Programming Fundamentals

CO102

MTE PROJECT

CAB MANAGEMENT SYSTEM

Submitted By:

Ritesh Dabas (2K20/A7/50)

Riya Jain (2K20/A7/53)

Under the supervision of:

Ms. Gull Kaur

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi- 110042

MARCH, 2021

CERTIFICATE

We hereby certify that the project Dissertation titled “Cab Management System”, which is submitted by Ritesh Dabas(2K20/A7/50) and Riya Jain (2K20/B6/53), comprises original work and has not been submitted in part or full for any Course/Degree to this university or elsewhere as per the candidate’s declaration. Delhi Technological University, Delhi in complete fulfilment of the requirement for the award of the degree of the Bachelor of Technology, is a record of the project work carried out by the students under my supervision. To the best of our knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi

ABSTRACT

Through working on this project we devised a basic functioning cab management system using C language. We understood the functioning and concepts from existing apps and projects to come up with the ideas and execute them as well as possible .

We used the code editor Sublime Text to write, edit and implement our code. The coding language being C, we primarily made use of structs, functions and arrays to create our program since this language is very limited. There were a lot of errors that had to be overcome but we were able to produce a well working system.

We used file handling to extract information from files stored on the system. This helped us maintain the hard coded details for our project. While the program is being run, input is taken at various stages to customize the output according to the user's choices.

This system may be run by a person logging in as a customer or a driver. The customer may book a cab, rate their ride and avail discounts using reference codes. Similarly, the driver will authenticate their identity and choose to get a new ride or end their day.

With dedicated hours and practice we were able to gather enough resources to produce the desired outcome. We took a lot of guidance from our peers and gained a lot of knowledge throughout the process.

ACKNOWLEDGEMENT

In performing our project, we had to take the help and guideline of some respected persons, who deserve our greatest gratitude. The completion of this assignment gives us much pleasure. We would like to show our gratitude to Ms. Gull Kaur, for giving us a good guideline for reporting throughout numerous consultations. We would also like to extend our deepest gratitude to all those who have directly and indirectly guided us in writing this assignment.

Many people, especially Ms. Gull Kaur, our classmates and team members, have made valuable comment suggestions on this proposal which gave us inspiration to improve our assignment. We thank all the people for their help directly and indirectly to complete our assignment.

In addition, we would like to thank the Department of Computer Science, Delhi Technological University for giving us the opportunity to work on this topic.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

Why is the topic important ?

Through this project we aimed to devise a handy and easy to use cab management system using the C programming language. We incorporated the basic features of a cab service into the program to make it as user friendly as possible. This cab management system enabled us to understand the basic principles of various cab service platforms that we use in our daily life.

Applications

In our daily lives, we all have come across a situation where we might need to book a cab so we resort to the apps on our phones like UBER, OLA, LYFT etc. So taking inspiration from that, we tried to develop a cab management system. We mapped out how the program works and how to go about the procedure to devise a system which works in the same way. We gathered some ideas and implemented them in the best way possible with the resources given.

What did we do in the project?

In this cab management system, the user may login as a customer or a driver and then proceed according to the flow of the system. The customer will input their coordinates and their choice of cab which then yields the probable output. They are allotted the nearest cab depending on their coordinates. Once the ride ends, they are displayed the amount which they can avail a discount upon if their reference code is valid. They also rate the driver at the end. The driver on logging in can see their rating and the amount they earned and also get coordinates for a new customer to drive to their destination. At the end of the ride the driver is rated, which affects their average rating, and the earning is added to the total amount. This system depicts the basic functioning of any cab management system.

PROJECT

The project aims to build a program in C language for a Cab Management System from scratch.

We started off with creating a basic flowchart and decided to build upon it as we went ahead with the project. To create an algorithm from the flowchart, we thought what major components of C do we require? We identified them as header files, output formatting functions, structures, user defined functions and file handling.

All the above mentioned topics were a part of our curriculum, and some parts which weren't were studied by us from various resources. What we actually did in the project is described in the forthcoming pages in detail.

DESIGN

We used the code editor Sublime Text to write, edit and implement our code. We already had references for various cab management systems in C++ and Python, taken from our seniors and from the internet.

But replicating them in C was not easy as C doesn't support member functions in its structures, vectors and other various aspects. Hence, we found the solutions to these problems, using functions defined outside structures, making use of arrays and not vectors and so on.

We also included various original ideas such as creating a single system for both the customer and the driver, using referral codes for discounts, formatting the output to make it look more organised etc.

However, we weren't able to create an attractive GUI using C. Also, interlinking of maps is not there and neither is interlinking of information of drivers and customers.

FILE HANDLING IN C

File handling in C refers to the task of storing data in the form of input or output produced by running C programs in data files, namely, a text file or a binary file for future reference and analysis.

We also made use of file handling in our project to approach the text files on the system so that we can maintain some details in the form of text files that are accessed at runtime.

Some operations we used in data handling:

1. Opening an existing file (fopen with attributes as “r” or “r+” or “w” or “w++”)
2. Reading from file (fscanf or fgets)
3. Writing to a file (fprintf or fputs)
4. Moving to a specific location in a file (fseek, rewind)
5. Closing a file (fclose)

We use the fscanf function in scan_data function and the fprintf function in the print_data function

r refers to the read only mode (Searches file. If the file is opened successfully, fopen( ) loads it into memory and sets up a pointer which points to the first character in it. If the file cannot be opened, fopen( ) returns NULL.)

w refers to write mode (Searches file. If the file exists, its contents are overwritten. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open the file.)

