RAILWAY MANAGEMENT SYSTEM

Data Structures And Algorithms

Project Report

Group Members

NAME	REGISTRATION NUMBER
ARPIT RATHI	16BCI0065
PIYUSH GUTGUTIA	16BCI0063
JISHU DOHARE	16BCE0984
PARMEET SINGH	16BCE0184

Under the guidance of
Prof. SHAIK NASEERA

School of Computer Science And Engineering
VIT University, Vellore

MARCH 2017

TABLE OF CONTENTS

1.) ABSTRACT

It’s really difficult for the train passengers these days specifically the train travelling passengers to find that one train which is the perfect for them to reach their destination, someone has a problem with the route, some with cost while some people are having problems with the duration of the journey. It happens to cause the railway system is not organised in India, it has many problems in its structure. It’s the real need that someone makes some specific and result worthy changes in the Indian Railway system.

What we aim to achieve is to build a shortest route finder in terms of Distance, Cost and Duration as well. A program through which the user-end i.e. the passengers can easily find the ideal train for their ideal destination by all means. The program will show the passenger all the choices he has in terms of distance, cost and duration. A program using which the reservation is made easy and convenient for the passengers. Also if there is any problem the user can easily do the cancellation and book another train which he finds is more convenient.

This program will also help the Railway Department to handle the big data of the passengers and will make it easy for the railway department to show the most optimal train for one place to another, so they can even charge more those trains and make a reasonable profit.

2.) INTRODUCTION

In our documented research we have proposed a program with its system design, through which the railway travellers can easily book the trains which are best for their journey, in this program we will be using Stack and Queue data structures to hold the data. The algorithm which we will be using is the DIJIKSTRA'S ALGORITHM, which gives the shortest path possible between two places.

Since passengers are going to use this so program, so we will also be using the File Handling to store the data and details of the passenger.

What we aim to achieve is to build a shortest route finder in terms of Distance, Cost and Duration as well. A program through which the user-end i.e. the passengers can easily find the ideal train for their ideal destination by all means. The program will show the passenger all the choices he has in terms of distance, cost and duration. A program using which the reservation is made easy and convenient for the passengers.

3.) LITERATURE SURVEY

Dijkstra's algorithm-is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. It was conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years later. The algorithm exists in many variants; Dijkstra's original variant found the shortest path between two nodes, but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a shortest path tree.
For a given source node in the graph, the algorithm finds the shortest path between that node and every other. It can also be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined.

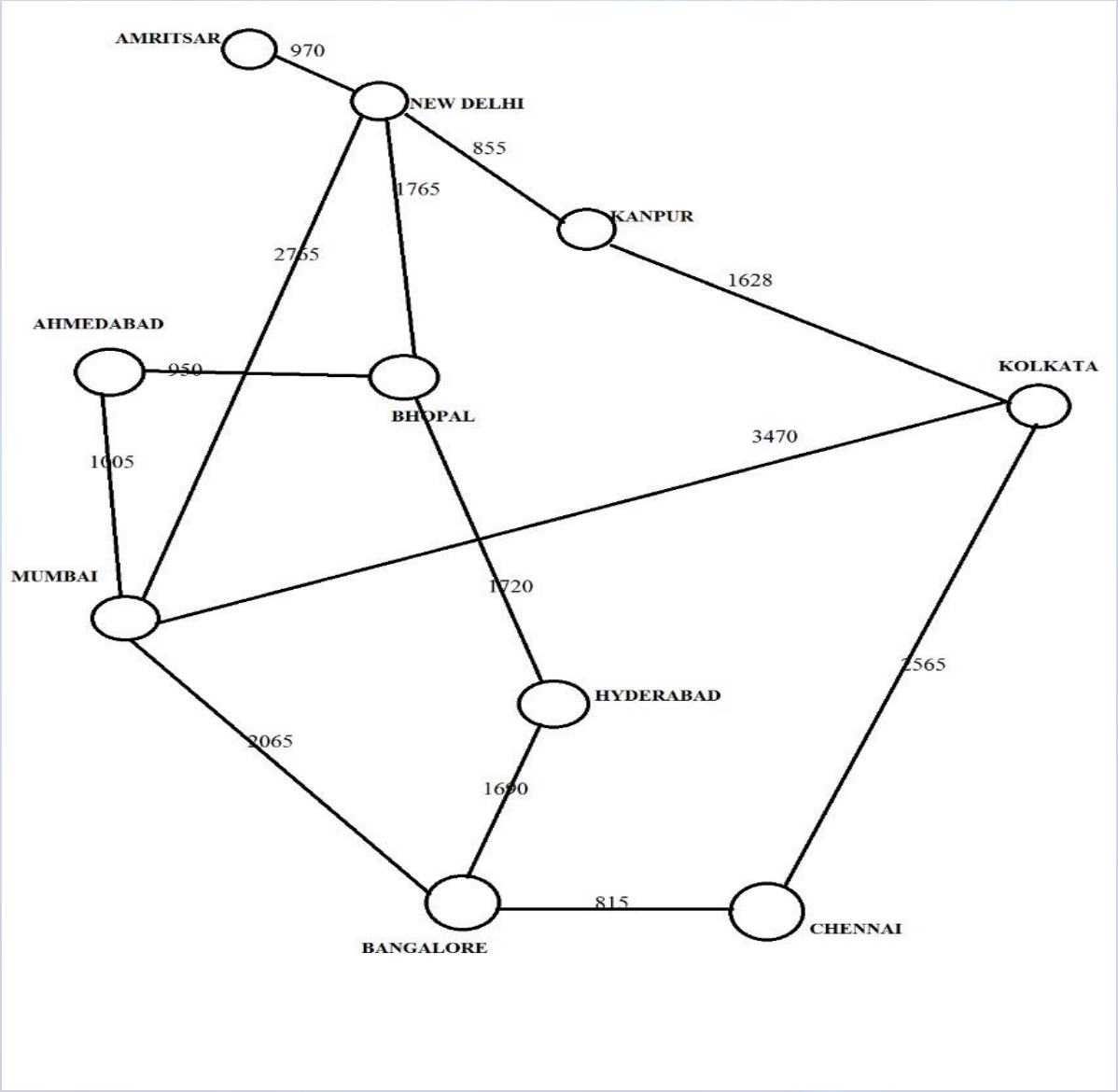
File Handling-A **file** represents a sequence of bytes on the disk where a group of related data is stored. File is created for permanent storage of data. It is a readymade structure.

