

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY



B.tech | CSE

Data Structures Project

METRO MANAGEMENT SYSTEM

Submitted By:

Himanshu	21103173
Sanskar Khandelwal	21103172
Twarit Agarwal	21103171
Krish Agrawal	21103170

INTRODUCTION:

AIM:

To provide a comprehensive solution for the **Metro Management System**.

PROBLEM STATEMENT:

What is the need of a Metro Management System?

Why isn't the offline, physical system sufficient to deal with modern day problems?

Does the system actually reduce the load and make the system more effective?

Purpose of the Software:

- The system allows interactive, self-describing working environment where even standalone users can work very comfortably and easily.
- The project simulates (on a rudimentary level) a metro management system. A metro management system eliminates the logistical nightmare that a physical ticket booking system entails, eliminating problems such as keeping track of ticket fares, reducing paper waste, reducing corruption in the department, reduces redundancies and human error.
- This system contains a User module to provide different facilities to the customer. There is also an admin module where admin can add stations, trains, routes and also update the fares. The admin is a panel consisting of a group of authorized persons.
- All the data pertaining to contact information is kept at central database from where it can be easily retrieved, viewed or updated. But, such kind of technical details are hidden from the standalone user. He just needs to select the relevant option from the given menu-driven interface.

Materials and Methods:

The System is designed using C++ Programming Language and Data Structures. To make the system more interactive and user-friendly, several functionalities have been provided. Proper files are maintained for storing the data and reducing data redundancy. Separate functionalities are made like:

+ Metro Card:

Users need to login with their card number and password and can recharge their tickets. This system shows the users balance and their journey details.

+ Metro stations and fare:

This feature allows the user to see all the metro stations and their respective fares. Users are required to enter the source and destination station, when they enter the data then the system will display fair details and the route map.

+ New card Registration:

A person can register for new card through payment via online mode or can a book a station for cash payment.

+ Functional requirements:

System must be able to retrieve information from the database. File handling is being used in this system.

HARDWARE AND SOFTWARE REQUIREMENTS:

To create a Metropolis management system that is user-friendly, the software is capable enough to allow the concerned person to store and retrieve any type of record. We have to make use of Data Structures and Algorithms. So we decided to use Visual Studio Code for optional coding and easy debugging. Henceforth for optimal usage of such software, a windows-based operating system – preferably, Windows 10 must be there. Also, There must be enough storage available on the device to store the data in form of files.

Tools and Technologies used:

In this project, we will make extensive use of Data Structures and C++ Programming Language. We will have a login id system initially. Separate files will be assembled for different features of the Metro Management System. The following Data Structures will be implemented for the proper working of the project:

- ✓ Arrays and Linked lists
- ✓ Searching
- ✓ File Handling

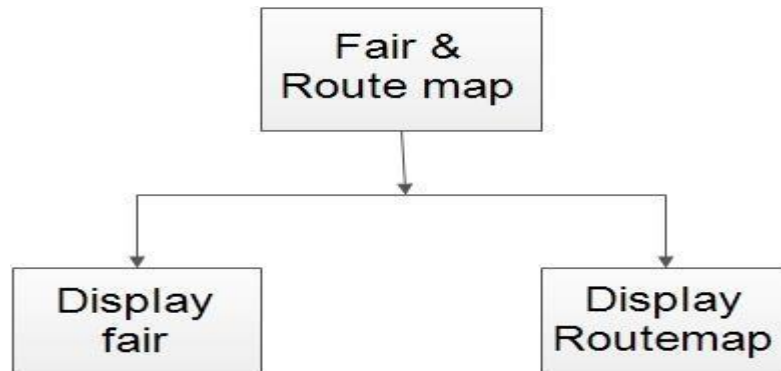
ABSTRACT:

This system is developed keeping in view the general need required by the user while using the Metro Department. Hence, following functionalities are provided:

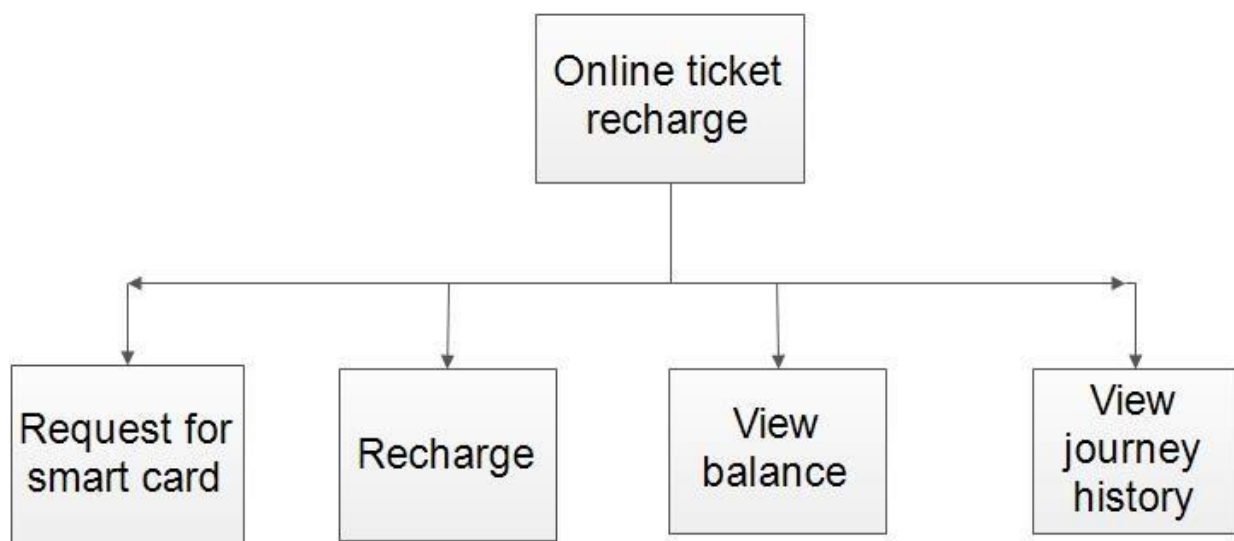
- An Admin login page, where the admin has the authority to add or delete or update the details of metro stations, update routes, and update fares between stations.
- A User login page where users are provided with various options like:
 - ✓ Apply for a new metro card
 - ✓ Check routes for their journey
 - ✓ Check fares between two stations
 - ✓ Check card balance and recharge metro card
 - ✓ Book tickets and pay bills using card or cash
- To provide the search result within a short interval of time, optimized search algorithm code has been incorporated which will provide the results within seconds.
- The background processing system will take care of all processing tasks and maintain data integrity to reduce the redundancy of data.

OVERVIEW:

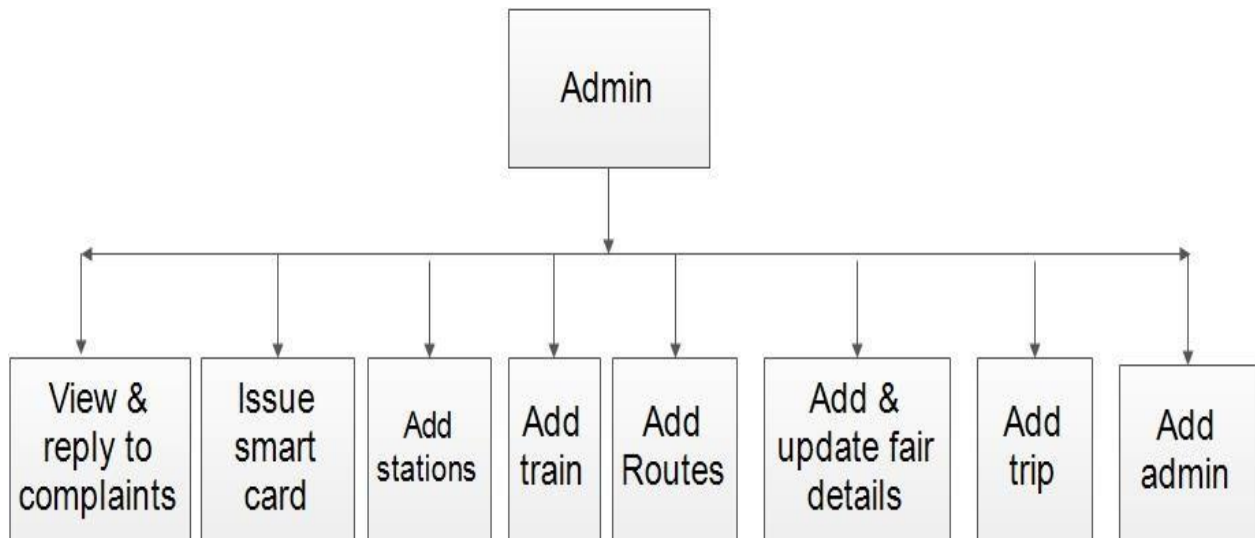
Fair and route map module: This module contains various facilities like display fair and display route map.



