens file in read mode

ifstream file("mostrentedcar.txt");

//Reads each line

while (getline(file, line)) {

transform(line.begin(), line.end(), line.begin(), ::tolower);

string string;

for (auto c : line) {

if (c == ' ' || c == ',' || c == '.') {

words.push\_back(string);

string.clear();

} else {

string += c;

}

}

}

//Determine the most repeated word in a file

for (int i = 0; i < words.size(); i++) {

count = 1;

//Count each word in the file and store it in variable count

for (int j = i + 1; j < words.size(); j++) {

if (words[i] == words[j]) {

count++;

}

}

//If maxCount is less than count then store value of count in maxCount

//and corresponding word to variable word

if (count > maxCount) {

maxCount = count;

word = words[i];

}

}

cout << "Most rented car: " << word << std::endl;

file.close();

Sleep(5000);

system("cls");

}

void rentedcarsperday()

{

int count = 0;

string line;

/\* Creating input filestream \*/

ifstream file("availtemp.txt");

while (getline(file, line))

count++;

cout << "Numbers rented cars per day : " << count << endl;

Sleep(6000);

system("cls");

}

void newCustData ()

{

cout<<"\n\n";

art();

cout << endl;

availCar();

ofstream ofs,log; //It is used to create files and to write data to files

ofs.open("Customer.txt", fstream::app); // used to read the data from the files

log.open("Log.txt", fstream::app);

ofs << endl;

ofstream availTemp;

availTemp.open("availtemp.txt");

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

cout << endl;

cust[custCount()].id = custCount() + 1001;

ofs << cust[custCount()].id;

ofs << ";";

cout << "\n\n";

cout << "\n\t |\t\t\t\t\t Please enter the data below : " << endl;

cout << "\n\t |\t\t\t\t\t Name : ";

cin >> ws;//tells the compiler to ignore buffer and also to discard all the whitespaces before the actual content of string or character array

cin.getline(cust[custCount()].name,100);//allows accepting and reading single and multiple line strings from the input stream

ofs << cust[custCount()].name<<";";

log <<"\nNAME: "<< cust[custCount()].name;

cout << "\t |\t\t\t\t\t Phone Number : ";

cin.getline(cust[custCount()].phone,15);

ofs << cust[custCount()].phone;

ofs << ";";

log <<"\nPHONE: "<< cust[custCount()].phone;

cout << "\t |\t\t\t\t\t IC : ";

cin.getline(cust[custCount()].ic,15);

ofs << cust[custCount()].ic;

ofs << ";";

log <<"\nIC: "<< cust[custCount()].ic;

cout << "\t |\t\t\t\t\t Address : ";

cin.getline(cust[custCount()].address,1500);////allows accepting and reading single and multiple line strings from the input stream

ofs << cust[custCount()].address;

ofs.close();

system("cls"); //terminal/command prompt commands used to clear the screen

int tempCust, hour;

char carSelect[10];

tempCust = custCount() - 1;

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

art();

cout << endl;

cout<<"\n\t |\t\t\tWELCOME ";

cout << cust[tempCust].name;

cout<<"\t |\tID ";

cout << cust[tempCust].id;

cout << "\n\n";

dispAvailCar();

cout<<"\n\n\t |\tPlease select car";

cout<<"\n\t |\tPlate Number : ";

cin >> ws;//tells the compiler to ignore buffer and also to discard all the whitespaces before the actual content of string or character array

cin.getline(carSelect,10);////allows accepting and reading single and multiple line strings from the input stream

int x=countAvail();

for(int i = 0; i < countAvail(); i++)

{

if (strcmp(carSelect,avail[i].plate\_num) != 0) //compares both the strings character by character

{

availTemp << avail[i].plate\_num;

availTemp << " ";

availTemp << avail[i].brand;

availTemp << " ";

availTemp << avail[i].model;

availTemp << " ";

availTemp << avail[i].capacity;

availTemp << " ";

availTemp << avail[i].colour;

availTemp << " ";

availTemp << avail[i].rate\_per\_hour;

availTemp << " ";

availTemp << avail[i].rate\_per\_half;

availTemp << " ";

availTemp << avail[i].rate\_per\_day;

availTemp << " ";

availTemp << avail[i].transmission;

if(i != countAvail())

{

availTemp<<endl;

}

}

}

ofs.close();

remove("available.txt"); //deletes a specified file

rename("availtemp.txt","available.txt"); //renames the file represented by the string pointed to by oldname to the string pointed to by newname

cout << "\t |\tHours of rent : ";

cin >> hour;

int j;

for(int i = 0; i < carCount(); i++)

{

if (strcmp(carSelect,rent[i].plate\_num) == 0)