In C language, we use a structure **pointer of the type** to declare a file.

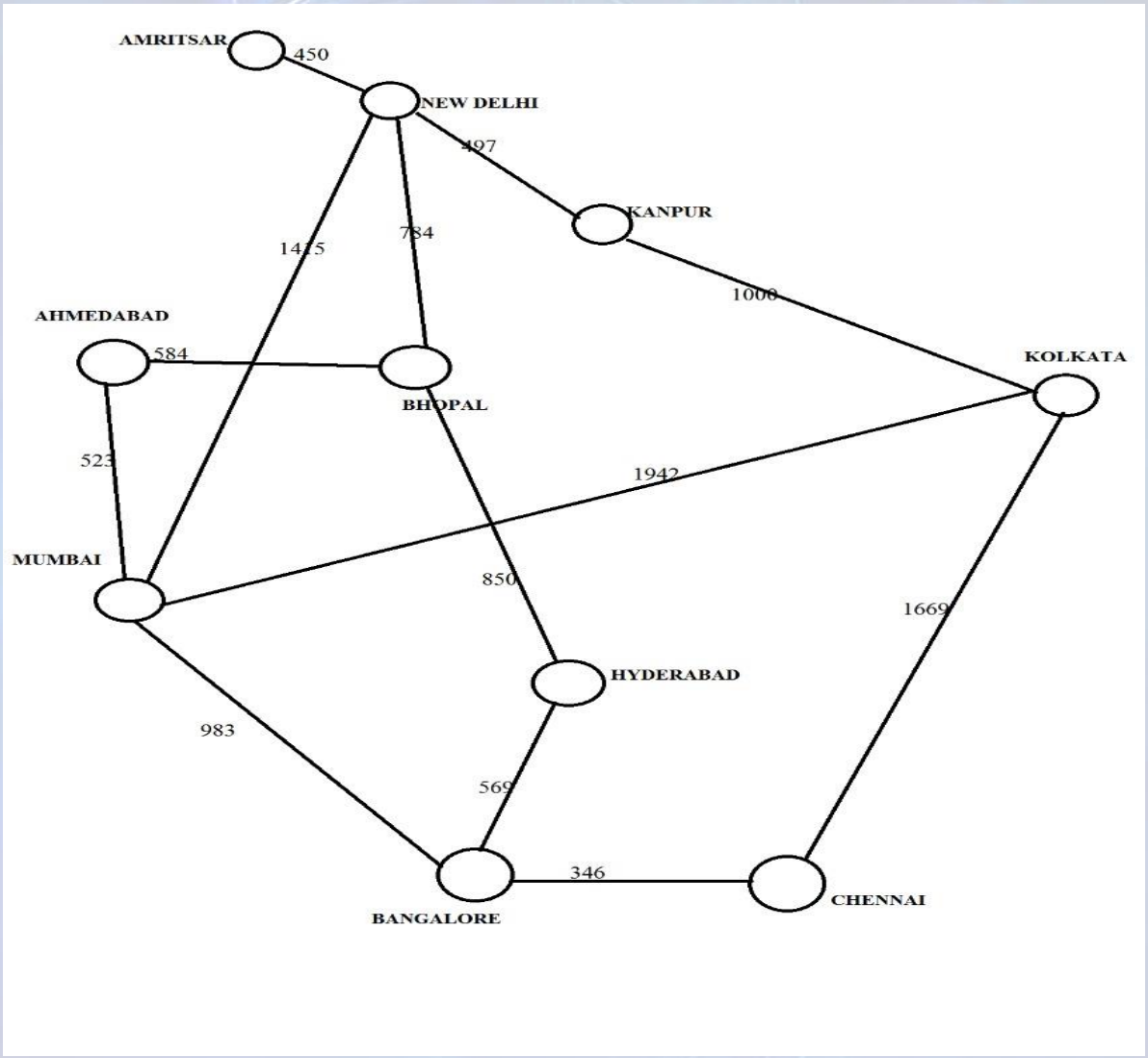
A file represents a sequence of bytes, regardless of it being a text file or a binary file. C programming language provides access on high level functions as well as low level (OS level) calls to handle file on your storage devices. This chapter will take you through the important calls for file management.