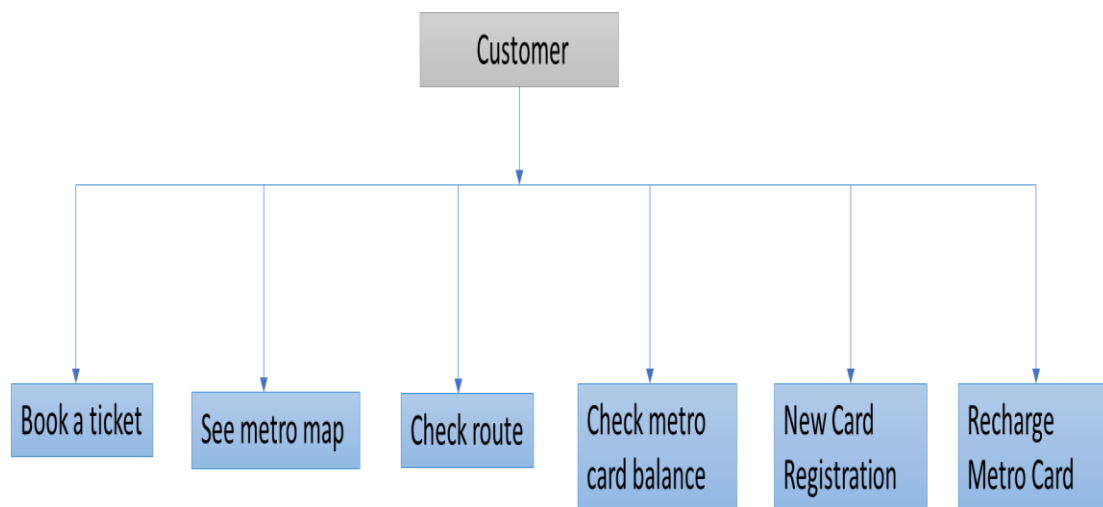
Online Metro Card Recharge Module: This system also contains an online card recharge module where users can recharge their smart cards online through the site.



Admin Menu: There is an admin module where admin can add stations, trains, routes and also update the fairs. The admin is a panel consisting of a group of authorized persons.

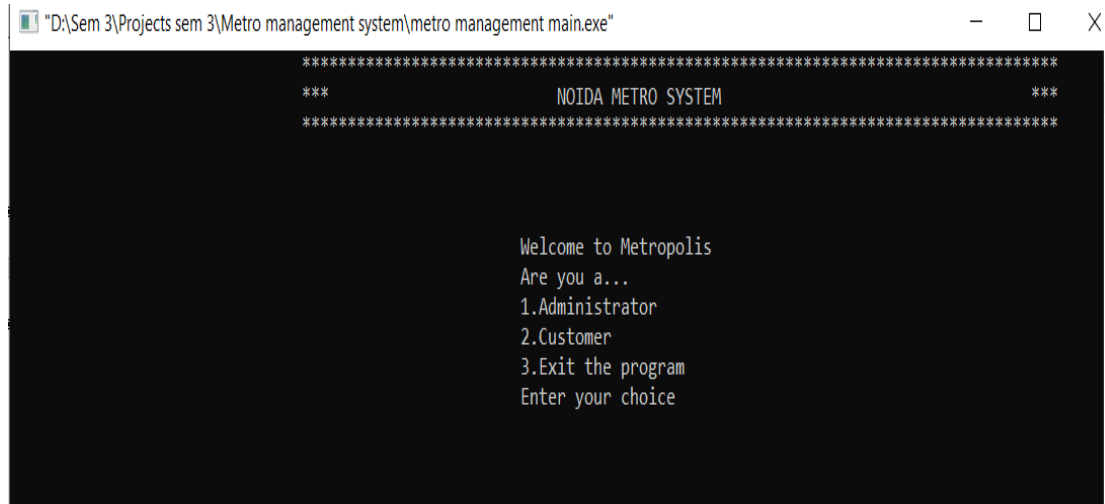


Customer Menu: There is a customer module that provides various facilities to the customers. They can book tickets, see their previous journeys, recharge metro cards, etc.



WORKING OF THE PROJECT:

MENU: To guide the users how to use this system for optimum usage, a user is given choices at each point to access different portals of the system.

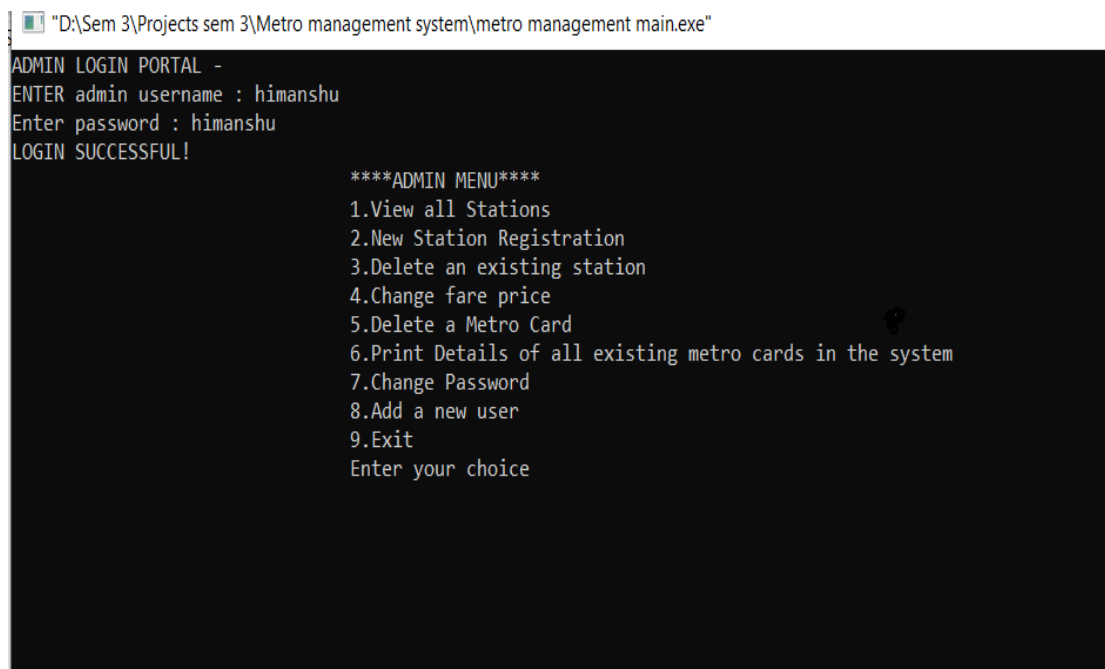
A screenshot of a Windows application window titled "D:\Sem 3\Projects sem 3\Metro management system\metro management main.exe". The window has a black background with white text. At the top, it says "NOIDA METRO SYSTEM" between two lines of asterisks. Below that, it says "Welcome to Metropolis" and "Are you a...". Then it lists three options: "1.Administrator", "2.Customer", and "3.Exit the program". At the bottom, it says "Enter your choice".

```
"D:\Sem 3\Projects sem 3\Metro management system\metro management main.exe"

*****
***                                ***
*****
NOIDA METRO SYSTEM
*****

Welcome to Metropolis
Are you a...
1.Administrator
2.Customer
3.Exit the program
Enter your choice
```

If the user press 1, then he is taken to the admin login portal where he has to enter right credentials to access admin menu.