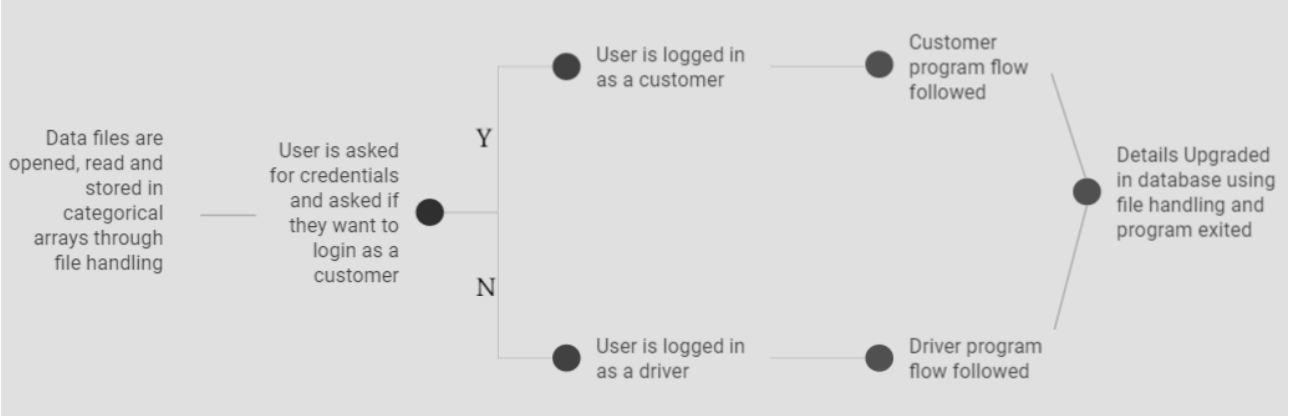
The syntax for the file handling function:

```
FILE * filePointer;
filePointer = fopen(“fileName.txt”, “r”);
fscanf(filePointer, "%s %s %s %d", str1, str2, str3, &year);
fclose(filePointer);
```

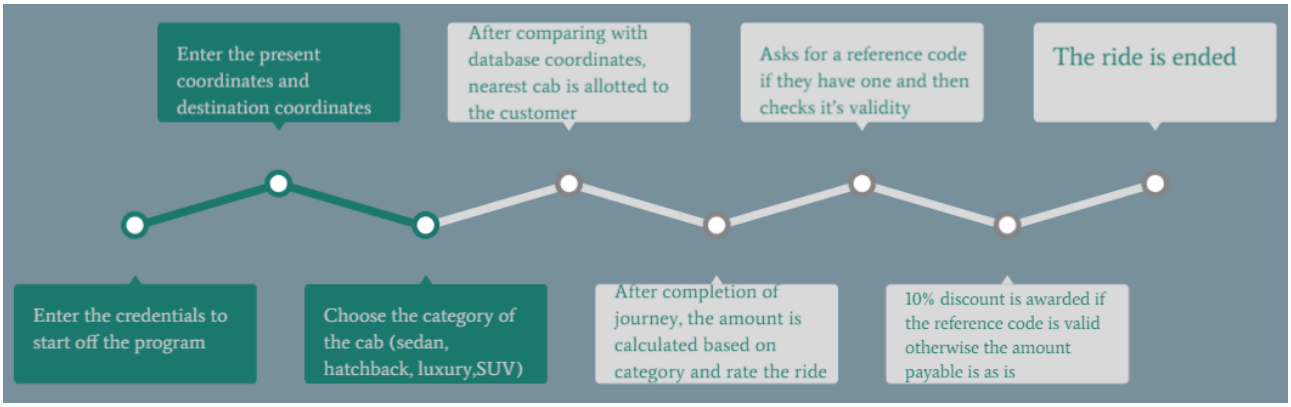
```
FILE *filePointer ;
filePointer = fopen(“fileName.txt”, “w”);
fprintf(filePointer, "%s %s %s %d", "We", "are", "in", 2012);
fclose(filePointer);
```

FLOW OF PROGRAM

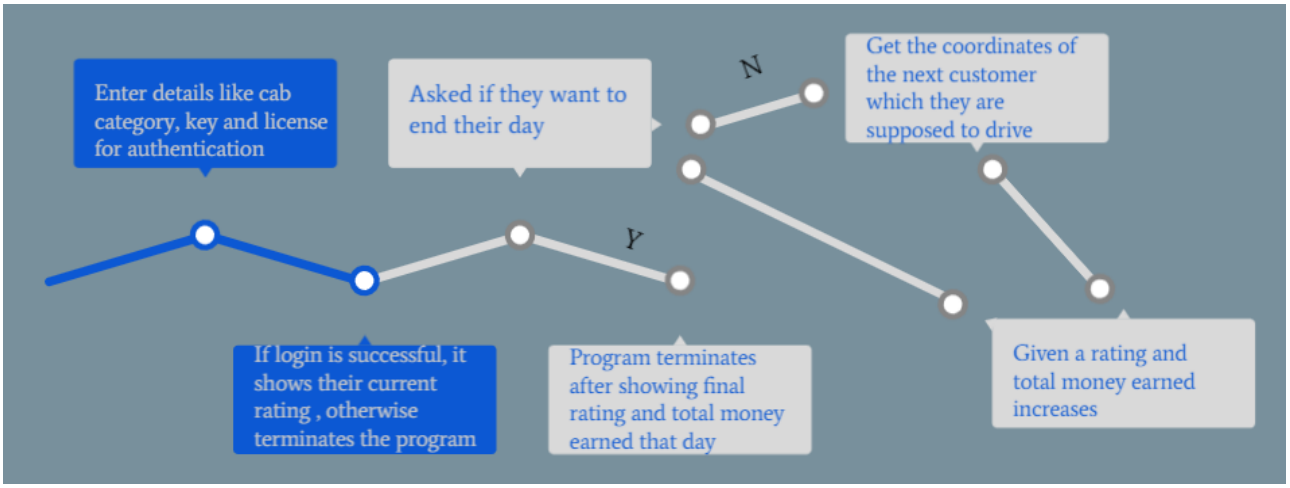
In this project, the user may login as a customer or a driver. The following is how the main program works



When the user logins as a customer:



When the user logins as a driver:



CODE FILE

HEADER FILES INCLUDED

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <ctype.h>
```

OUTPUT FORMATTING FUNCTIONS

```
COORD coord= {0,0}; // this is global variable
void gotoxy(int x,int y) //to set the cursor at given row and column value
{
    coord.X=x;
    coord.Y=y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),coord);
}
```

The **gotoxy() function** is used to place the control cursor to a required position on the console screen. The function gotoxy() can have two arguments i.e., one for the X coordinate and other for the Y coordinate. It formats the output placement on the output screen so that it is easy on the eyes and doesn't look cluttered one after the another.