4.) Proposed Method/System

i.) By DISTANCE



ii) By COST


AMRITSAR
450
NEW DELHI
497
KANPUR
1415
784
1000
AHMEDABAD
584
KOLKATA
BHOPAL
523
1942
MUMBAI
850
1669
HYDERABAD
983
569
346
CHENNAI
BANGALORE

5.) Implementation

This code is the implementation of the ALGORITHM:

```
#include<iostream>
#include<windows.h>
#include<cstring>
#include<fstream>
#include<map>
#include<cstdio>
#include<climits>
#include<vector>
#include<algorithm>

using namespace std;

string u_c="UNDER CONSRUCTION";
```

//Function to convert string to integer equivalent

```
int convert(string s)
{
   int t=0;
   for(int i=0;i<s.length();i++)
   {
      char x=s[i];
      int x1=x-48;
      t=t*10+x1;
   }
   return t;
}
string convert1(int n)
{
   string s="";
   while(n!=0)
   {
      int a=n%10;
      s=char(a+48)+s;
      n=n/10;
   }
   return s;
}

int reduce(string name,int num)
{
   int flag=0;
   ifstream myReadFile;
   myReadFile.open("test.txt");
   char output1[10000];
```

```
    string output2;
    string arr1[100];
    map<string,string>lists;
    int k=0;
    if (myReadFile.is_open())
    {
        while (!myReadFile.eof())
        {
            myReadFile >> output1;
            myReadFile >> output2;
            int t=0;
            if(output1==name)
            {
                t=convert(output2);
                t=t-num;
                output2=convert1(t);
                flag=1;
            }
            lists[output1]=output2;
            arr1[k++]=output1;
        }
        myReadFile.close();
    }
    if(flag==1)
    {
        ofstream fout;
        fout.open("test.txt");
        for(int i=0;i<k;i++)
        {
            fout<<arr1[i]<<"\t\t"<<lists[arr1[i]]<<"\n";
        }
        fout.close();
    }
    return flag;
}
```

//Function to print a string in the middle of the screen

```
int space(string &s1)
{
    int l=s1.length();
    int pos=(int)((80-l)/2);
    for(int i=0;i<pos;i++)
    cout<<" ";
    cout<<s1<<endl;
}
```

//Function to get a line break

```
int line_break()
{
    for(int i=0;i<80;i++)
```

```
    {
        cout<<"-";
    }
    cout<<endl;
}
```

//Function to print the heading

```
int heading()
{
    system("color B");
    string s="T.R.A.I.N.S";
    string s1="THE RAILWAY APPLICATION WITH INTEGRATED NEW SELF-HELP";
    line_break();
    space(s);
    cout<<"\n";
    space(s1);
    cout<<"\n";
    line_break();
}
```