A screenshot of the same Windows application window, now showing the admin login portal. It prompts the user to enter an admin username and password. After successful login, it displays the "ADMIN MENU" with nine options: "1.View all Stations", "2.New Station Registration", "3.Delete an existing station", "4.Change fare price", "5.Delete a Metro Card", "6.Print Details of all existing metro cards in the system", "7.Change Password", "8.Add a new user", and "9.Exit". It ends with "Enter your choice".


```
"D:\Sem 3\Projects sem 3\Metro management system\metro management main.exe"

ADMIN LOGIN PORTAL -
ENTER admin username : himanshu
Enter password : himanshu
LOGIN SUCCESSFUL!

****ADMIN MENU****
1.View all Stations
2.New Station Registration
3.Delete an existing station
4.Change fare price
5.Delete a Metro Card
6.Print Details of all existing metro cards in the system
7.Change Password
8.Add a new user
9.Exit
Enter your choice
```

Description of Feature

1. Admin can view all the stations.

 "D:\Sem 3\Projects sem 3\Metro management system\metro management main.e)

```
1
                                     Station -
-----
Station Number : 1
Station Name   : sector62
-----
                                     Station -
-----
Station Number : 2
Station Name   : sector59
-----
                                     Station -
-----
Station Number : 3
Station Name   : sector61
-----
                                     Station -
-----
Station Number : 4
Station Name   : sector52
-----
                                     Station -
-----
Station Number : 5
Station Name   : sector51
-----
                                     Station -
-----
Station Number : 6
Station Name   : sector34
-----
                                     Station -
-----
Station Number : 7
Station Name   : sector30
-----
                                     Station -
-----
Station Number : 8
Station Name   : sector25
-----
                                     Station -
-----
Station Number : 9
Station Name   : sector18
-----
```


2. **New Station Registration :-** admin can add up to as many station through nodes.

```

Station -
-----
Station Number : 1
Station Name   : sector62
-----
Station -
-----
Station Number : 2
Station Name   : sector59
-----
Station -
-----
Station Number : 3
Station Name   : sector61
-----
Station -
-----
Station Number : 4
Station Name   : Ambala
-----
Station -
-----
Station Number : 5
Station Name   : sector52
-----
Station -
-----
Station Number : 6
Station Name   : sector51
-----
Station -
-----
Station Number : 7
Station Name   : sector34
-----
Station -
-----
Station Number : 8
Station Name   : sector30
-----
Station -
-----
Station Number : 9
Station Name   : sector25
-----
Station -
-----
Station Number : 10
Station Name   : sector18
-----
Station -
-----
Station Number : 11
Station Name   : botanical
-----
Station -
-----
Station Number : 12
Station Name   : knowledge_park
-----
```

After adding some stations, the updated file is shown above.

3. Delete existing station :- A station is inserted at the 4th position, named AMBALA, after deletion this is reflected in the screenshot below.

```
1
                                     Station -
-----
Station Number : 1
Station Name   : sector62
-----
                                     Station -
-----
Station Number : 2
Station Name   : sector59
-----
                                     Station -
-----
Station Number : 3
Station Name   : sector61
-----
                                     Station -
-----
Station Number : 4
Station Name   : sector52
-----
                                     Station -
-----
Station Number : 5
Station Name   : sector51
-----
                                     Station -
-----
Station Number : 6
Station Name   : sector34
-----
                                     Station -
-----
Station Number : 7
Station Name   : sector30
-----
                                     Station -
-----
Station Number : 8
Station Name   : sector25
-----
                                     Station -
-----
Station Number : 9
Station Name   : sector18
-----
                                     Station -
-----
Station Number : 10
Station Name   : botanical
-----
                                     Station -
-----
Station Number : 11
Station Name   : knowledge_park
-----
```

4. **Change fare price:-** through this price per each station can be regulated(increased or decreased).

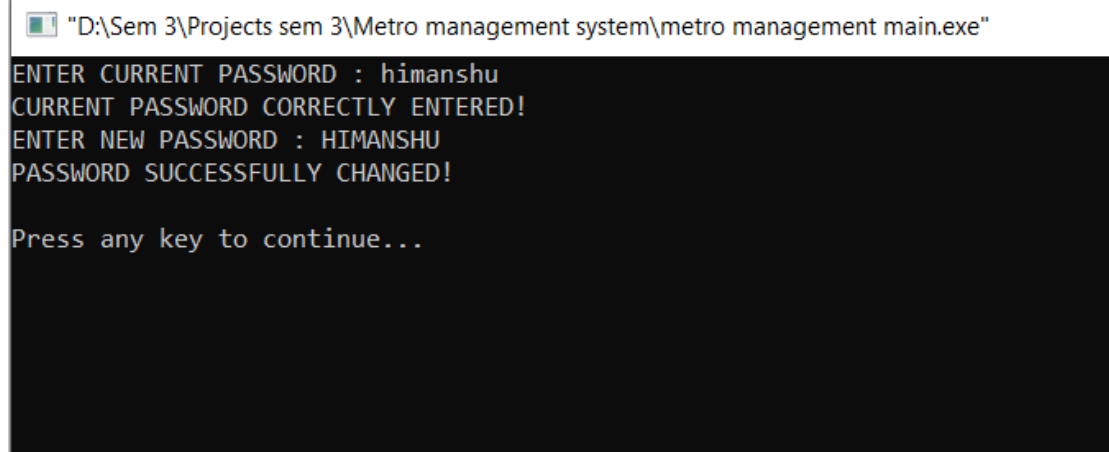
```
****ADMIN MENU****
1.View all Stations
2.New Station Registration
3.Delete an existing station
4.Change fare price
5.Delete a Metro Card
6.Print Details of all existing metro cards in the system
7.Change Password
8.Add a new user
9.Exit
Enter your choice
4
Current Fare Price = 20
Enter new fare price = 25
```

5. **Delete a Metro Card:-** admins can access any user's metro card and the can delete them

6. **View All Metro Cards:** Amins can view the details of all existing Metro cards in the system.

```
"D:\Sem 3\Projects sem 3\Metro management system\metro management main.exe"
1.View all Stations
2.New Station Registration
3.Delete an existing station
4.Change fare price
5.Delete a Metro Card
6.Print Details of all existing metro cards in the system
7.Change Password
8.Add a new user
9.Exit
Enter your choice
6
CREDIT CARD -
-----
CARD HOLDER : sanskar
CREDIT CARD NUMBER : 3714496353984319
BALANCE : 120
-----
CREDIT CARD -
-----
CARD HOLDER : himanshu
CREDIT CARD NUMBER : 6250941006528599
BALANCE : 415
-----
Press any key to continue...
```

7. **Change password:** Admins can change their password to ensure safety of their account. For this, they have enter their current password first, then only they get the privilege to change the password.



```
"D:\Sem 3\Projects sem 3\Metro management system\metro management main.exe"
ENTER CURRENT PASSWORD : himanshu
CURRENT PASSWORD CORRECTLY ENTERED!
ENTER NEW PASSWORD : HIMANSHU
PASSWORD SUCCESSFULLY CHANGED!
Press any key to continue...
```

➤ **Before moving to the Customer Module, lets have a look at the structures of different classes:**

A vector class for credit card is created used mainly for storing credit card number , making payment and storing user name . It contains function like:-

- ❖ Checking the balance in credit card **void check balance()**
- ❖ Update amount in Metro card via payment **void card Recharge()**
- ❖ Print details of credit card holder once payment is made **void printCred()**

Before accessing any data in admin, a window will pop up verifying the user of the portal through a simple code of encryption and decryption. The admin portal is main body of code as it includes deletion and addition of data.

Functional Requirements –

The system must allow admin to view all stations, add train, stations ,routes, fair ,metro timetable and even add a new admin.

The system should be designed in such a way that only authorized people should be allowed to access some particular modules.

The records should be modified by only administrators and no one else. All these are done using different function prototypes.

➤ **The structures used in the making of this project:-**

1. Vectors :

Vectors are the same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the container. Vector elements are placed in contiguous storage so that they can be accessed and traversed using iterators. In vectors, data is inserted at the end. Inserting at the end takes differential time, as sometimes there may be a need of extending the array. Removing the last element takes only constant time because no resizing happens. Inserting and erasing at the beginning or in the middle is linear in time.

Vector functions used in the program are –

a. **size()** – Returns the number of elements in the vector.

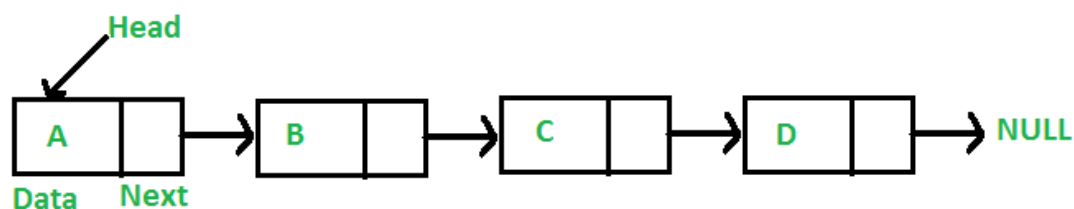
b. **erase()** – It is used to remove elements from a container from the specified position or range.

c. **vector::push_back()**

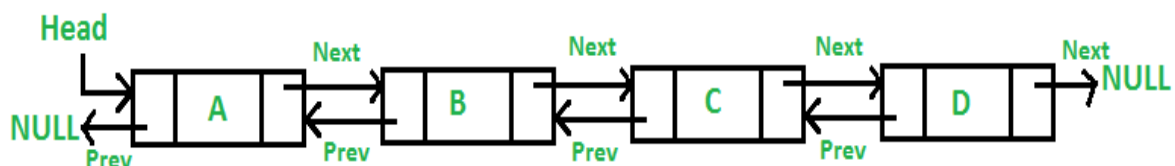
push_back() function is used to push elements into a vector from the back. The new value is inserted into the vector at the end, after the current last element and the container size is increased by 1.