```
void clrscr() //to clear the output screen
{
    system("cls");
}
```

The **clrscr() function** is used to clear the output screen. The system("cls") function is used to run system/command prompt commands and here cls clears the output screen.

STRUCTS CREATED

```
struct Customer  //class for customers
{
    int x=0;        //x,y are position coordinates; d_x,d_y are destination coordinates
    int y=0;
    int d_x=0;
    int d_y=0;
};
```

The **Customer struct** is used to create customer objects and store the present coordinates (x,y) and destination coordinates (d_x,d_y) of the customer

```
struct Cab //class for cab
{
    int base_cost, cost_per_km, base_distance;  // base_cost -- is the cost before base_distance
    int x, y, total_cost, license, total_rides;    // x,y are coordinates of the car, total_rides is total number of rides of the car so far
    double rating;     // rating is the rating (out of 5) of the car so far
    char name[10], type[10];   //name is name of car and type is the category
};
```

The **Cab struct** is used to create the cab objects and store all their properties and details in the variables declared. The rating variable is edited based on the input of the user but all the other variables are hard coded.

- base_cost, cost_per_km and base_distance provide the values to calculate the fare of the ride.
- x,y are the coordinates of the cab.
- license is a unique property of every car.
- total_rides is incremented every time a ride in that specific cab is completed.
- rating is the present rating of the car out of 5.
- name is the model of the cab while type is the category.

OTHER FUNCTIONS

```
void Customer(struct Customer &c, int x, int y, int d_x, int d_y) //Constructor for Customer
{
    c.x = x;
    c.d_x = d_x;
    c.y = y;
    c.d_y = d_y;
}
```

The **Customer() function** is a void returning function that takes a customer object and four integers as its parameters. It is used to initialize the customer object. The first two integers (x,y) are assigned as the customer's present coordinates and the next two integers are assigned as the customer's destination coordinates.

```
void prior(struct Customer &c) //Updates the Customer's coordinates to their destination coordinates after they finish their ride
{
   c.x = c.d_x;
   c.y = c.d_y;
}
```

The **prior() function** is a void returning function that takes a customer object as the parameter. It is used after the ride is completed to update the present coordinates of the customer as their destination coordinates.

```
int get_distance(int x1, int y1, int x2, int y2) //gives |x2-x1|+|y2-y1|
{
  if (x1 >= x2 && y1 >= y2)
    return x1 - x2 + y1 - y2;
  else if (x1 >= x2 && y2 >= y1)
    return x1 - x2 + y2 - y1;
  else if (x2 >= x1 && y1 >= y2)
    return x2 - x1 + y1 - y2;
  else
    return x2 - x1 + y2 - y1;
}
```

The **get_distance() function** is an integer returning function that takes four integers as its parameters. It returns the modulus of the distance between the two sets of coordinates passed as parameters.

```
void Cab(struct Cab &c, char n[], int x, int y, int l, float r, int t)
//Constructor that assigns all values, including type of the car based on name
{
    c.x = x;
    c.y = y;
    c.license = l;
    c.rating = r;
    c.total_rides = t;
    strcpy(c.name,n);

    if (strcmp(n,"Dzire")==0 || strcmp(n,"Amaze")==0 || strcmp(n,"Vento")==0 ||
        strcmp(n,"City")==0 || strcmp(n,"Verna")==0 || n == "Rapid" || strcmp(n,"Ciaz")==0)
    {
      strcpy(c.type,"Sedan");
      c.base_distance = 5;
      c.base_cost = 80;
      c.cost_per_km = 15;
    }
    else if (strcmp(n,"i20")==0 || strcmp(n,"i10")==0 || strcmp(n,"Swift")==0 ||
        strcmp(n,"Baleno")==0 || strcmp(n,"Kwid")==0 || strcmp(n,"WagonP")==0 || strcmp(n,"Polo")==0)
    {
      strcpy(c.type , "Hatchback");
      c.base_distance = 10;
      c.base_cost = 100;
      c.cost_per_km = 13;
    }
    else if (strcmp(n,"Innova")==0 || strcmp(n,"Brezza")==0 || strcmp(n,"Seltos")==0 ||
        strcmp(n,"Sonet")==0 || strcmp(n,"Creta")==0 || strcmp(n,"Ertiga")==0 || strcmp(n,"Fortuner")==0)
    {
      strcpy(c.type , "SUV");
      c.base_distance = 8;
      c.base_cost = 125;
      c.cost_per_km = 18;
    }
    else if (strcmp(n,"BMW")==0 || strcmp(n,"Audi")==0 || strcmp(n,"Mercedes")==0 ||
        strcmp(n,"LandRover")==0 || strcmp(n,"Jaguar")==0 || strcmp(n,"Porsche")==0 || strcmp(n,"Lexus")==0)
    {
      strcpy(c.type , "Luxury");
      c.base_distance = 15;
      c.base_cost = 175;
      c.cost_per_km = 20;
    }
}
```

The **Cab() function** is a void returning function that takes various parameters including a cab object. This function is used to initialize the cab object and assign the various details.

- It initializes the car coordinates, license, rating, total_rides and name of the cab from the parameters passed.
- The name is then compared with the model using conditional statements and further initializes the type, base_cost, base_distance and cost_per_km based on the category match
- The strcpy function copies the n parameter to name of cab and strcmp function returns true/false depending if the two strings match

```
int get_distance_from_person(struct Customer &c, struct Cab &b)  // gives the distance of the cab from the customer
{
    return get_distance(c.x, c.y, b.x, b.y);
}
```

The **get_distance_from_person() function** is an integer returning function that takes a customer object and a cab object as parameters. It is used to calculate and return the distance between the cab and the customer using the get_distance function.