//Main reservation function

```
struct ticket
{
    string passenger_name;
    int passenger_age;
};
int train_schedules()
{
    system("cls");
    heading();
    ifstream myReadFile;
    myReadFile.open("test.txt");
    char output1[10000];
    char output2[10000];
    if (myReadFile.is_open())
    {
        while (!myReadFile.eof())
        {
            myReadFile >> output1;
            myReadFile >> output2;
            cout<<output1<<"\t\t"<<output2<<endl;
        }
        myReadFile.close();
    }
    system("pause");
    system("cls");
    heading();
}
int check_avalibility()
{
```

```
system("cls");
heading();
string name;
int n;
string tick_req;
cout<<"ENTER TRAIN NAME"<<endl;
cin>>name;
cout<<"ENTER NUMBER OF TICKETS EQUIRED"<<endl;
cin>>tick_req;
ifstream myReadFile;
myReadFile.open("test.txt");
string output1;
string output2;
map<string,string> avalibility;
string train_list[100];
int k=0;
int flag=0;
if (myReadFile.is_open())
{
   while (!myReadFile.eof())
   {
      myReadFile >> output1;
      myReadFile >> output2;
      avalibility[output1]=output2;
      train_list[k++]=output1;
   }
}
int av=0;
   for(int i=0;i<k;i++)
   {
      if(name==train_list[i])
      {
         av=1;
         break;
      }
   }
   if (av==1)
   {
   if(convert(avalibility[name])>=convert(tick_req))
   {
      cout<<"TICKETS AVALIABLE"<<endl;
      flag=1;
   }
   else
   {
      cout<<"TICKETS NOT AVALIABLE"<<endl;
      flag=0;
   }
   }
   else
```

```
        {
            cout<<"Train not found"<<endl;
        myReadFile.close();
        }
    system("pause");
    system("cls");
    heading();
    return flag;
}
int book_ticket()
{
    system("cls");
    heading();
    string name;
    string tick_req;
    cout<<"ENTER THE TRAIN NAME"<<endl;
    cin>>name;
    cout<<"ENTER THE NUMBER OF TICKETS REQUIRED"<<endl;
    cin>>tick_req;
     //CHANGE HERE
    int n=convert(tick_req);
    //DECREASE THE NUMBER OF TICKETS HERE
    int t=reduce(name,n);
    if(t==1)
    {
         struct ticket book[n];
        cout<<"ENTER THE PASSENGERS NAMES AND AGE"<<endl;
        for(int i=0;i<n;i++)
        {
            cin>>book[i].passenger_name;
            cin>>book[i].passenger_age;
        }
        ofstream myfile;
        string pnr="";
        for (int i = 0; i < n; ++i)
        {
            pnr += book[i].passenger_name[0];
        }
        myfile.open (pnr+".txt");
        myfile <<"TRAIN_NAME :"<<name<<endl;
        myfile <<"NAME\t\t:AGE"<<endl;
        for(int i=0;i<n;i++)
        {
            myfile << book[i].passenger_name<<"\t\t:"<<book[i].passenger_age<<endl;
        }
        myfile.close();
        cout<<"YOUR PNR NUMBER IS : "<<pnr<<endl;
        cout<<"TICKET SUCESSFULLY BOOKED"<<endl;
    }
```

```
    else
    {
        cout<<"NO TRAIN FOUND"<<endl;
    }
    system("pause");
    system("cls");
    heading();
    return 0;
}
int check_PNR()
{
    system("cls");
    heading();
    string pnr="";
    cout<<"ENTER PNR NUMBER"<<endl;
    cin>>pnr;
    ifstream myReadFile;
    myReadFile.open(pnr+".txt");
    char output1[10000];
    char output2[10000];
    if (myReadFile.is_open())
    {
        while (!myReadFile.eof())
        {
            myReadFile >> output1;
            myReadFile >> output2;
            cout<<output1<<"\t"<<output2<<endl;
        }
        myReadFile.close();
    }
    system("pause");
    system("cls");
    heading();
}
int reservation()
{
    system("cls");
    heading();
    //space(u_c);
    int ch=10;
    while(ch)
    {
        cout<<"1.Train Schedules\n2.Check Avalibility\n3.Book Ticket\n4.Check PNR
Status\n0.Exit"<<endl;
        cin>>ch;
        if(ch==1)
            train_schedules();
        else if(ch==2)
            check_avalibility();
        else if(ch==3)
```

```
            book_ticket();
        else if(ch==4)
            check_PNR();
        else if(ch==0)
            cout<<"\nTHANKYOU FOR USING OUR APPLICATION"<<endl;
        else
        {
            cout<<"\n\nINVALID INPUT"<<endl;
            Sleep(2000);
            system("cls");
            heading();
        }
    }
    system("pause");
    system("cls");
    heading();
    return 0;
}
```

//Main cancellation function

```
int cancellation()
{
    system("cls");
    heading();
    //space(u_c);
    char pnr[100];
    cout<<"ENTER THE PNR NUMBER OF THE TICKET YOU WANT TO
CANCEL"<<endl;
    cin>>pnr;
    const int result =remove(pnr);
    system("pause");
    system("cls");
    heading();
    return 0;
}
```