2. Linked List:

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers as shown in the below image:



Singly Linked List



Doubly Linked List

In simple words, a linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list.

A **Doubly Linked List** (DLL) contains an extra pointer, typically called *previous pointer*, together with next pointer and data which are there in singly linked list.

3. File Handling :

In C++, files are mainly dealt by using three classes `fstream`, `ifstream`, `ofstream` available in `fstream` headerfile.

ofstream: Stream class to write on files

ifstream: Stream class to read from files

fstream: Stream class to both read and write from/to files.

4. Luhn's Algorithm :

So, what's the secret formula? Well, most cards use an algorithm invented by Hans Peter Luhn of IBM. According to Luhn's algorithm, you can determine if a credit card number is (syntactically) valid as follows:

Multiply every other digit by 2, starting with the number's second-to-last digit, and then add those products' digits together.

Add the sum to the sum of the digits that weren't multiplied by 2.

If the total's last digit is 0 (or, put more formally, if the total modulo 10 is congruent to 0), the number is valid!

That's kind of confusing, so let's try an example with David's Visa: 4003600000000014.

For the sake of discussion, let's first underline every other digit, starting with the number's second-to-last digit: 4003600000000014

Okay, let's multiply each of the underlined digits by 2:

$$1 \cdot 2 + 0 \cdot 2 + 0 \cdot 2 + 0 \cdot 2 + 0 \cdot 2 + 6 \cdot 2 + 0 \cdot 2 + 4 \cdot 2$$

That gives us: $2 + 0 + 0 + 0 + 0 + 12 + 0 + 8$

Now let's add those products' digits (i.e., not the products themselves) together:

$$2 + 0 + 0 + 0 + 0 + 1 + 2 + 0 + 8 = 13$$

Now let's add that sum (13) to the sum of the digits that weren't multiplied by 2 (starting from the end):

$$13 + 4 + 0 + 0 + 0 + 0 + 0 + 3 + 0 = 20$$

Yup, the last digit in that sum (20) is a 0, so David's card is legit!

So, validating credit card numbers isn't hard, but it does get a bit tedious by hand. That's why we have used the `check_validity(long long Input)` function to do this for us.

WORKING OF THE PROJECT:

CUSTOMER MENU:

If the user enter choice 2 in the main menu, he is directed to the customer menu where he gets several function as a customer of the metro management system. A new menu will pop that takes the customer to the customer portal.

```
HI!
*****
***          NOIDA METRO SYSTEM          ***
*****

Welcome to Metropolis
Are you a...
1.Administrator
2.Customer
3.Exit the program
Enter your choice

2
```

```
"D:\Sem 3\Projects sem 3\Metro management system\metro management main.exe"

*****CUSTOMER MENU*****

1.Book a ticket
2.See the Metro map
3.Check Route
4.Check metro card balance
5.New Card Registration
6.Recharge metro card
7.View your tickets
8.Exit
ENTER YOUR CHOICE
```

Description of the Features:

1. **Book a ticket:** A user can book a ticket between two stations. He just have give his details like his name and phone number and make payment.

```
"D:\Sem 3\Projects sem 3\Metro management system\metro management main.exe"

*****CUSTOMER MENU*****

1.Book a ticket
2.See the Metro map
3.Check Route
4.Check metro card balance
5.New Card Registration
6.Recharge metro card
7.View your tickets
8.Exit
ENTER YOUR CHOICE

1
Please enter the name of your current station
sector62
Please enter your destination
sector51
```

```
"D:\Sem 3\Projects sem 3\Metro management system\

STARTING POINT:

Station Number:1
Station Name:sector62

Station Number:2
Station Name:sector59

Station Number:3
Station Name:sector61

Station Number:4
Station Name:sector52

Station Number:5
Station Name:sector51

ENDING POINT:
Number of stations to travel:4
Total Cost:100
Do you wish to proceed?Enter Y or N: y
```

```
"D:\Sem 3\Projects sem 3\Metro management system\metro

How do you wish to make the payment
1.Cash
2.Metro Card
3.Exit
Enter your choice
1
-----
Enter your name: krish
Enter your phone: 7742440811
-----
Booking successful
Press Enter
```

2. View the metro map: A customer can view the whole metro map before booking a ticket to ensure he is on the correct platform.

```
2
Station Number:1
Station Name:sector62

Station Number:2
Station Name:sector59

Station Number:3
Station Name:sector61

Station Number:4
Station Name:sector52

Station Number:5
Station Name:sector51

Station Number:6
Station Name:sector34

Station Number:7
Station Name:sector30

Station Number:8
Station Name:sector25

Station Number:9
Station Name:sector18

Station Number:10
Station Name:botanical

Station Number:11
Station Name:knowledge_park

Press any key to continue...
```

3. **Check route between stations:** A customer can check the route between two stations. All he need to do is to enter the stations and he can check all the stations between them.

```
3
Enter station number 1:sector62
Enter station 2: sector61
STARTING POINT:


Station Number:1
Station Name:sector62

Station Number:2
Station Name:sector59

Station Number:3
Station Name:sector61

ENDING POINT:
Press any key to continue...
```

4. **Check balance:** To check balance, a user has to enter his metro card number and then his balance will show.

 "D:\Sem 3\Projects sem 3\Metro management system\metro management main.exe"

```
*****CUSTOMER MENU*****

1.Book a ticket
2.See the Metro map
3.Check Route
4.Check metro card balance
5.New Card Registration
6.Recharge metro card
7.View your tickets
8.Exit
ENTER YOUR CHOICE

4
ENTER METRO CARD NUMBER : 6250941006528599
BALANCE IN CARD IS : 415
Press any key to continue...
```

5. **New Card Registration:** A user can register for a new card. He just have enter few details. Then his card number is checked using Luhn's Algorithm and he is assigned a new card.

```
"D:\Sem 3\Projects sem 3\Metro management system\metro management main.exe"

*****CUSTOMER MENU*****

1.Book a ticket
2.See the Metro map
3.Check Route
4.Check metro card balance
5.New Card Registration
6.Recharge metro card
7.View your tickets
8.Exit
ENTER YOUR CHOICE

5

ENTER DETAILS TO CREATE NEW CARD - ....
ENTER CARD HOLDER NAME : himanshu
PLEASE ENTER A CREDIT CARD NUMBER FOR VERIFICATION!
4556590909693921
YOUR CARD IS : VISA
ENTER BALANCE IN CARD : 155
Press any key to continue...
```

6. **Recharge metro card:** A user can recharge his metro card in case there is insufficient balance.

```
"D:\Sem 3\Projects sem 3\Metro management system\metro management main.exe"


*****CUSTOMER MENU*****

1.Book a ticket
2.See the Metro map
3.Check Route
4.Check metro card balance
5.New Card Registration
6.Recharge metro card
7.View your tickets
8.Exit
ENTER YOUR CHOICE

6

ENTER METRO CARD NUMBER : 4251760001841321
ENTER AMOUNT TO BE TOPPED-UP : 3000
CURRENT BALANCE : 3150
Press any key to continue...
```

7. **View your tickets:** A customer can view all his past journeys just by entering his name and phone number.


