DATA STRUCTURES AND ALGORITHM

SUBJECT CODE - 15B17CI578

REPORT ON – Commute App

SUBMITTED BY:

NAME	ENROLLMENT NO.
SRIJAN BHARDWAJ	19102118
ANMOL ARORA	19102110
NIKHIL BHATT	19102091
TUSHAR SINGH TOMAR	19102093

INTRODUCTION

This project is used to find the shortest and most economical route the passenger will take to reach the destination. It can also be used to calculate the fare and time depending on the distance travelled by the passenger. This is implemented using graphs and Djiktra's Algorithm which is used to find the shortest path between nodes in a graph which in this case represents a network of metro stations.

BACKGROUND

For some of us living in big cities, imagining our lives without the Metro would be difficult. The method of transport allows us to travel major distances much faster while simultaneously helping us skip the road traffic and greater exposure to pollution. When compared to private automobiles, this mode of transportation is also quite affordable. Based on all these reasons, the popularity of the Metro rail has gone up significantly.

Hence, we used this idea to create an application to calculate the total fare and give us the shortest path for easy travel.

APPLICATIONS

1. **Calculation of the Fare -** This application will give a fair idea about the fare of the metro. This project is limited to Delhi and has the price according to Delhi Public Commute.
   This will give a predefined idea about the money needed to go from one metro station to another.

2. **Comparing the fastest path and shortest path-** Delhi Public Commute has a vast network of metro stations and connects different areas of Delhi via different lines such as yellow line, pink line, violet line, red line etc. For a newly Delhi tourist it is hard to find the correct route to be taken so that less time is required to reach the destination. Through this application it will be easy to calculate the shortest distance between two points.
   This will definitely save time.
3. **Path length defining-** Description of path followed in difference cases and choices including an intermediate stop to make the application highly flexible and customized