//Get Route function

```
#define V 10

int minDistance(int dist[], bool sptSet[])
{
    int min = INT_MAX, min_index;

    for (int v = 0; v < V; v++)
        if (sptSet[v] == false && dist[v] <= min)
            min = dist[v], min_index = v;

    return min_index;
}
void printPath(int parent[], int j)
```

```
{
   map<int,string>cities_1;
   cities_1[0]="amritsar";
   cities_1[1]="delhi";
   cities_1[2]="kanpur";
   cities_1[3]="kolkata";
   cities_1[4]="ahemdabad";
   cities_1[5]="mumbai";
   cities_1[6]="bhopal";
   cities_1[7]="hyderabad";
   cities_1[8]="banglore";
   cities_1[9]="chennai";
   if (parent[j]==-1)
      return;
                printPath(parent, parent[j]);

   cout<<" "<<cities_1[j];
}
int printSolution(int dist[], int n, int parent[],int src)
{
   map<int,string>cities_1;
   cities_1[0]="amritsar";
   cities_1[1]="delhi";
   cities_1[2]="kanpur";
   cities_1[3]="kolkata";
   cities_1[4]="ahemdabad";
   cities_1[5]="mumbai";
   cities_1[6]="bhopal";
   cities_1[7]="hyderabad";
   cities_1[8]="banglore";
   cities_1[9]="chennai";
   cout<<"Distance\tPath";
   for (int i = 0; i < V; i++)
   {
                cout<<"\n"<<dist[i]<<"\t\t"<<cities_1[src]<<" ";
      printPath(parent, i);
   }
}
void dijkstra(int graph[10][10], int src)
{
   int dist[V];
         bool sptSet[V];
         int parent[V];
         for (int i = 0; i < V; i++)
   {
      parent[src] = -1;
      dist[i] = INT_MAX;
      sptSet[i] = false;
   }
         dist[src] = 0;
```

```
   for (int count = 0; count < V-1; count++)
   {
                  int u = minDistance(dist, sptSet);
                  sptSet[u] = true;
      for (int v = 0; v < V; v++)
         if (!sptSet[v] && graph[u][v] &&
            dist[u] + graph[u][v] < dist[v])