 "D:\Sem 3\Projects sem 3\Metro management system\metro management main.exe"

```
Enter your name: himanshu
Enter your phone: 9963524187
```

```
1.
```

```
-----
Source Station :    botanical
Destination Station: sector59
Method of Payment : Card
Cost of the ticket: 200
-----
```

```
Press any key to continue....
```

 "D:\Sem 3\Projects sem 3\Metro management system"

```
Enter your name: krish
Enter your phone: 7732440811
```

```
1.
```

```
-----
Source Station :    sector62
Destination Station: sector51
Method of Payment : Cash
Cost of the ticket: 100
-----
```

```
2.
```

```
-----
Source Station :    sector62
Destination Station: sector34
Method of Payment : Cash
Cost of the ticket: 150
-----
```

```
Press any key to continue....
```

Source Code:

```
#include <iostream>
#include <cstring>
#include <cmath>
#include <string>
#include <windows.h>
#include <stdlib.h>
#include <vector>
#include <cstring>
#include <fstream>
#include <conio.h>
#include <unistd.h>
#include <bits/stdc++.h>

using namespace std;

// Prototypes
int farecost=10;
class CreditCard;
class admin;
class station;
vector<admin> users;
vector<CreditCard> C;

int cost_calculation(station*, string, string);
void check_route(station*, string, string);
void customer_menu(station*);
int check_validity(long long);
void delCard();
void newCard();
admin *adminlogin();

class station {
    string station_name;
    int station_number;
public:
    static int count;
    station *next;
    station *previous;
    station()
    {
        cout << "Enter station name" << endl;
        cin >> station_name;
        count++;
        station_number = count;
        next = NULL;
        previous = NULL;
    }
    station(string n, int sn)
    {
        station_name = n;
        station_number = sn;
    }
};
```

```

        count++;
        next = NULL;
        previous = NULL;
    }
    void printstat(){
        cout << "\t\t\t\tStation - " << endl;
        cout << "-----" << endl;
        cout << "Station Number : " << station_number << endl;
        cout << "Station Name  : " << station_name << endl;
        cout << "-----" << endl;
    }
    string getName()
    {
        return station_name;
    }
    int getNumber()
    {
        return station_number;
    }
    void IncreaseStationNumber()
    {
        station_number++;
    }
    void DecreaseStationNumber()
    {
        station_number--;
    }
};
int station::count = 0;

class CreditCard {
    string username;
    long long cred;
    float balance;

public:
    void create(string n, long long num, float bal)
    {
        cred = num;
        username = n;
        balance = bal;
    }
    void update_amount(string name, long long phone, string s1, string s2, int bill)
    {
        fstream tick;
        tick.open("ticket.txt", ios_base::app);
        cout << "CURRENT AMOUNT : " << balance << endl;
        cout << "CHARGES : " << bill << endl;
        if (bill > balance)
        {
            cout << "-----" << endl;
            cout << "NOT ENOUGH BALANCE IN ACCOUNT! PAYMENT FAILED!\n";
            cout << "-----" << endl;
            return;
        }
        balance -= bill;
    }
};

```



```

tick<<name<<endl<<phone<<endl<<s1<<endl<<s2<<endl<<"Card"<<endl<<bill<<endl;
cout << "PAYMENT SUCCESSFULLY DONE! THANK YOU FOR USING NON-CONTACT METHOD OF
PAYMENT." << endl;
cout << endl<<"BALANCE LEFT : " << balance << endl;
ofstream re;
int counter = 0;
re.open("creditCards.txt");
for (CreditCard o : C)
{
    if (counter != C.size() - 1)
    {
        re << o.info_name() << endl;
        re << o.info_cred() << endl;
        re << o.info_balance() << endl;
    }
    else
    {
        re << o.info_name() << endl;
        re << o.info_cred() << endl;
        re << o.info_balance();
    }
    counter++;
}
re.close();
tick.close();
}

void printCred()
{
    cout << "\t\t\t\t\tCREDIT CARD - " << endl;
    cout << "-----" << endl;
    cout << "CARD HOLDER : " << username << endl;
    cout << "CREDIT CARD NUMBER : " << cred << endl;
    cout << "BALANCE : " << balance << endl;
    cout << "-----" << endl;
}

void card_Recharge()
{
    float recharge;
    cout << "ENTER AMOUNT TO BE TOPPED-UP : ";
    cin >> recharge;
    balance += recharge;
    cout << "CURRENT BALANCE : " << balance;
    ofstream re;
    re.open("creditCards.txt");
    int counter = 0;
    for (CreditCard o : C)
    {
        if (counter != C.size() - 1)
        {
            re << o.info_name() << endl;
            re << o.info_cred() << endl;
            re << o.info_balance() << endl;
        }
        else
        {
            re << o.info_name() << endl;

```

```

        re << o.info_cred() << endl;
        re << o.info_balance();
    }
    counter++;
}
re.close();
}

void check_balance()
{
    cout << "BALANCE IN CARD IS : " << balance;
}

string info_name()
{
    return username;
}
long long info_cred()
{
    return cred;
}
float info_balance()
{
    return balance;
}
};

```

```

class admin {
    string username;
    string password;

public:
    void printStationDetails(station **list){
        station*temp=*list;
        while(temp!=NULL){
            temp->printstat();
            temp=temp->next;
        }
    }

    void printCardDetails(vector<CreditCard> C)
    {
        for (CreditCard temp : C)
        {
            temp.printCred();
        }
    }
    void setData(string n, string p)
    {
        username = n;
        password = p;
    }
    void newUser(vector<admin> &l)
    {
        admin obj;

```

```

cout<<"-----"<<endl;
cout << "\nCREATING NEW USER...." << endl;
cout << "ENTER USERNAME : ";
cin >> username;
string encryp;
cout << "ENTER PASSWORD : ";
cin >> encryp;
password = encryp;
for(admin o: l){
    string temp=o.username_info();
    if(temp==username){
        cout<<"-----"<<endl;
        cout<<"userid already exists.....";
        sleep(2);
        system("cls");
        return;
    }
}
sleep(1);
cout<<"-----"<<endl;
cout<<"New User added to the system....\n";
cout<<"Press any key to continue...";
getchar(); getchar();
system("cls");

obj.setData(username, password);
l.push_back(obj);

ofstream f2;
f2.open("admins.txt");
for (admin o : l)
{
    f2 << o.username_info() << endl;
    f2 << o.password_info() << endl;
}
f2.close();

/*ofstream f;
f.open("ADMIN_INFO.dat", ios::binary | ios::app);

f.write(reinterpret_cast<char *>(&obj), sizeof(admin));
cout << "NEW USER SUCCESSFULLY CREATED! " << endl;
cout << "DETAILS WILL BE REFLECTED WHEN PROGRAM IS RUN AGAIN!" << endl;
f.close();*/
}

int check(string u, string p)
{
    if (username == u && password == p)
    {
        return 1;
    }
    return 0;
}

void changePassword()
{

```



```

string name, pass;
cout << "ENTER admin username : ";
cin >> name;
cout << "Enter password : ";
cin >> pass;
admin *now;
for (int i = 0; i < users.size(); i++)
{
    if (users[i].check(name, pass))
    {
        now = &users[i];
        cout << "LOGIN SUCCESSFUL!" << endl;
        return now;
        break;
    }
    if (i == users.size() - 1)
    {
        cout << "NO SUCH USER FOUND! CHECK USERNAME OR PASSWORD!" << endl;
        return NULL;
    }
}
return NULL;
}

```

```

void AddNode(station **list, string n, int sn) {
    station *s = new station(n, sn);
    if (*list == NULL)
    {
        *list = s;
    }
    else if ((*list)->previous == NULL && (*list)->next == NULL)
    {
        (*list)->next = s;
        s->previous = *list;
    }
    else
    {
        station *p;
        for (p = *list; p->next != NULL; p = p->next)
            ;
        p->next = s;
        s->previous = p;
    }
}

```

```

void readFile(station **list) {
    ifstream f1;
    f1.open("stations.txt");
    int t;
    f1>>t;
    while (t--)
    {
        string s;
        f1 >> s;
        int i;
    }
}

```

```

        f1 >> i; // 1
        AddNode(list, s, i);
    }
    f1>>farecost;
    f1.close();
}

```

```

void AddNodeByAdmin(station **list, int pos)
{
    cout << "Enter station name\n";
    string s1;
    cin >> s1;
    station* temp1=*list;
    while(temp1!=NULL){
        string tt=temp1->getName();
        if(tt==s1) goto l1;
        temp1=temp1->next;
    }
    {station *s = new station(s1, pos);
    if (*list == NULL && pos == 1)
    {
        *list = s;
    }
    else if ((*list)->previous == NULL && (*list)->next == NULL && pos == 1)
    {
        (*list)->previous = s;
        s->next = *list;
        (*list)->IncreaseStationNumber();
        *list = s;
    }
    else if ((*list)->previous == NULL && (*list)->next == NULL && pos == 2)
    {
        (*list)->next = s;
        s->previous = *list;
    }
    else if ((*list)->previous != NULL && (*list)->next != NULL && pos == 1)
    {
        s->next = *list;
        (*list)->previous = s;
        station *p = *list;
        *list = s;
        while (p != NULL)
        {
            p->IncreaseStationNumber();
            p = p->next;
        }
    }
    else if (pos == station::count)
    {
        station *p;
        for (p = *list; p->next != NULL; p = p->next)
            ;
        p->next = s;
        s->previous = p;
    }
    else