```
int get_cost(struct Customer &c, struct Cab &b) //gives cost of the taxi ride of this cab for customer c
{
    int d = get_distance_from_person(c,b) +   //Total ditance travelled by car = distance from person + distance from person to destination
            get_distance(c.x, c.y, c.d_x, c.d_y);
    if (d <= b.base_distance)
      return b.base_cost;
    else
      return (((d - b.base_distance) * b.cost_per_km) + b.base_cost);
}
```

The **get_cost() function** is an integer returning function that takes a customer object and a cab object as parameters. It calculates and returns the total cost of the ride by checking if the distance covered is less or greater than the base distance

- It calculates the total distance as the sum of distance between cab and the person and distance between the person and destination.
- If total distance is less than or equal to base_distance, it returns base_cost as total cost
- If total distance is more than the base_distance, it returns total cost as base_cost + (d-base_distance)*cost_per_km (values depending on category of the cab)

```
void print_values(struct Customer &c, struct Cab &b)  //to print all the details of the cab
{
    printf("Cab: %s \tLicense: %d \tLocated at (%d,%d) \tDistance from you: %d",b.name,b.license,b.x,b.y,get_distance_from_person(c,b));
}
```

The **print_value() function** is a void returning function that takes a customer object and a cab object as parameters. It uses a simple printf function to display the details of the cab object passed as parameter.

```
void rate(struct Cab &c) //Allows user to rate their ride and updates the rating of the car as well as total rides
{
    printf("Rate your ride from 1-5 : ");
    double r;
    scanf("%lf",&r);
    c.total_rides++;
    c.rating = c.rating + (r/c.total_rides);
}
```

The **rate() function** is a void returning function that takes a cab object as a parameter. It is used to assign a rating to the cab after the ride is completed. It takes the input from the user and assigns the rating as the average of new plus old rating. It also increments the total_rides of the cab by 1.

```
double driver_rate(struct Cab &c) //This is for when user has logged in as a taxi driver
{
    srand(time(0));
    double r = 1 + (rand() % (5));
    c.total_rides++;
    c.rating = c.rating + (r/c.total_rides);
    return r;
}
```

The **driver_rate() function** is a double datatype returning function that takes a cab object as a parameter. It is used when the user is logged in as a driver and it generates a random variable r using the rand() function. r is used to alter the rating of that car.

```
int journey(struct Cab arr[], struct Customer &d) //provides the closest car from v to Person p and returns the cost
{
    struct Cab c = arr[0];
    int min = get_distance_from_person(d,c);
    int f = 0;
    clrscr();
    gotoxy(5,2);
    printf("Cars available in your chosen category are: ");

    for (int i=0;i<10;i++)
    {
        c = arr[i];
        if (get_distance_from_person(d,c) < min)  //if arr[i] cab is closer, update least distance
        {
          min = get_distance_from_person(d,c);
          f = i;
        }
        gotoxy(5,i+4);
        print_values(d,c);  //prints all available cars
    }

    gotoxy(5,16);
    printf("================================================================================");
    gotoxy(10,17);
    printf("The closest car to you has been chosen, which is: ");
    gotoxy(10,18);
    print_values(d,arr[f]);
    gotoxy(10,20);
    printf("Total cost of journey is Rs.%d" ,get_cost(d,arr[f]));
    gotoxy(10,21);
    printf("Driver will reach you in %d minutes. Enjoy your ride!\n", get_distance_from_person(d,arr[f])*2);
    gotoxy(5,22);
    printf("================================================================================");

    int cost1 = get_cost(d,arr[f]);     //total cost of ride
    prior(d);   //update coordinates of customer

    Sleep(8000);    //pause the output screen for 8 seconds
    clrscr();
    gotoxy(5,2);
    printf("Your ride is finished! You are now at %d,%d ", d.x, d.y);
    gotoxy(5,3);
    rate(arr[f]);   //rating the cab ride

    return cost1;
}
```

The **journey() function** is an integer returning function that takes an array of cab objects and a customer object as parameters.

- It clears the output screen and displays the details of all the cabs available in the category array.
- It declares an integer min as the distance between the customer and the first cab object in the array. Then using a for loop it assigns min as the minimum distance between the customer and the cab objects in the array.
- Based on the minimum distance, it chooses the cab that is nearest to the customer and displays the total cost of the journey using get_cost() function
- The output screen is then paused for 8 seconds
- The output screen is cleared and the ride is ended. The customer is asked for a rating using rate() function

```
bool isValid(int n, char ref[])     //checks the validity of reference code
{
    int d=0,u=0,l=0,s=0;

    for(int i=0;i<n;i++){
        if(ref[i]<=122 && ref[i]>=97)
            l=1;
        else if(ref[i]>=65 && ref[i]<=90)
            u=1;
        else if(ref[i]>=48 && ref[i]<=57)
            d=1;
        else
            s=1;
    }

    if((l+u+d+s)==4)
        return true;
    else
        return false;
}
```

The **isValid() function** is a boolean value returning function that takes a character in a string and an integer as parameters. It is used to check the validity of the reference code entered by the user and make sure that the reference code contains one uppercase alphabet, one lowercase alphabet, one digit and a special character.

FILE HANDLING FUNCTIONS

```
void scan_data(struct Cab Arr[], FILE &f) //reads data from input file and stores it in array arr
{
    for (int i = 0; i < 10; i++)
    {
        struct Cab c;
        char name[10];
        int x, y, l, t;
        double r=0.0;
        fscanf(&f,"%s %d %d %d %lf %d",&name,&x,&y,&l,&r,&t);
        Cab(c,name, x, y, l, r, t);
        Arr[i] = c;
    }

}
```

The **scan_data() function** is a void returning function that takes a file and an array of cab objects as parameters. This function is used to scan the file passed as the parameter. A cab object is created along with all the variables for its details.

The file is scanned line by line and the values are stored into the variables. The cab object is then initialised by assigning these variables by passing them in the Cab() function. The object is then stored into the array of cab objects.