         {
            parent[v]  = u;
            dist[v] = dist[u] + graph[u][v];
         }
   }
   printSolution(dist, V, parent,src);
}
int get_route()
{
   system("cls");
   heading();
   int graph1[10][10]={{0,450,0,0,0,0,0,0,0,0},
            {450,0,497,0,0,1415,784,0,0,0},
            {0,497,0,1000,0,0,0,0,0,0},
            {0,0,1000,0,0,1942,0,0,0,1669},
            {0,0,0,0,0,523,584,0,0,0},
            {0,1415,0,1942,523,0,0,0,983,0},
            {0,784,0,0,584,0,0,850,0,0},
            {0,0,0,0,0,0,850,0,0,0},
            {0,0,0,0,0,983,0,0,0,346},
            {0,0,0,1669,0,0,0,0,346,0}};        // distance

int graph[10][10]={{0,970,0,0,0,0,0,0,0,0},
                   {970,0,855,0,0,2765,1765,0,0,0},
                   {0,855,0,1628,0,0,0,0,0,0},
                   {0,0,1628,0,0,3470,0,0,0,2565},
                   {0,0,0,0,0,1005,950,0,0,0},
                   {0,2765,0,3470,1005,0,0,0,2065,0},
                   {0,1765,0,0,950,0,0,1720,0,0},
                   {0,0,0,0,0,0,1720,0,1690,0},
                   {0,0,0,0,0,2065,0,1690,0,815},
                   {0,0,0,2565,0,0,0,0,815,0}};        // cost
map<string,int>city;
city["amritsar"]=0;
city["delhi"]=1;
city["kanpur"]=2;
city["kolkata"]=3;
city["ahemdabad"]=4;
city["mumbai"]=5;
city["bhopal"]=6;
city["hyderabad"]=7;
city["banglore"]=8;
```

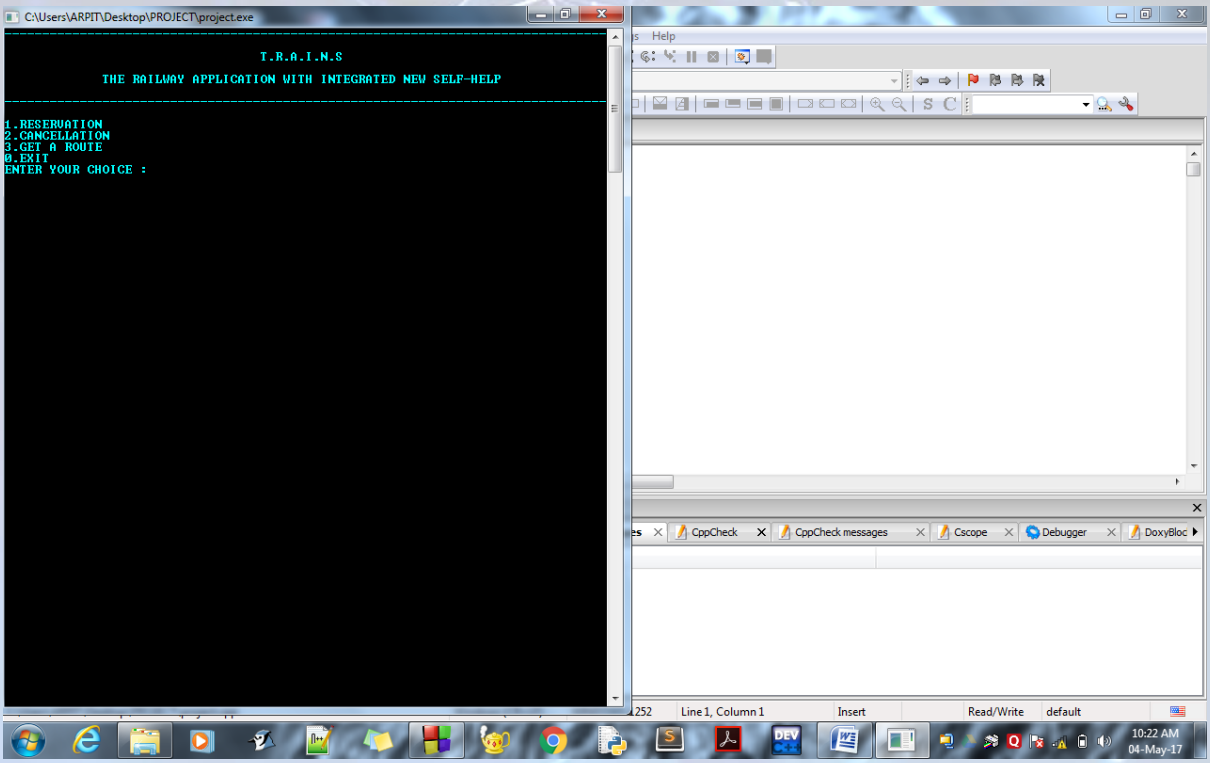
```
city["chennai"]=9;

int ch=0;
cout<<"1.BY DISTANCE\n2.BY MONEY"<<endl;
cin>>ch;
cout<<"Enter the source node\n";

int src=0;
string city_name="";
cin>>city_name;
cout<<"\n\n\n\n";
src=city[city_name];
if(ch==1)
   dijkstra(graph, src);
else
   dijkstra(graph1, src);
   cout<<endl;
   system("pause");
   system("cls");
   heading();
   return 0;
}
```