```

```

{
    station *p;
    for (p = *list; p != NULL; p = p->next)
    {
        if (p->getNumber() == pos)
        {
            break;
        }
    }
    (p->previous)->next = s;
    s->previous = p->previous;
    s->next = p;
    p->previous = s;
    while (p != NULL)
    {
        p->IncreaseStationNumber();
        p = p->next;
    }
}
cout << "Station Added successfully\n";
}
if(0){
    l1: cout << "Station Already Exists\n";
}
}
}

```

```

void DeleteNode(station **list, int pos)
{
    station::count--;
    if (*list == NULL)
    {
        station::count = 0;
        cout << "There are no stations\n";
    }
    else if ((*list)->next == NULL && pos == 1)
    {
        delete *list;
        *list = NULL;
    }
    else if (pos == 1)
    {
        station *p = (*list)->next;
        delete *list;
        *list = p;
        while (p != NULL)
        {
            p->DecreaseStationNumber();
            p = p->next;
        }
    }
    else if ((*list)->next == NULL && pos != 1)
    {
        station *p;
        for (p = *list; p->next != NULL; p = p->next)
            ;
        (p->previous)->next = NULL;
    }
}

```

```

        delete p;
    }
    else
    {
        station *p, *p1;
        for (p = *list; p != NULL; p = p->next)
        {
            if (p->getNumber() == pos)
                break;
        }
        (p->previous)->next = p->next;
        if(p->next!=NULL) (p->next)->previous = p->previous;
        p1 = p->next;
        delete p;
        while (p1 != NULL)
        {
            p1->DecreaseStationNumber();
            p1 = p1->next;
        }
    }
}

```

```

void writefile(station *list)
{
    ofstream f1;
    f1.open("stations.txt");
    f1<<station::count<<endl;
    for (station *p = list; p != NULL; p = p->next)
    {
        f1 << p->getName() << endl;
        if (p->next == NULL)
            f1 << p->getNumber();
        else
            f1 << p->getNumber() << endl;
    }
    f1<<endl<<farecost;
    f1.close();
}

```

```

void displayList(station *list)
{
    for (station *p = list; p != NULL; p = p->next)
    {
        cout << "Station Number:" << p->getNumber() << endl;
        cout << "Station Name:" << p->getName() << endl
            << endl;
    }
}

```

```

int cost_calculation(station *list, string s1, string s2)
{
    int ssn = 0, esn = 0;
    transform(s1.begin(), s1.end(), s1.begin(), ::toupper);
    transform(s2.begin(), s2.end(), s2.begin(), ::toupper);
}

```



```

station *p;
for (p = list; p != NULL; p = p->next)
{
    string s3 = p->getName();
    transform(s3.begin(), s3.end(), s3.begin(), ::toupper);
    if (s1 == s3)
        ss = p->getNumber();
    if (s2 == s3)
        es = p->getNumber();
}
int tcost = (es - ss > 0 ? es - ss : ss - es) * farecost;
return tcost;
}

```

```

void delCard() {
    sleep(1);
    system("cls");
    long long test;
    cout << "ENTER METRO CARD NUMBER to be deleted : ";
    cin >> test;
    int flag = 0;
    vector<CreditCard>::iterator q = C.begin();
    for (int i = 0; i < C.size(); i++)
    {
        if (C[i].info_cred() == test)
        {
            flag = 1;
            C.erase(C.begin() + i);
            cout << "CARD SUCCESSFULLY DELETED!" << endl;
            break;
        }
    }
    if (flag == 0)
    {
        cout << "NO RECORD FOUND OF SUCH CARD! INVALID INPUT! " << endl;
        return;
    }
    ofstream re;
    re.open("creditCards.txt");
    int counter = 0;
    for (CreditCard o : C)
    {
        if (counter != C.size() - 1)
        {
            re << o.info_name() << endl;
            re << o.info_cred() << endl;
            re << o.info_balance() << endl;
        }
        else
        {
            re << o.info_name() << endl;
            re << o.info_cred() << endl;
            re << o.info_balance();
        }
        counter++;
    }
}

```

```

    re.close();
}

int check_station_Vailidity(station *list, string s1)
{
    transform(s1.begin(), s1.end(), s1.begin(), ::toupper);
    for (station *p = list; p != NULL; p = p->next)
    {
        string s2 = p->getName();
        transform(s2.begin(), s2.end(), s2.begin(), ::toupper);
        if (s1 == s2)
            return (p->getNumber());
    }
    return -1;
}

void check_route(station *list, string s1, string s2)
{
    transform(s1.begin(), s1.end(), s1.begin(), ::toupper);
    transform(s2.begin(), s2.end(), s2.begin(), ::toupper);
    station *p;
    bool b = false;
    int n1 = check_station_Vailidity(list, s1), n2 = check_station_Vailidity(list, s2);
    if (n1 == n2 || n1 == -1 || n2 == -1)
    {
        cout << "Invalid Input" << endl;
        return;
    }
    else if (n1 < n2)
    {
        for (p = list; p != NULL; p = p->next)
        {
            string s3 = p->getName();
            transform(s3.begin(), s3.end(), s3.begin(), ::toupper);
            if (s1 == s3)
            {
                b = true;
                cout << "STARTING POINT:" << endl
                    << endl;
            }
            if (b)
            {
                cout << "Station Number:" << p->getNumber() << endl;
                cout << "Station Name:" << p->getName() << endl
                    << endl;
            }
            if (s2 == s3)
            {
                b = false;
                cout << "\nENDING POINT:" << endl;
            }
        }
    }
    else
    {

```

```

    for (p = list; p != NULL; p = p->next)
    {
        if (n1 == p->getsNumber())
            break;
    }
    cout << "STARTING POINT:" << endl
        << endl;
    while (p->getsNumber() != n2)
    {
        cout << "Station Number:" << p->getsNumber() << endl;
        cout << "Station Name:" << p->getName() << endl
            << endl;
        p = p->previous;
    }
    cout << "Station Number:" << p->getsNumber() << endl;
    cout << "Station Name:" << p->getName() << endl;
    cout << "\nENDING POINT:" << endl;
}
}

```

```

void newCard()
{
    cout << "\t\t\t\tENTER DETAILS TO CREATE NEW CARD - ....\n";
    string name;
    long long credit;

    cout << "ENTER CARD HOLDER NAME : ";
    cin >> name;
    // PROMPTS THE USER TO ENTER A CREDIT CARD NUMBER
    cout << "PLEASE ENTER A CREDIT CARD NUMBER FOR VERIFICATION!\n";
    cin >> credit;
    cout << "YOUR CARD IS : ";
    if (check_validity(credit) == -1)
    {
        return;
    }
    float bal;
    cout << "ENTER BALANCE IN CARD : ";
    cin >> bal;
    CreditCard temp;
    temp.create(name, credit, bal);
    C.push_back(temp);

    ofstream re;
    re.open("creditCards.txt");
    int counter = 0;
    for (CreditCard o : C)
    {
        if (counter != C.size() - 1)
        {
            re << o.info_name() << endl;
            re << o.info_cred() << endl;
            re << o.info_balance() << endl;
        }
        else
        {

```

```

        re << o.info_name() << endl;
        re << o.info_cred() << endl;
        re << o.info_balance();
    }
    counter++;
}
re.close();
}

void book_a_ticket(station *list)
{
    fstream tick;
    tick.open("ticket.txt", ios_base::app);
    string name;
    int counter=0;
    long long phone;
    cout << "Please enter the name of your current station" << endl;
    string s1;
    cin >> s1;
    cout << "Please enter your destination\n";
    string s2;
    cin >> s2;
    int n1 = check_station_Vailidity(list, s1);
    int n2 = check_station_Vailidity(list, s2);
    if (n1 == -1 || n2 == -1)
    {
        cout << "INVALID INPUT\n";
        cout<<"Press any key to continue...";getchar();
        return;
    }
    else
    {
        system("cls");
        check_route(list, s1, s2);
        cout << "Number of stations to travel:" << ((n1 > n2) ? n1 - n2 : n2 - n1) << endl;
        cout << "Total Cost:" << cost_calculation(list, s1, s2) << endl;
        cout << "Do you wish to proceed?Enter Y or N: ";
        char ch;
        cin >> ch;
        if (ch == 'Y' || ch=='y')
        {
            system("cls");
            cout << "How do you wish to make the payment\n";
            cout << "1.Cash\n";
            cout << "2.Metro Card" << endl;
            cout << "3.Exit" << endl;
            cout << "Enter your choice" << endl;
            int choice1;
            cin >> choice1;
            ofstream er;
            CreditCard *temp = NULL;
            int flag = 0;
            switch (choice1)
            {
            case 1:
                cout<<"-----"<<endl;