```
void print_data(struct Cab Arr[], FILE &f) //reads data from array arr and writes it into the file
{
    for (int i = 0; i < 10; i++)
    {
        struct Cab c = Arr[i];
        fprintf(&f,"%s %d %d %d %lf %d\n",c.name,c.x,c.y,c.license,c.rating,c.total_rides);
    }
}
```

The **print_data() function** is a void returning function that takes a file and an array of cab objects as parameters. This function is used to iterate through the array of cab objects and print the details into the file line by line.

MAIN FUNCTION

```
int main()
{
    //declaring arrays of struct cab
    struct Cab arr_sd[10];
    struct Cab arr_hb[10];
    struct Cab arr_su[10];
    struct Cab arr_lx[10];

    FILE *in_sedans, *in_hatchbacks, *in_SUVs, *in_Luxury;
    //open all the 4 text files
    in_sedans = fopen("sedans.txt","r");
    in_hatchbacks = fopen("hatchbacks.txt","r");
    in_SUVs = fopen("SUVs.txt","r");
    in_Luxury = fopen("Luxury.txt","r");
    //read data from all text files into their respective arrays
    scan_data(arr_sd, *in_sedans);
    scan_data(arr_hb, *in_hatchbacks);
    scan_data(arr_su, *in_SUVs);
    scan_data(arr_lx, *in_Luxury);
    //close the input files
    fclose(in_sedans);
    fclose(in_hatchbacks);
    fclose(in_SUVs);
    fclose(in_Luxury);



    gotoxy(30,2);
    printf("==================== CAB MANAGEMENT SYSTEM ====================");
    gotoxy(28,3);
    printf("( PRESENTED BY Ritesh Dabas(2K20/A7/50) and Riya Jain(2K20/A7/53) )");

    gotoxy(10,5);
    printf("Do you want to login as a customer [Y/N]? ");
    char input;
    scanf("%c", &input);
```

In the beginning of the main function, four arrays of cab objects are declared for different categories. Four file pointers are declared and then using data handling and its functions the files are opened in read only mode, the data is stored into the arrays and the files are then closed.

Using the gotoxy() function we display the heading in the needed place and then take the input from the user to know if they want to login as a customer or a driver.

```
if (input == 'Y' || input == 'y')
{ //LOGGED IN AS CUSTOMER
    int go = 0, ride_cost = 0;
    struct Customer c;      //initialise person at 0,0
    char f_name[20], l_name[20];
    int c_x, c_y;

    while (go==0)
    {
        gotoxy(10,6);
        printf("Welcome! Enter your full name. (Type exit program to exit) : ");
        scanf("%s %s",&f_name,&l_name);

        if (f_name == "exit" && l_name == "program")
            return 0;

        gotoxy(10,8);
        printf("Welcome %s %s! Enter your present coordinates separated by a space - ");
        scanf("%d %d",&c_x, &c_y);
        //update person coordinates from 0,0 to coordinates user entered
        c.x = c_x;
        c.y = c_y;

        gotoxy(10,9);
        printf("Enter your destination coordinates, separated by a space - ");
        int d_x, d_y;
        scanf("%d %d",&d_x, &d_y);
        //Update Customer destination coordinates to user entered destination coordinates
        c.d_x = d_x;
        c.d_y = d_y;
```

If the user is logged in as a customer we declare a customer object and ask them to enter the credentials like their name, their present coordinates and their the destination coordinates. These values are assigned accordingly to the customer object.

```
        bool check = true; //flag is there as a check to see if the user input is valid or not
        while (check)
        {
            char category[10];
            gotoxy(5,11);
            printf("Choose the category of the cab you want to ride in :");
            gotoxy(5,12);
            printf("Pricing model");
            gotoxy(5,13);
            printf("1. Sedan      base distance = 5 KM    base cost = Rs. 80      Cost per km = Rs.15 ");
            gotoxy(5,14);
            printf("2. SUV        base distance = 10 KM   base cost = Rs. 100     Cost per km = Rs.13 ");
            gotoxy(5,15);
            printf("3. Hatchback  base distance = 8 KM    base cost = Rs. 125     Cost per km = Rs.18 ");
            gotoxy(5,16);
            printf("4. Luxury     base distance = 15 KM   base cost = Rs. 175     Cost per km = Rs.20 ");
            gotoxy(5,18);
            scanf("%s", &category);
            //call journey function according to chosen category and store the cost of ride
            if (strcmp(category,"sedan")==0)
            {
              ride_cost = journey(arr_sd, c);
              check = false;
            }
            else if (strcmp(category,"hatchback")==0)
            {
              ride_cost = journey(arr_hb, c);
              check = false;
            }
            else if (strcmp(category,"SUV")==0 || strcmp(category,"suv")==0)
            {
              ride_cost = journey(arr_su, c);
              check = false;
            }
            else if (strcmp(category,"luxury")==0)
            {
              ride_cost = journey(arr_lx, c);
              check = false;
            }
            else
                printf("Oops! Try again! ");
        }

        go = 1;
    }
```

Then using a while loop which goes on till the customer has completed a ride, the pricing module is displayed so that they can decide on a category of the cab to ride in.

Then depending on the category the journey() function is called by passing the category array and the customer object as parameters.
This completes their ride and exits the while loop.