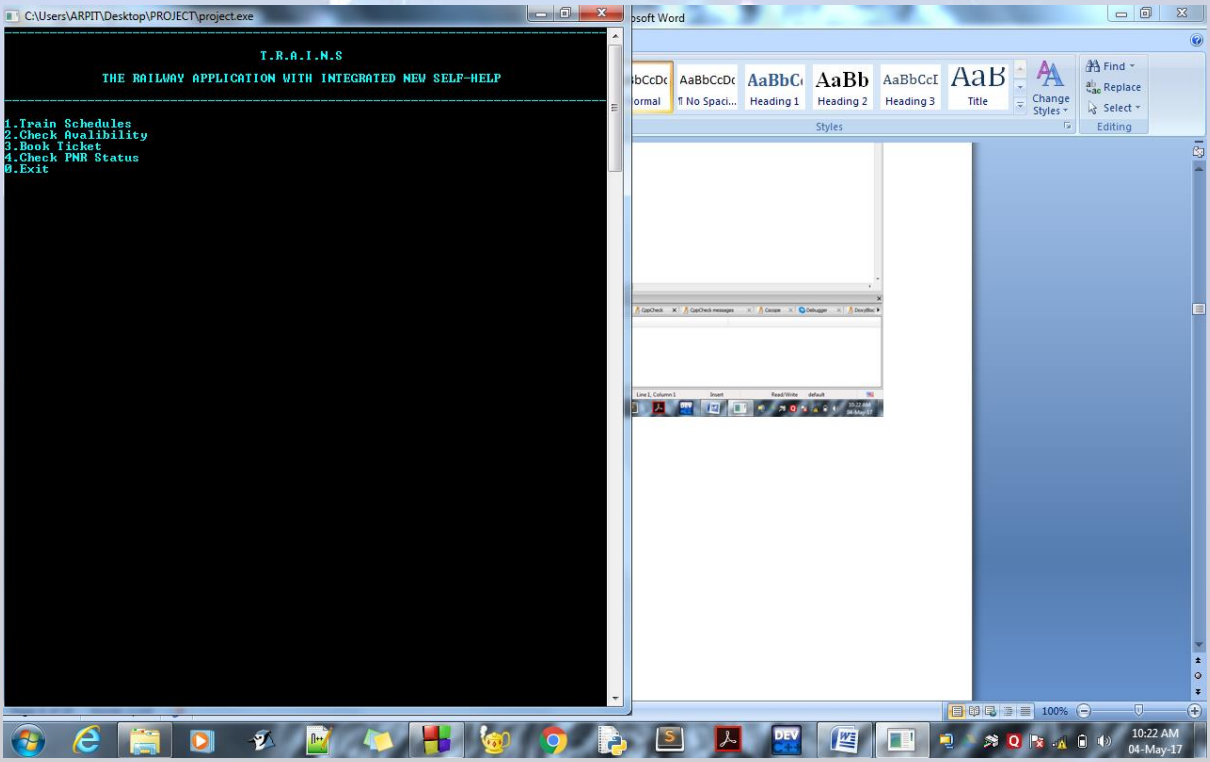
//MAIN FUNCTION

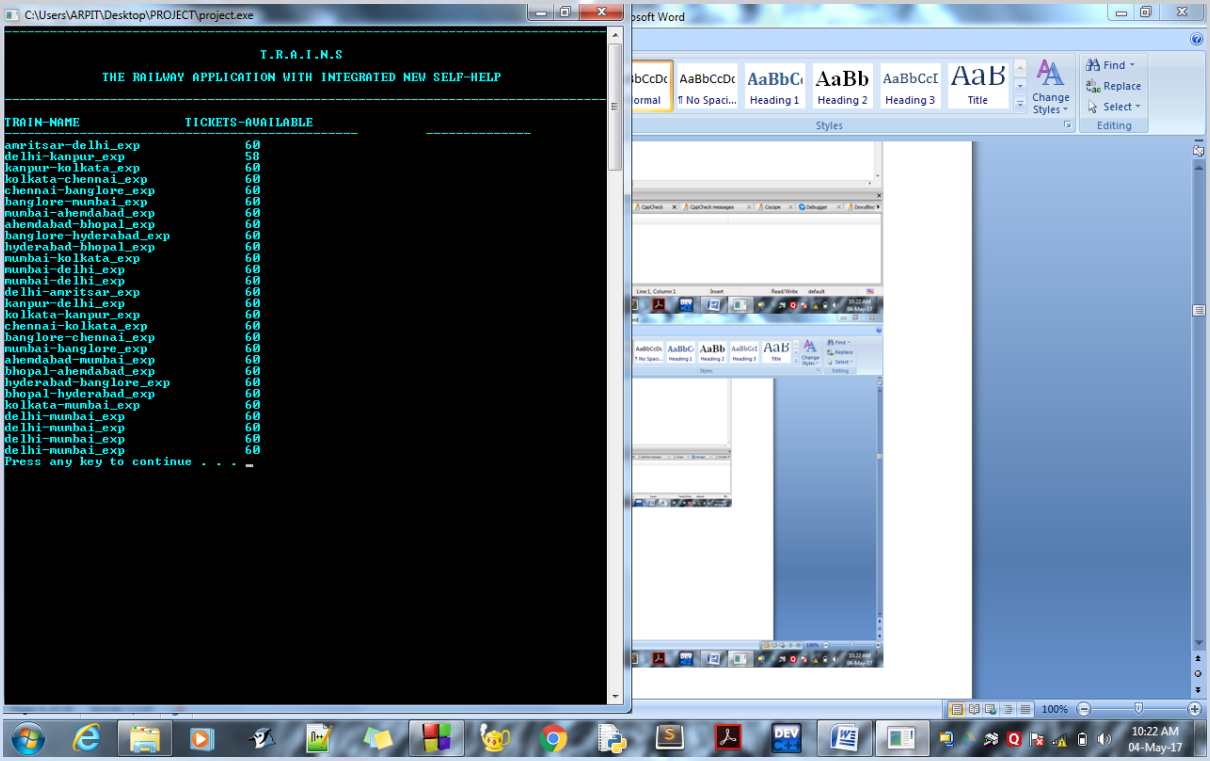
```
int main()
{
   heading();
   int ch=10;
   while(ch)
   {
      cout<<"1.RESERVATION\n2.CANCELLATION\n3.GET A
ROUTE\n0.EXIT\nENTER YOUR CHOICE : ";
      cin>>ch;
      if(ch==1)
         reservation();
      else if(ch==2)
         cancellation();
      else if(ch==3)
         get_route();
      else if(ch==0)
         cout<<"\nTHANKYOU FOR USING OUR APPLICATION"<<endl;
      else
      {
         cout<<"\n\nINVALID INPUT"<<endl;
         Sleep(2000);
```