{

j = i;

rate(hour,j);

log<<"\nCAR: "<<rent[i].plate\_num;

log<<"\nBRAND: "<<rent[i].brand;

log<<"\nMODEL: "<<rent[i].model;

log <<"\nHOUR: "<< hour;

log <<"\nPAYMENT: "<< rate(hour,j);

}

}

cout << "\n\t |\tPrice for " << hour << " hours of rent : RM ";

cout << rate(hour,j);

log <<"\nDATE: "<<currentDateTime();

log<<"\n==========================================================";

log.close();

availTemp.close();

availCar();

Sleep(5000); //used to sleep the particular thread of the C++ that is running in the particular program

system("cls"); //terminal/command prompt commands used to clear the screen

menu();

}

void newCarData()

{

cout<<"\n\n";

art();

cout << endl;

ofstream ofs; //used to create files and to write information to files

ofs.open("car rental.txt" , fstream::app); //allows both reading from and writing to files by default

ofs<<endl;

int newCar = carCount() + 1;

cout << "\n\n";

cout << "\n\t |\t\t\t\t\t Please enter the car data below : " << endl;

cout << "\n\t |\t\t\t\t\t Plate Number : ";

cin >> ws;//tells the compiler to ignore buffer and also to discard all the whitespaces before the actual content of string or character array

cin.getline(rent[newCar].plate\_num,10);

ofs << rent[newCar].plate\_num;

ofs << " ";

cout << "\t |\t\t\t\t\t Brand : ";

cin.getline(rent[newCar].brand,20);

ofs << rent[newCar].brand;

ofs << " ";

cout << "\t |\t\t\t\t\t Model : ";

cin.getline(rent[newCar].model,20);////allows accepting and reading single and multiple line strings from the input stream

ofs << rent[newCar].model;

ofs << " ";

cout << "\t |\t\t\t\t\t Capacity : ";

cin >> rent[newCar].capacity;

ofs << rent[newCar].capacity;

ofs << " ";

cout << "\t |\t\t\t\t\t Colour : ";

cin >> ws;//tells the compiler to ignore buffer and also to discard all the whitespaces before the actual content of string or character array

cin.getline(rent[newCar].colour,20);////allows accepting and reading single and multiple line strings from the input stream

ofs << rent[newCar].colour;

ofs << " ";

cout << "\t |\t\t\t\t\t Rate Per Hour : ";

cin >> rent[newCar].rate\_per\_hour;

ofs << rent[newCar].rate\_per\_hour;

ofs << " ";

cout << "\t |\t\t\t\t\t Rate Per 12 Hour : ";

cin >> rent[newCar].rate\_per\_half;

ofs << rent[newCar].rate\_per\_half;

ofs << " ";

cout << "\t |\t\t\t\t\t Rate Per 24 Hour : ";

cin >> rent[newCar].rate\_per\_day;

ofs << rent[newCar].rate\_per\_day;

ofs << " ";

cout << "\t |\t\t\t\t\t Tranmission (A/M) : ";

cin >> ws;//tells the compiler to ignore buffer and also to discard all the whitespaces before the actual content of string or character array

cin.getline(rent[newCar].transmission,6);////allows accepting and reading single and multiple line strings from the input stream

ofs << rent[newCar].transmission;

ofs.close();

system("cls"); //terminal/command prompt commands used to clear the screen

carData();

admin();

}

void availCar ()

{

ifstream ifs; //reads data from files and displays it

ifs.open("available.txt");

int carNum = 0;

while(!ifs.eof())

{

ifs.getline(avail[carNum].plate\_num,10,' ');

ifs.getline(avail[carNum].brand,20,' ');

ifs.getline(avail[carNum].model,20,' ');

ifs >> avail[carNum].capacity;

ifs.ignore();

ifs.getline(avail[carNum].colour,20,' ');

ifs >> avail[carNum].rate\_per\_hour;

ifs.ignore();

ifs >> avail[carNum].rate\_per\_half;

ifs.ignore();

ifs >> avail[carNum].rate\_per\_day;

ifs.ignore();

ifs.getline(avail[carNum].transmission,6);

carNum++;

ifs>>ws;//tells the compiler to ignore buffer and also to discard all the whitespaces before the actual content of string or character array

}

ifs.close();

}

void resetAvail()

{

Boarder();

ofstream ofs; //used to create files and to write information to files

ofs.open("temp2.txt");

for(int i = 0; i < carCount(); i++)

{

ofs << rent[i].plate\_num;

ofs << " ";

ofs << rent[i].brand;

ofs << " ";

ofs << rent[i].model;

ofs << " ";

ofs << rent[i].capacity;

ofs << " ";

ofs << rent[i].colour;

ofs << " ";

ofs << rent[i].rate\_per\_hour;

ofs << " ";

ofs << rent[i].rate\_per\_half;

ofs << " ";

ofs << rent[i].rate\_per\_day;

ofs << " ";

ofs << rent[i].transmission;

if(i != carCount())