```
    char inp;
    gotoxy(5,5);
    printf("Do you have a reference code to avail a 10 percent discount [Y/N]: ");
    scanf(" %c", &inp);

    double discount = 0.1*ride_cost;    // discount is 10% of total cost

    if(inp == 'Y' || inp == 'y'){
        gotoxy(5,6);
        printf("Enter the 6 character reference code : ");
        char ref[7];
        scanf("%s",&ref);

        if(isValid(6,ref)==true){   //if valid, 10% discount is applied, else amount is as is
            gotoxy(5,8);
            printf("Congratulations! Your reference code is valid and you avail a discount of %lf.
                Your effective fair is %lf ", discount, (double)ride_cost-discount);
        }
        else{
            gotoxy(5,8);
            printf("Your reference code is invalid. Your effective fair is %d", ride_cost);
        }
    }
    else{
        gotoxy(5,6);
        printf("Effective Fair is %d ",ride_cost);
    }

    gotoxy(20,10);
    printf("=================== Thanks for riding! Hope you had a great journey! ===================");
} //Customer Login Ends
```

After the ride is completed and the cost is displayed, the customer is asked for a reference code to avail a 10% discount.

If the reference code is not available the cost of the ride is as is but in case it is available, the customer is asked to enter the reference code and its validity is checked.

If the reference code is valid or 10% discount is given otherwise the cost is as is.

```
else
{ //LOGGED IN AS TAXI DRIVER
    struct Cab car_owned;
    int amount=0; // total money earned

    gotoxy(5,7);
    printf("Enter your category (sedans/hatchbacks/SUVs/Luxury) :");
    char categ[10];
    scanf("%s", &categ);

    gotoxy(5,8);
    printf("Enter your login key: "); //each driver has a unique login key, which is equal to their car's index in their respective array
    int key;
    scanf("%d", &key);

    gotoxy(5,9);
    printf("Enter your License No: ");
    int lic_check;
    scanf("%d", &lic_check);
    //for each driver, login key serves as username whereas their license number serves as password
    //check to see if username and password match:
    if (strcmp(categ,"sedans")==0)
    {
        if (lic_check == arr_sd[key].license)
        {
            car_owned = arr_sd[key]; //Assign car_owned as the car that user logged in as
            gotoxy(15,11);
            printf("================ Sucessful Login as a driver ================ ");
            gotoxy(5,12);
            printf("Welcome %s!", arr_sd[key].name);
            gotoxy(5,13);
            printf("Your rating is %lf", arr_sd[key].rating);
        }
        else
        {
            gotoxy(15,11);
            printf("================ ERROR. WRONG DETAILS ================");
            return 1;
        }
    }
    //similar code for other categories:
```

If the user logs in as a driver, a Cab object is declared and the total amount earned is set as zero. The user is asked for login credentials which include the category of the car owned, the login key and the licence number of the car owned.

For authentication, the licence entered is compared to the license of the cab object at the key index of the array.

If the login is successful, the driver is welcomed and displayed their rating.
In case the licence number entered doesn't match the actual licence number, the login is unsuccessful and the program is ended.

```
    while (true)        // infinite loop that runs till user says they want to end their day asked after each ride
    {
        Sleep(5000);
        clrscr();
        gotoxy(5,2);
        char a;
        printf("Would you like to end your day? [Y/N] ");
        scanf(" %c", &a);

        if (a == 'Y' || a == 'y')
        {
            gotoxy(5,4);
            printf("You earned Rs.%d today. Goodbye!", amount);
            break;
        }

        else
        {
            int x, y, dx, dy;
            //Assign passenger with random coordinates to Driver
            srand(time(0));
            x = rand() % 50;
            dx = rand() % 50;
            y = rand() % 50;
            dy = rand() % 50;
            //Create a Customer object
            struct Customer c;
            Customer(c,x, y, dx, dy);
            gotoxy(5,4);
            printf("Your passenger is at %d,%d and their destination is %d,%d.", x, y, dx , dy);
            gotoxy(5,5);
            printf("Go to your passenger and drive them to their destination");
            Sleep(5000);
            gotoxy(15,7);
            printf("Your ride is over. You earned Rs.%d !", get_cost(c, car_owned));

            double rat;
            amount += get_cost(c, car_owned); //Add money earned to total_money
            if (strcmp(categ,"sedans")==0)
              rat = driver_rate(arr_sd[key]);   //updates car rating and assigns rat as whatever value they were rated
            else if (strcmp(categ,"hatchbacks")==0)
              rat = driver_rate(arr_hb[key]);
            else if (strcmp(categ,"SUVs")==0)
              rat = driver_rate(arr_su[key]);
            else if(strcmp(categ,"luxury")==0)
              rat = driver_rate(arr_lx[key]);

            gotoxy(15,8);
            printf("You were rated %lf", rat);      //shows the driver what they were rated
        }
    }
}
```

The driver is asked if they want to end their days until they answer yes, otherwise the procedure for the next ride is followed.

If they don't end their day, random coordinates are generated and displayed as the coordinates of the customer and the destination. The program sleeps for 5 seconds and the ride is over. The driver is allotted and displayed a random rating.

If they choose to end their day, they are displayed their earnings for the day and the program is exited.

```
    FILE *out_hatchbacks, *out_sedans, *out_Luxury, *out_SUVs;
    //open the text files
    out_hatchbacks = fopen("hatchbacks.txt","w");
    out_SUVs = fopen("SUVs.txt","w");
    out_Luxury = fopen("Luxury.txt","w");
    out_sedans = fopen("sedans.txt","w");
    //output the updated data of all the cars into their respective txt fies:
    print_data(arr_sd, *out_sedans);
    print_data(arr_hb, *out_hatchbacks);
    print_data(arr_su, *out_SUVs);
    print_data(arr_lx, *out_Luxury);
    //close all the output streams:
    fclose(out_sedans);
    fclose(out_hatchbacks);
    fclose(out_SUVs);
    fclose(out_Luxury);

    return 0;

}
```

In the end of the main function, four file pointers are declared and using the data handling functions the files are open in read and write only mode, the modified data is printed into them in the same format as before and then the files are closed.

RESULTS

For customer login

```
                    ==================== CAB MANAGEMENT SYSTEM ====================
                  ( PRESENTED BY Ritesh Dabas(2K20/A7/50) and Riya Jain(2K20/A7/53) )