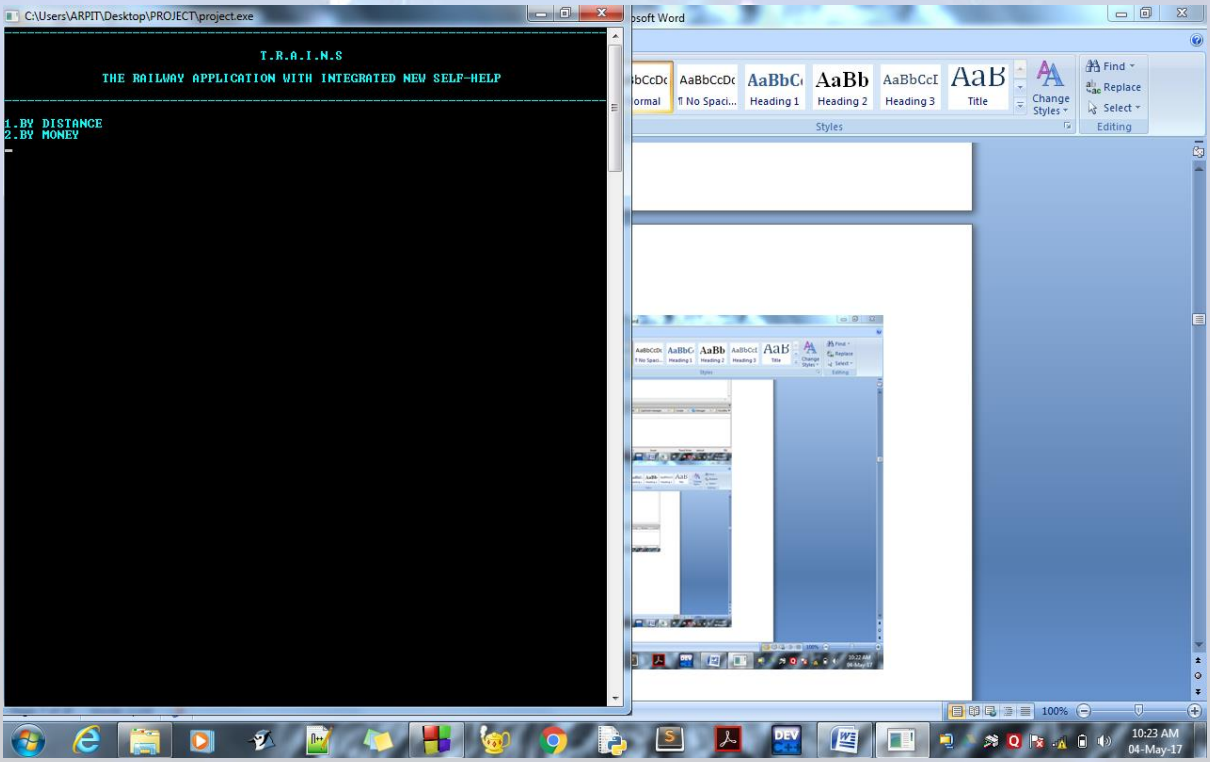
```
            system("cls");
            heading();
        }
    }
    system("pause");
}
```

6.) Results and Discussions






C:\Users\ARPIT\Desktop\PROJECT\project.exe
T.R.A.I.N.S
THE RAILWAY APPLICATION WITH INTEGRATED NEW SELF-HELP
TRAIN-NAME TICKETS-AVAILABLE
amritsar-delhi_exp 60
delhi-kanpur_exp 58
kanpur-kolkata_exp 60
kolkata-chennai_exp 60
chennai-banglore_exp 60
banglore-mumbai_exp 60
mumbai-ahemdabad_exp 60
ahemdabad-bhopal_exp 60
banglore-hyderabad_exp 60
hyderabad-bhopal_exp 60
mumbai-kolkata_exp 60
mumbai-delhi_exp 60
mumbai-delhi_exp 60
delhi-amritsar_exp 60
kanpur-delhi_exp 60
kolkata-kanpur_exp 60
chennai-kolkata_exp 60
banglore-chennai_exp 60
mumbai-banglore_exp 60
ahemdabad-mumbai_exp 60
bhopal-ahemdabad_exp 60
hyderabad-banglore_exp 60
bhopal-hyderabad_exp 60
kolkata-mumbai_exp 60
delhi-mumbai_exp 60
delhi-mumbai_exp 60
delhi-mumbai_exp 60
delhi-mumbai_exp 60
Press any key to continue . . .
Heading 1
Heading 2
Heading 3
Title
Change Styles
Find
Replace
Select
Styles
Editing
100%
10:22 AM
04-May-17


C:\Users\ARPIT\Desktop\PROJECT\project.exe
T.R.A.I.N.S
THE RAILWAY APPLICATION WITH INTEGRATED NEW SELF-HELP
1.BY DISTANCE
2.BY MONEY
Heading 1
Heading 2
Heading 3
Title
Change Styles
Find
Replace
Select
Styles
Editing
100%
10:23 AM
04-May-17

7.) Conclusion

We are successful in completing our programme by djikstra and file handling.

Our program is running good.

In this project we are able to book a ticket, check availability of tickets, cancel a ticket and find the shortest path based on the distance and money.

The Program is a result of the hard work and perseverance of
Our team mates. At last we would like to conclude this by
Saying that the DSA course has really helped in opening us to
And giving the exposure to a new realm of Algorithms.
After this course we can see the difference, now as a CSE
Student we are more Interested in the Algorithmic part of the
Machines and according to which Algorithm the machines
Works like the Human Less Drone, Quad Copter, and
Other computing machines.

Algorithms like Tower of Hanoi, Dijkstra's Shortest Path Algorithm, Knap Sack Greedy Algorithm and not just algorithms but also a number of sorting techniques like Insertion Sort, Merge Sort, Quick Sort etc.

We cannot forget the Data Structures like Queue's, Linked Lists, Hash Maps, Stack, Heap and Graphs.

Many of these we have also used in our program of the shortest route.

8.) References

http://www.studytonight.com/c/file-input-output.php

https://www.tutorialspoint.com/cprogramming/c_file_io.htm

http://www.sanfoundry.com/c-programming-examples-file-handling/

http://www.geeksforgeeks.org

https://stackoverflow.com/