REQUIREMENT SPECIFICATION

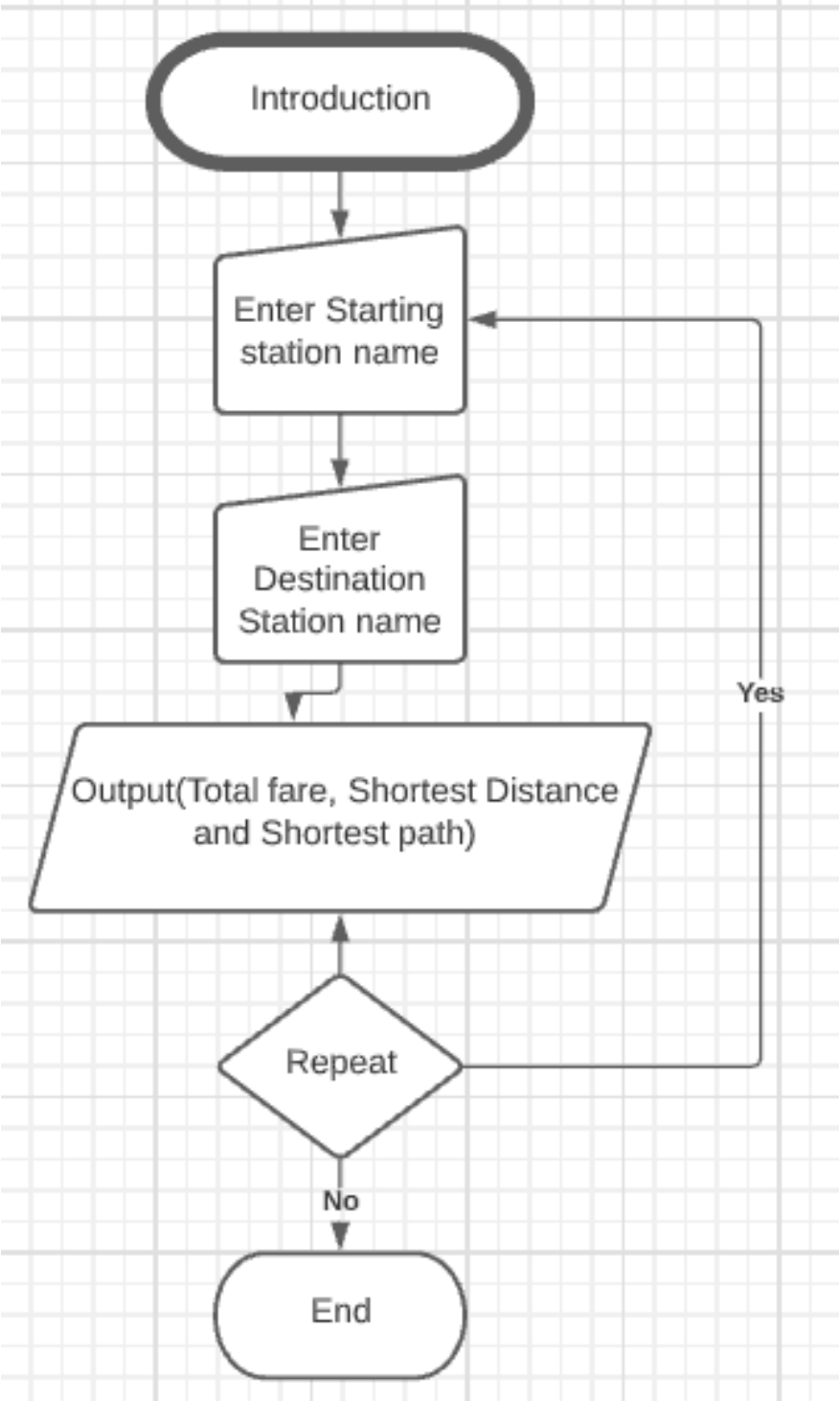
Hardware/ Software : C++ environment

Graphs : Graph is a non-linear data structure consisting of nodes and edges. The nodes are sometimes also referred to as vertices and the edges are lines or arcs that connect any two nodes in the graph. More formally a Graph can be defined as,
A structure that consists of a finite set of vertices(or nodes) and a set of Edges which connect a pair of nodes.

Dijkstra's Algorithm : Dijkstra's original algorithm found the shortest path between two given 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.

FLOW CHART


Introduction
Enter Starting station name
Enter Destination Station name
Output(Total fare, Shortest Distance and Shortest path)
Repeat
Yes
No
End