{

ofs<<endl;

}

}

ofs.close();

remove("available.txt");

rename("temp2.txt","available.txt");

admin();

}

void newUserPass()

{

Boarder();

ofstream ofs;

ofs.open("UserPass.txt", fstream::app);

userPass[countUser()].ID = countUser()+1;

string password;

char c;

gotoxy(40,20);

cout << "\n\n\n\n\t |\t\t\tID(DEFAULT): "<<userPass[countUser()].ID;

cout << "\n\t |\t\t\tADD PASSWORD: ";

while(c != '\r') //Loop until 'Enter' is pressed

{

c = getch(); //we can hide the input character provided by the users

if(c == 0)

{

switch(getch())

{

default:

break;

};

}

else if(c == '\b') //If the 'Backspace' key is pressed

{

if(password.size() != 0) //If the password string contains data, erase last character

{

cout << "\b \b";

password.erase(password.size() - 1, 1);

}

continue;

}

else if(c <= '9' && c >= '0' || c >= 'a' && c <= 'z' || c >= 'A' && c <= 'Z') //If user enters 1-9, a-z, or A-Z, add it to the password and display an asterisk

{

password += c;

cout << "\*";

}

else

continue;

}

strcpy(userPass[countUser()].passWord,password.c\_str()); //used to copy one string to another

ofs<<userPass[countUser()].ID;

ofs<<";";

ofs<<userPass[countUser()].passWord;

ofs<<endl;

system("cls");

admin();

}

void showCarData()

{

int x;

cout<<"\n\n";

art();

cout << endl;

cout << "\t | Plate Number\tBrand\t Model\tCapacity Colour Rate Per Hour Rate Per 12 Hour Rate Per 24 Hour Transmission" << endl;

cout << "\t | =======================================================================================================================" << endl;

for (int i = 0; i < carCount(); i++)

{

cout << "\t | " << rent[i].plate\_num << "\t\t" << rent[i].brand << "\t " << rent[i].model << "\t " << rent[i].capacity << "\t "

<< rent[i].colour << "\t " <<rent[i].rate\_per\_hour << "\t\t " << rent[i].rate\_per\_half << "\t\t " << rent[i].rate\_per\_day

<< "\t\t " << rent[i].transmission << endl;

}

cout << "\n\n";

cout<<"\n\t |\t\t\t\t\t 1. BACK";

cout<<"\n\t |\t\t\t\t\t 2. MAIN MENU"<<endl<<"\n\n";

cout<<"\t |\t\t\t\t\tINPUT :";

cin >> x;

system("cls"); //terminal/command prompt commands used to clear the screen

if (x==1)

{

admin();

}

if (x==2)

{

menu();

}

}

void carData ()

{

ifstream ifs; //reads data from files and displays it

ifs.open("car rental.txt");

int carNum = 0;

while(!ifs.eof())

{

ifs.getline(rent[carNum].plate\_num,10,' ');

ifs.getline(rent[carNum].brand,20,' ');

ifs.getline(rent[carNum].model,20,' ');

ifs >> rent[carNum].capacity;

ifs.ignore();

ifs.getline(rent[carNum].colour,20,' ');

ifs >> rent[carNum].rate\_per\_hour;

ifs.ignore();

ifs >> rent[carNum].rate\_per\_half;

ifs.ignore();

ifs >> rent[carNum].rate\_per\_day;

ifs.ignore();

ifs.getline(rent[carNum].transmission,6);

carNum++;

ifs>>ws;//tells the compiler to ignore buffer and also to discard all the whitespaces before the actual content of string or character array

}

ifs.close();

}

void admin()

{

int x;

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

art();

cout << endl;

cout<<"\n\t |\t\t\t\t\t\t\t 1. ADD ADMIN ";

cout<<"\n\t |\t\t\t\t\t\t\t 2. LIST ALL THE CARS ";

cout<<"\n\t |\t\t\t\t\t\t\t 3. LIST ALL THE RESERVATIONS ";

cout<<"\n\t |\t\t\t\t\t\t\t 4. ADD CAR ";

cout<<"\n\t |\t\t\t\t\t\t\t 5. DELETE CAR ";

cout<<"\n\t |\t\t\t\t\t\t\t 6. MOST RENTED CAR ";

cout<<"\n\t |\t\t\t\t\t\t\t 7. total number of rented cars per day ";

cout<<"\n\t |\t\t\t\t\t\t\t 8. RESET AVAILABLE CAR ";

cout<<"\n\t |\t\t\t\t\t\t\t 9. LOG OFF"<<endl<<"\n";

cout<<"\t |\t\t\t\t\t\t\tINPUT :";

cin>>x;

system("cls"); //terminal/command prompt commands used to clear the screen

if (x==9)