```

```

        cout<<"Enter your name: ";
        cin>>name;
        cout<<"Enter your phone: ";
        cin>>phone;
        cout<<"-----"<<endl;

tick<<name<<endl<<phone<<endl<<s1<<endl<<s2<<endl<<"Cash"<<endl<<cost_calculation(list, s1,
s2)<<endl;
        cout << "Booking successful\n";
        sleep(2);
        cout << "Press Enter\n";
        getchar();
        break;
case 2: // metro card fuctions
        system("cls");

        long long test;
        cout<<"-----"<<endl;
        cout<<"Enter your name: ";
        cin>>name;
        cout<<"Enter your phone: ";
        cin>>phone;
        cout << "ENTER METRO CARD NUMBER : ";
        cin >> test;
        cout<<"-----"<<endl;

        // vector<CreditCard>::iterator q = C.begin();
        for (int i = 0; i < C.size(); i++)
        {

                if (C[i].info_cred() == test)
                {
                        flag = 1;
                        cout << "CARD FOUND IN SYSTEM!" << endl;
                        int cost = cost_calculation(list, s1, s2);
                        temp = &C[i];
                        temp->update_amount(name, phone, s1, s2, cost);
                        sleep(2);
                        cout<<"\nPress any key to continue...\n";getchar();
                        break;
                }
        }
        if (flag == 0)
        {
                cout << "NO RECORD FOUND OF SUCH CARD! INVALID INPUT! " << endl;
                return;
        }
        er.open("creditCards.txt");
        for (CreditCard o : C)
        {
                er << o.info_name() << endl;
                er << o.info_cred() << endl;
                if(counter==C.size()-1) er << o.info_balance();
                else er<<o.info_balance()<<endl;
                counter++;
        }
        er.close();

```

```

        break;
    case 3:

        break;
    default:
        cout << "Invalid Input\n";
    }
}
else if (ch == 'N' || ch=='n')
{
    cout << "BOOKING CANCELLED\n";
    cout<<"\nPress any key to continue...\n";
    getchar();
    return;
}
else
{
    cout << "INVALID INPUT\n";
    return;
}
}
tick.close();
}

void viewticket(){
    string name;
    long long phone;
    cout<<"Enter your name: ";
    cin>>name;
    cout<<"Enter your phone: ";
    cin>>phone;
    string tname;
    long long tphone;
    string s1, s2, mode;
    int cost;
    ifstream tick("ticket.txt");
    tick>>tname>>tphone>>s1>>s2>>mode>>cost;
    int sr=1;
    while(!tick.eof()){
        if(tname==name && tphone==phone){
            cout<<endl<<sr<<". "<<endl;
            cout<<"-----"<<endl;
            cout<<"Source Station :   "<<s1<<endl;
            cout<<"Destination Station: "<<s2<<endl;
            cout<<"Method of Payment : "<<mode<<endl;
            cout<<"Cost of the ticket:  "<<cost<<endl;
            cout<<"-----"<<endl;
            sr++;
        }
        tick>>tname>>tphone>>s1>>s2>>mode>>cost;
    }
    if(sr==1){
        cout<<"No tickets found...\n";
    }
    tick.close();
}

```

```

void customer_menu(station *list)
{
    int choice;
    while(choice!=8){
        cout << "\t\t\t\t\t*****CUSTOMER MENU*****\n\n\n";
        cout << "\t\t\t\t\t1.Book a ticket\n";
        cout << "\t\t\t\t\t2.See the Metro map\n";
        cout << "\t\t\t\t\t3.Check Route\n";
        cout << "\t\t\t\t\t4.Check metro card balance\n";
        cout << "\t\t\t\t\t5.New Card Registration\n";
        cout << "\t\t\t\t\t6.Recharge metro card\n";
        cout << "\t\t\t\t\t7.View your tickets\n";
        cout << "\t\t\t\t\t8.Exit\n";
        cout << "\t\t\t\t\tENTER YOUR CHOICE\n";
        string m1, m2;
        long long test;
        int flag = 0;
        cin >> choice;
        switch (choice)
        {
            case 1:
                book_a_ticket(list);
                getchar();
                sleep(2);
                system("cls");
                break;
            case 2:
                displayList(list);
                sleep(2);
                cout<<"\nPress any key to continue...";getchar();
                getchar();
                system("cls");
                break;
            case 3:
                cout << "Enter station number 1:";
                cin >> m1;
                cout << "Enter station 2: ";
                cin >> m2;
                check_route(list, m1, m2);
                sleep(2);
                cout<<"Press any key to continue...";getchar();
                getchar();
                system("cls");
                break;
            case 4:
                cout << "ENTER METRO CARD NUMBER : ";
                cin >> test;
                flag = 0;
                // vector<CreditCard>::iterator q = C.begin();
                for (int i = 0; i < C.size(); i++)
                {
                    if (C[i].info_cred() == test)
                    {
                        flag = 1;
                        C[i].check_balance();
                        break;
                    }
                }
            }
        }
    }
}

```

```

    }
}
if (flag == 0)
{
    cout << "NO RECORD FOUND OF SUCH CARD! INVALID INPUT! " << endl;
}
sleep(3);
cout<<"\nPress any key to continue...";getchar();
getchar();
getchar();
system("cls");
break;
case 5:
    newCard();
    sleep(2);
    cout<<"\nPress any key to continue...";getchar();
    getchar();
    system("cls");
    break;
case 6:
    long long test;
    cout << "ENTER METRO CARD NUMBER : ";
    cin >> test;
    // vector<CreditCard>::iterator q = C.begin();
    for (int i = 0; i < C.size(); i++)
    {

        if (C[i].info_cred() == test)
        {
            flag = 1;
            C[i].card_Recharge();
            break;
        }
    }
}
if (flag == 0)
{
    cout << "NO RECORD FOUND OF SUCH CARD! INVALID INPUT! " << endl;
}
sleep(2);
cout<<"\nPress any key to continue...";getchar();
getchar();
system("cls");
break;
case 7:
    sleep(1);
    system("cls");
    viewticket();
    cout<<"Press any key to continue....";getchar();
    getchar();
    sleep(2);
    system("cls");
    break;
case 8:
    cout << "YOU HAVE CHOSEN TO EXIT THE CUSTOMER PORTAL!" << endl;
    cout << "THANK YOU FOR USING OUR SERVICES, SEE U SOON!" << endl;
    cout << "RETURNING TO MAIN MENU" << endl;
    sleep(2);

```



```

        system("cls");
        break;
    default:
        cout << "Invalid Input\n";
        system("cls");
    }
}

}

void adminmenu(station **list, admin *now) {
    l1:
    cout << "\t\t\t\t****ADMIN MENU****" << endl;
    cout << "\t\t\t\t1.View all Stations\n";
    cout << "\t\t\t\t2.New Station Registration\n";
    cout << "\t\t\t\t3.Delete an existing station\n";
    cout << "\t\t\t\t4.Change fare price" << endl;
    cout << "\t\t\t\t5.Delete a Metro Card" << endl;
    cout << "\t\t\t\t6.Print Details of all existing metro cards in the system\n";
    cout << "\t\t\t\t7.Change Password\n";
    cout << "\t\t\t\t8.Add a new user\n";
    cout << "\t\t\t\t9.Exit" << endl;
    cout << "\t\t\t\tEnter your choice\n";
    int choice;
    cin >> choice;
    switch (choice)
    {
    case 1:
        now->printStationDetails(list);
        sleep(2);
        cout<<"\nPress any key to continue...\n";
        getchar();
        getchar();
        sleep(1);
        system("cls");
        break;
    case 2:
        cout << "Enter the position which you want to add the station\n";
        int pos;
        cin >> pos;
        if(pos<1 || pos>station::count+1) {
            cout<<"Wrong station number....\n";
        }
        else{
            AddNodeByAdmin(list, pos);
            system("cls");
            writefile(*list);
        }
        cout<<"\nPress any key to continue...\n";
        getchar();
        sleep(1);
        system("cls");
        break;
    case 3:
        cout << "Enter the station number which you want to delete\n";
        int pos1;