     Do you want to login as a customer [Y/N]? y
     Welcome! Enter your full name. (Type exit program to exit) : riya jain

     Welcome riya jain! Enter your present coordinates separated by a space - 4 8
     Enter your destination coordinates, separated by a space - 9 14

Choose the category of the cab you want to ride in :
Pricing model
1. Sedan      base distance = 5 KM    base cost = Rs. 80      Cost per km = Rs.15
2. SUV        base distance = 10 KM   base cost = Rs. 100     Cost per km = Rs.13
3. Hatchback  base distance = 8 KM    base cost = Rs. 125     Cost per km = Rs.18
4. Luxury     base distance = 15 KM   base cost = Rs. 175     Cost per km = Rs.20

luxury
```

Logged in as a customer and input credentials, coordinates & category

```
Cars available in your chosen category are:

Cab: BMW   License: 5224   Located at (10,14)      Distance from you: 12
Cab: BMW   License: 5158   Located at (42,74)      Distance from you: 104
Cab: Jaguar        License: 1915   Located at (19,27)      Distance from you: 34
Cab: Audi  License: 7125   Located at (22,12)      Distance from you: 22
Cab: Porsche       License: 5511   Located at (13,39)      Distance from you: 40
Cab: LandRover     License: 8721   Located at (22,55)      Distance from you: 65
Cab: Lexus         License: 8246   Located at (49,77)      Distance from you: 114
Cab: Lexus         License: 9511   Located at (3,29)       Distance from you: 22
Cab: Audi  License: 8990   Located at (28,59)      Distance from you: 75
Cab: Audi  License: 7002   Located at (12,12)      Distance from you: 12


================================================================================
     The closest car to you has been chosen, which is:
     Cab: BMW      License: 5224   Located at (10,14)      Distance from you: 12

     Total cost of journey is Rs.335
     Driver will reach you in 12 minutes. Enjoy your ride!
================================================================================
```

Shows available cabs and chooses the nearest one

```
    Your ride is finished! You are now at 9,14
    Rate your ride from 1-5 : 3.75

    Do you have a reference code to avail a 10 percent discount [Y/N]: y
    Enter the 6 character reference code : 6yt7H#

    Congratulations! Your reference code is valid and you avail a discount of 33.500000. Your effective fair is 301.500000
                  ==================== Thanks for riding! Hope you had a great journey! ====================

Press any key to continue . . .
```

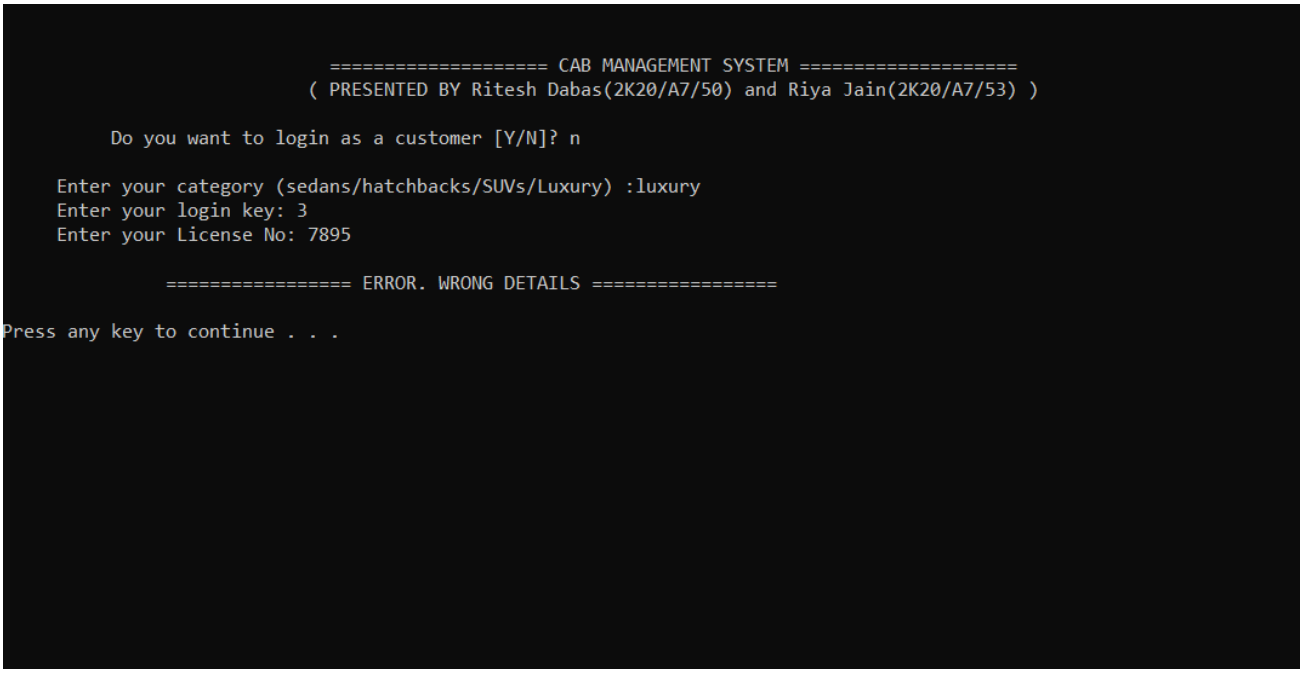
```
    Your ride is finished! You are now at 12,54
    Rate your ride from 1-5 : 3.75

    Do you have a reference code to avail a 10 percent discount [Y/N]: y
    Enter the 6 character reference code : 87gh$h

    Your reference code is invalid. Your effective fair is 965

                  ==================== Thanks for riding! Hope you had a great journey! ====================