{

load\_EXIT();

menu();

}

else if (x==2)

{

showCarData();

}

else if (x==3)

{

allreservations();

}

else if (x==4)

{

newCarData();

}

else if (x==6)

{

mostrentedcar();

}

else if (x==5)

{

delCar();

}

else if (x==7)

{

rentedcarsperday();

}

else if (x==1)

{

newUserPass();

}

else if(x==8)

system("cls");

resetAvail();

}

void user()

{

int x;

art();

cout << endl;

cout<<"\n\t |\t\t\t\t\t\t\t 1. NEW";

cout<<"\n\t |\t\t\t\t\t\t\t 2. EXISTING";

cout<<"\n\t |\t\t\t\t\t\t\t 3. MAIN MENU"<<endl;

cout<<"\n\n\t |\t\t\t\t\t\t\tINPUT :";

cin>>x;

system("cls"); //terminal/command prompt commands used to clear the screen

if (x==1)

{

newCustData();

}

else if (x==2)

{

system("cls"); //terminal/command prompt commands used to clear the screen

load\_UPDATE();

existingCust();

}

else if (x==3)

{

menu();

}

}

void menu()

{

int x;

art();

cout << endl;

cout<<"\n\t |\t\t\t\t\t\t\t 1. ADMIN";

cout<<"\n\t |\t\t\t\t\t\t\t 2. USER";

cout<<"\n\t |\t\t\t\t\t\t\t 3. T%C";

cout<<"\n\t |\t\t\t\t\t\t\t 4. EXIT"<<endl<<"\n";

cout<<"\n\t |\t\t\t\t\t\t\tINPUT :";

cin>>x;

system("cls"); //terminal/command prompt commands used to clear the screen

if (x==2)

{

user();

}

else if (x==1)

{

password();

}

if(x==3)

{

tNc();

getch(); //hold the output screen for some time until the user passes a key from the keyboard to exit the console screen

}

if(x==5)

time();

else if (x==4)

{

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

exitArt();

Sleep(1000); //used to sleep the particular thread of the C++ that is running in the particular program

exit(0);

}

else if (x==6)

{

menu();

}

}

void delCar()

{

char plate[10];

ofstream ofs;

ofs.open("temp.txt");

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

art();

cout << endl;

displayCar();

cout << "\n\n";

cout << "\t | \t\tEnter the car plate number that are going to be deleted (CAPITAL LETTER W/OUT SPACE): "<<endl;

cout<< "\t | \t\t\t\t\t\t\t";

cin >> ws;//tells the compiler to ignore buffer and also to discard all the whitespaces before the actual content of string or character array

cin.getline(plate,10); //extracts characters from the input stream and appends it to the string object until the delimiting character is encountered

int x=carCount();

for(int i = 0; i < carCount(); i++)

{

if (strcmp(plate,rent[i].plate\_num) != 0)

{

ofs << rent[i].plate\_num;

ofs << " ";

ofs << rent[i].brand;

ofs << " ";

ofs << rent[i].model;

ofs << " ";

ofs << rent[i].capacity;

ofs << " ";

ofs << rent[i].colour;

ofs << " ";

ofs << rent[i].rate\_per\_hour;

ofs << " ";

ofs << rent[i].rate\_per\_half;

ofs << " ";

ofs << rent[i].rate\_per\_day;

ofs << " ";

ofs << rent[i].transmission;

if(i != carCount())

{

ofs<<endl;

}

}

}

ofs.close();

remove("car rental.txt");

rename("temp.txt","car rental.txt");

system("cls"); //terminal/command prompt commands used to clear the screen

carData();

admin();

}

void tNc() // this are the rules

{

cout<<endl;

cout<<"\n\t |\t\t\t1. MINIMUM RENT TIME IS 2 HOURS";

cout<<"\n\t |\t\t\t2. RISK BY YOUR OWN";

cout<<"\n\t |\t\t\t3. INCASE ACCIDENT, ALL COSTING IS PAID BY THE DRIVER";

cout<<"\n\t |\t\t\t4. PREPARE ITEM AS BELOW";

cout<<"\n\t |\t\t\t \3COPY OF IC, LICENSE";

cout<<"\n\t |\t\t\t \3DEPOSIT (BASED ON CAR TYPE)";

cout<<"\n\t |\t\t\t5. ADDITIONAL CHARGE IF LATE";

cout<<"\n\t |\t\t\t6. SAFE DRIVE!";

cout<<"\n\n\n\n\t |\t\t\t PRESS ANY KEY TO CONTINUE :)";

cout<<"\n\t |\t\t\t ";

getch();

menu();

}

main()

{

// all the inputs we need

welcome();

load();

Boarder();

readUserPass();

availCar();

customerData();

carData();

menu();

}