```

```

cin >> pos1;
if(pos1<1 || pos1>station::count) {
    cout<<"Wrong station number....\n";
}
else{
    DeleteNode(list, pos1);
    cout << "Station Deleted successfully\n";
    writefile(*list);
}
cout<<"\nPress any key to continue...\n";
getchar();
sleep(1);
system("cls");
break;

case 4:
    cout << "Current Fare Price = " << farecost << endl;
    cout << "Enter new fare price = ";
    cin >> farecost;
    cout << "Fare Price has been successfully changed!\n";
    writefile(*list);
    sleep(1);
    cout<<"\nPress any key to continue...\n";getchar();
    system("cls");
    break;

case 5:
    delCard();
    sleep(1);
    cout<<"\nPress any key to continue...\n";getchar();
    system("cls");
    break;

case 6:
    now->printCardDetails(C);
    sleep(3);
    cout<<"\nPress any key to continue...\n";getchar();
    getchar();
    system("cls");
    break;

case 7:
    now->changePassword();
    sleep(1);
    cout<<"\nPress any key to continue...\n"; getchar(); getchar();
    system("cls");
    break;

case 8:
    now->newUser(users);
    break;

case 9:
    cout << "ADMIN HAS BEEN LOGGED OUT SUCCESSFULLY!" << endl;
    cout << "THANK YOU FOR USING ADMIN PORTAL" << endl;
    sleep(2);
    cout<<"Press any key to continue....";
    getchar();
    system("cls");
    goto l3;
    break;

default:

```

```
if(choice>7) cout << "Invalid Input\nTRY AGAIN!\n";  
sleep(1);  
cout<<"\nPress any key to continue...\n";getchar();  
system("cls");  
break;  
}  
l2: goto l1;  
l3::  
  
}  
  
void menu(station **list) {  
  
    while (true)  
    {  
        cout <<  
"\t\t\t*****"  
*****" << endl;  
        cout << "\t\t\t\t***                NOIDA METRO SYSTEM                ***" << endl;  
        cout <<  
"\t\t\t\t*****"  
*****\n\n\n";  
        cout << "\t\t\t\t\tWelcome to Metropolis\n";  
        cout << "\t\t\t\t\tAre you a..." << endl;  
        cout << "\t\t\t\t\t1.Administrator" << endl;  
        cout << "\t\t\t\t\t2.Customer" << endl;  
        cout << "\t\t\t\t\t3.Exit the program\n";  
        cout << "\t\t\t\t\tEnter your choice\n";  
        int choice;  
        cin >> choice;  
        if (choice == 3)  
            break;  
        else if (!(choice >= 1 && choice <= 3))  
        {  
            cout << "INVALID INPUT! TRY AGAIN!" << endl;  
            sleep(1);  
            cout<<"\nPress any key to continue...\n";getchar();  
            system("cls");  
            continue;  
        }  
        else  
        {  
            if (choice == 1)  
            {  
                sleep(1);  
                system("cls");  
                admin *now = adminlogin();  
                if (now == NULL)  
                {  
                    continue;  
                }  
                adminmenu(list, now);  
            }  
            if (choice == 2)  
            {  
                sleep(1);
```

```

        system("cls");
        customer_menu(*list);
    }
}
}
}

```

```

void deletecompletelist(station *list) {
    station *p = list;
    list = p->next;
    while (p != NULL)
    {
        delete p;
        p = list;
        list = list->next;
    }
}

```

```

int check_validity(long long Input)
{

```

```

    int sumE = 0;
    int sumO = 0;
    long long credE = Input / 10;
    long long credO = Input;

```

```

    // final DESCRIBES THE VALUE WHEN THE SECOND LAST DIGIT IS SELECTED. IF IT IS MORE THAN
    9, THEN ITS DIGITS ARE ADDED AS SHOWN BELOW

```

```

    int final = 0;

```

```

    do
    {

```

```

        // remE DESCRIBES THE REMAINDER OR THE SECOND TO LAST DIGIT IN EACH LOOP (STARTING
        FROM ORIGINAL INPUT VALUE)

```

```

        int remE = credE % 10;

```

```

        // multiple INDICATES THE DOUBLING OF THE SECOND TO LAST DIGIT ACCORDING TO LUHN'S
        ALGORITHM

```

```

        int multiple = remE * 2;

```

```

        // credE IS DIVIDED BY 100 TO PROCEED TO THE SECOND TO SECOND TO LAST DIGIT AND SO ON
        credE = credE / 100;

```

```

        // AN IF-ELSE STATEMENT TO ADD UP THE DIGITS OF FACTOR IF IT IS A DOUBLE DIGIT NUMBER

```

```

        if (multiple < 10)

```

```

        {
            final = multiple;
        }

```

```

        else
        {

```

```

            // ones AND tens DESCRIBE (OBVIOUSLY) THE ONES AND TENS DIGITS OF multiple IF IT IS A
            DOUBLE DIGIT NUMBER

```

```

            int ones = multiple % 10;

```

```

            int tens = multiple / 10;

```

```

            final = ones + tens;

```

```

    }
    // THE SUM OF OPERATIONS ON EVEN VALUES FROM END OF NUMBER ACCORDING TO
ALGORITHM
    sumE += final;
    } while (credE > 0);

do
{
    int remO = credO % 10;
    credO = credO / 100;
    sumO += remO;
} while (credO > 0);

int sum = sumE + sumO;
int checksum = sum % 10;

// CALCULATING THE NUMBER OF DIGITS

int count = 0;
long long length = Input;
do
{
    length /= 10;
    count += 1;
} while (length > 0);

// CALCULATING THE FIRST DIGIT OF CREDIT CARD
long long length_1 = Input;
long long length_2 = Input;
long long first_digit;
long long second_digit;

do
{
    first_digit = length_1;
    length_1 /= 10;
} while (length_1 > 0);

do
{
    second_digit = length_2;
    length_2 /= 10;
} while (length_2 > 9);

// TO CHECK WHICH TYPE OF CREDIT CARD NUMBER WAS THE INPUT IF IT IS EVEN VALID AT ALL.
if (checksum == 0 && count == 13 && first_digit == 4)
{
    cout << "VISA\n";
    return 0;
}
else if (checksum == 0 && count == 15 && (second_digit == 34 || second_digit == 37))
{
    cout << "AMEX\n";
    return 0;
}
else if (checksum == 0 && count == 16 && first_digit == 4)
{

```

```

        cout << "VISA\n";
        return 0;
    }
    else if (checksum == 0 && count == 16 && second_digit > 50 && second_digit < 56)
    {
        cout << "MASTERCARD\n";
        return 0;
    }
    else
    // IF THE INPUT DOESN'T EVEN PASS THE CHECKSUM, THEN OBVIOUSLY IT IS AN INVALID CREDIT
    CARD NUMBER
    {
        cout << "INVALID\n";
        return -1;
    }
}

```

```

void load(){
    ifstream card;
    string s;
    card.open("creditCards.txt");
    if (!card)
    {
        card.close();
        goto next;
    }
    card>>s;
    while (card.eof() == false)
    {
        // cout << "2";
        CreditCard o;
        long long i;
        card >> i;
        float b;
        card >> b;
        o.create(s, i, b);
        C.push_back(o);
        card>>s;
    }
    card.close();
next:
    // cout << "3";
    // newCard();
    ifstream a1;
    string i;
    a1.open("admins.txt");
    if (!a1)
    {
        a1.close();
        goto next2;
    }
    a1>>s;
    a1 >> i; // 1
    while (!a1.eof())
    {
        // j++;

```

```
        admin o;  
        o.setData(s, i);  
        users.push_back(o);  
        a1>>s;  
        a1>>i;  
        // AddNode(vector,s,i);  
    }  
    // cout << j << endl;  
    a1.close();  
next2:  
    cout << "HI!" << endl;  
}
```

```
int main(){  
    load();  
    station *list = NULL;  
    readFile(&list);  
    menu(&list);  
    deletecompletelist(list);  
    return 0;  
}
```

External Files included:

admins.txt

creditCards.txt

stations.txt

ticket.txt

References:

- <https://www.geeksforgeeks.org/data-structures/linked-list/>
- <https://www.geeksforgeeks.org/searching-algorithms/>
- <https://www.geeksforgeeks.org/sorting-algorithms/>
- <https://stackoverflow.com/questions/tagged/data-structures>
- https://www.tutorialspoint.com/data_structures_algorithms/index.htm