Press any key to continue . . .
```

Rate the ride and check validity of reference code

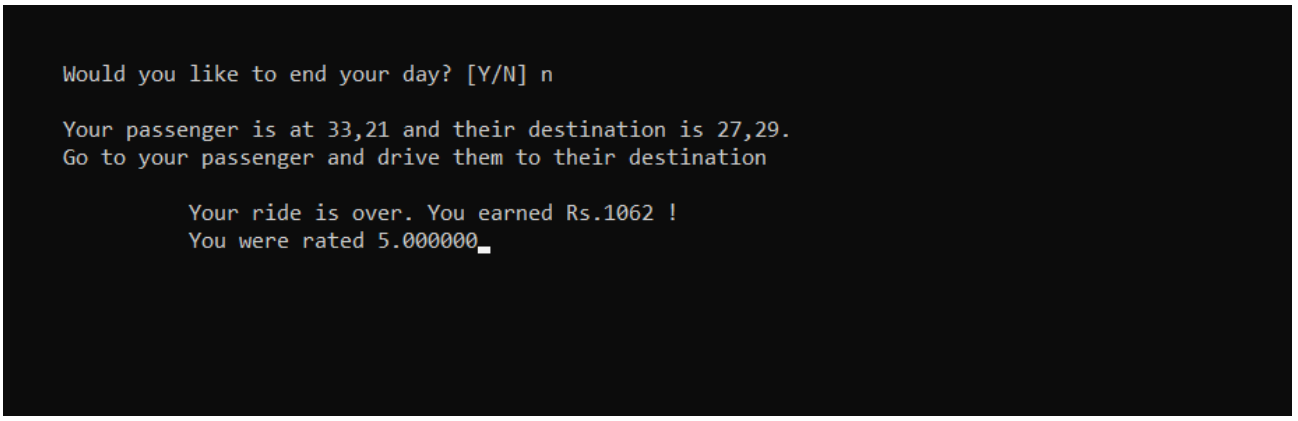
For driver login

```
                          ==================== CAB MANAGEMENT SYSTEM ====================
                        ( PRESENTED BY Ritesh Dabas(2K20/A7/50) and Riya Jain(2K20/A7/53) )

        Do you want to login as a customer [Y/N]? n

    Enter your category (sedans/hatchbacks/SUVs/Luxury) :hatchbacks
    Enter your login key: 2
    Enter your License No: 1115

              ================ Sucessful Login as a driver ================
    Welcome i10!
    Your rating is 2.702344
```

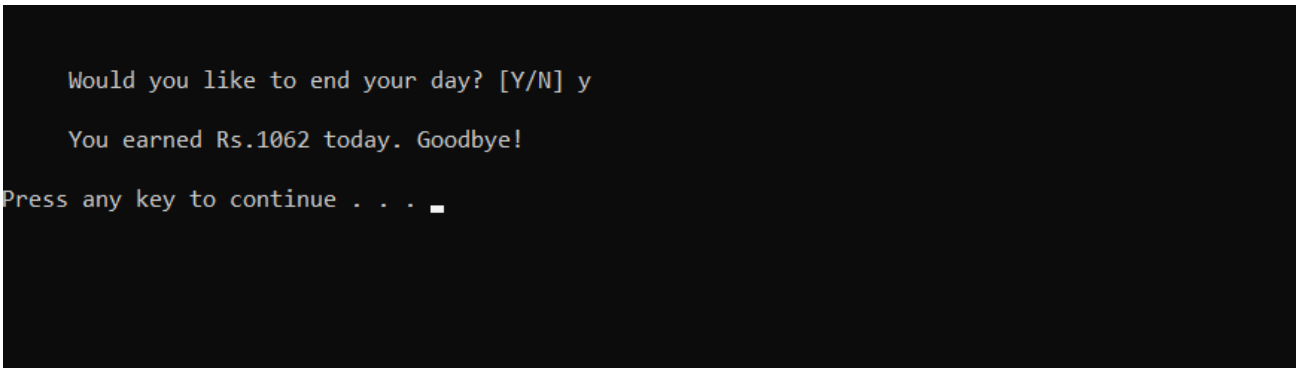
Logged in as a driver and successful authentication



Logged in as a driver and failed authentication



If the driver chooses not to end their day



If the driver chooses to end their day

CONCLUSION

To summarize, we devised a cab management system using the C programming language and taking inspiration from existing fully functioning apps and projects.

Outcome came out to be a basic functioning cab service system that enables the user to login the system as a customer or a driver and proceed according to role chosen. This project also makes use of file handling so that the system files can be accessed that include the hard-coded details and coordinates of the cabs, category wise.

On logging in as a customer, the system asks the user for their credentials and their present and destination coordinates. Then they choose the category of cab they want to ride in and the nearest cab is allotted to them by the program. The ride is ended when the customer rates the ride and uses the reference code to avail a discount.

On the other hand, if the user logs in as a driver, they are needed to authenticate their identity for a successful login using the key and license number. They can then choose to end their or continue to get allotted new rides.

The output screen is formatted to keep it easy on the eyes for the user and it also sleeps for a few seconds at appropriate times to display the information.

All in all, this project runs with basic features of any cab service system and can be developed into a user interface with many more features in the future.

LIMITATIONS

1. Interlinking of the user-interface for customers and drivers is not present
2. It doesn't store the data for the customers and the credentials have to be entered every time
3. Linking our code with some sort of global mapping is not there, hence it is assumed that the cars move in a straight line
4. Some part of the code being hard -coded, expansion would be really difficult
5. User interface could be seen as dull by some users

FUTURE ASPECTS

1. It can be developed into a GUI to make the system easier and more approachable
2. It can be expanded to include more vehicles other than cars
3. Use discounts and offer for continuous users
4. Use a database to store the information of customers and drivers
5. And of course, finding out solution for each limitation mentioned in the previous slide

REFERENCES

For basic and in depth learning of the theoretical and code writing part

1. Let Us C

For error handling and improving the program

2. GeeksForGeeks
3. Quora
4. StackOverflow

For getting ideas and visualization to expand on

5. Our friends who had done project on Cab Management System in the past