IMPLEMENTATION

Code.cpp

```
#include <iostream>
#include <limits.h>
#include <string>
#include <math.h>
#include <cstdlib>
#include <vector>
#include <fstream>
#include <time.h>
#include <windows.h>
using namespace std;
const int V = 248;
struct station_code
{
    string name;
    int code;
    string color;
};
struct station_code station[V];
float graph[V][V];
float graph_t[V][V];

struct node1
{
    vector<float> p;
} P[V];

string makecapital(string str)
{
    for (int i = 0; i < str.length(); i++)
    {
        if (str[i] > 96 && str[i] < 123)
            str[i] = str[i] - 32;
    }
    return str;
```

```
}

void drawbox(int, int, int, int, int);
void secondWindow();
void clrscreen();
void delay(unsigned int ms);
void gotoxy(int x, int y);
int timetaken(float dist);
int money(float dist);
void Details(int t);
void Path(float dist, int e, int st, int inter);
int minDistance(float dist[], bool sptSet[]);
int minTime(float dist[], bool sptSet[]);
void dijkstra(float graph[V][V], int src, int targ);
void dijkstra_time(float graph[V][V], int src, int targ);
void printSolution(float dist[], int n, int src, int temp);
void take_input();
void logo(int x, int y);
void secondWindow();
void UI();

COORD coord;
void clrscreen()
{
    system("cls");
    drawbox(1, 0, 117, 29, 0);
}

void delay(unsigned int ms)
{
    clock_t goal = ms + clock();
    while (goal > clock())
        ;
}

void gotoxy(int x, int y)
{
    coord.X = x;
    coord.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
```

```
}

void drawbox(int x1, int y1, int x2, int y2, int d)
{
    gotoxy(x1, y1);
    printf("+");
    delay(d);
    gotoxy(x2, y2);
    printf("+");
    delay(d);
    int i;
    for (i = 1; i < x2 - x1; i++)
    {
        gotoxy(x1 + i, y1);
        printf("-");
        printf("\a");
        delay(d);
        gotoxy(x2 - i, y2);
        printf("-");
        delay(d);
    }
    gotoxy(x1, y2);
    printf("+");
    gotoxy(x2, y1);
    printf("+");
    int j;
    for (j = 1; j < y2 - y1; j++)
    {
        gotoxy(x2, y1 + j);
        printf("|");
        delay(d);
        gotoxy(x1, y2 - j);
        printf("|");
        delay(d);
    }
}

int timetaken(float dist)
{
    float speed = 0.55;
```

```
    return ceil(dist / speed);
}

int money(float dist)
{
    if (dist <= 2)
        return 10;
    else if (dist > 2 && dist <= 5) // edit
        return 20;
    else if (dist > 5 && dist <= 12)
        return 30;
    else if (dist > 12 && dist <= 21)
        return 40;
    else if (dist > 21 && dist <= 32)
        return 50;
    else
        return 60;
}

void Details(int t)
{
    // cout<<"\nthe benchmark location present near "<<t<<"is:
"<<details[t].near<<endl;
}

void Path(float d, int e, int st, int inter)
{
    int t = e, s;
    static float dist = 0;
    dist += d;

    gotoxy(16, 11);
    cout << "THE SHORTEST PATH IS : ";

    static int pos_x = 12;
    static int pos_y = 15;
    vector<int> path;
    path.push_back(t);
    while (t != st)
    {
```

```
        s = P[t].p.size();
        t = P[t].p[s - 1];
        path.push_back(t);
    }
    vector<int>::iterator i = path.end();
    string str;
    string color;
    if (!inter)
    {
        gotoxy(44, 13);
        cout << "****** LOADING ******";
        delay(2000);
        gotoxy(42, 13);
        cout << "****** ROUTE FOUND ******";
    }
    vector<int>::iterator i2 = path.end();
    i2--;
    int n_of_stations = 0;
    while (i != path.begin())
    {
        i--;
        color = station[*i].color;
        if (color == "BLUE")
            system("color 09");
        else if (color == "YELLOW")
            system("color 06");
        else if (color == "PINK")
            system("color 0D");
        else if (color == "RED")
            system("color 04");
        else if (color == "MAGENTA")
            system("color 05");
        else if (color == "VOILET")
            system("color 01");
        else if (color == "GREEN")
            system("color 02");
        else if (color == "AQUA")
            system("color 03");
        else if (color == "ORANGE")
            system("color 0C");
```

```
        if (i != i2)
        {
            if (station[*(i2)].color != station[*(i)].color)
            {
                pos_x = 8;
                pos_y++;
                gotoxy(pos_x, pos_y);
                cout << "{change from " << station[*i2].color << " to " <<
station[*i].color << "}";
                pos_x = 12;
                pos_y++;
                gotoxy(pos_x, pos_y);
            }
            i2--;
        }
        str = station[*i].name;
        if (pos_x + 5 + str.size() < 106)
        {
            gotoxy(pos_x, pos_y);
            cout << " ->> " << str;
            pos_x = pos_x + str.size() + 5;
        }
        else
        {
            pos_x = 12;
            pos_y++;
            gotoxy(pos_x, pos_y);
            cout << " ->> " << str;
            pos_x = pos_x + str.size() + 5;
        }
        n_of_stations++;
        delay(700);
    }
    if (!inter)
    {
        gotoxy(12, ++pos_y);
        cout << "INTERMEDIATE Point";
        pos_y++;
        pos_x = 12;
```

```
    }

    if (inter)
    {
        delay(1000);
        gotoxy(72, 11);
        cout << "PATH LENGTH IS :";
        gotoxy(97, 11);
        cout << d << " KM";
        delay(1000);
        gotoxy(16, 10);
        cout << "AVERAGE TIME : ";
        gotoxy(39, 10);
        cout << timetaken(d) << " MIN";
        delay(1000);
        gotoxy(72, 10);
        cout << "AVERAGE FARE : ";
        gotoxy(94, 10);
        cout << " Rs. " << money(d);
        delay(1000);
        gotoxy(72, 12);
        cout << "NO OF STOPS : ";
        gotoxy(92, 12);
        cout << n_of_stations - 1;
    }
    else
    {
        delay(1000);
        gotoxy(72, 11);
        cout << "PATH LENGTH IS :";
        gotoxy(89, 11);
        cout << d << " KM,";
        delay(1000);
        gotoxy(16, 10);
        cout << "AVERAGE TIME : ";
        gotoxy(32, 10);
        cout << timetaken(d) << " MIN,";
        delay(1000);
        gotoxy(72, 10);
        cout << "AVERAGE FARE : Rs. ";
```

```
        gotoxy(91, 10);
        cout << money(d) << ",";
        delay(1000);
        gotoxy(72, 12);
        cout << "NO OF STOPS : ";
        gotoxy(89, 12);
        cout << n_of_stations - 1 << ",";
    }
    // if (inter)
    // {
    //     delay(2500);
    //     gotoxy(44, 9);
    //     cout << "WANT TO SEARCH AGAIN ?  ";
    //     string choice;
    //     pos_x = 12;
    //     pos_y = 15;
    //     // dekhna padega
    //     // gotoxy(68,9)
    //     cin >> choice;
    //     choice = makecapital(choice);
    //     if (choice == "Y" || choice == "YES")
    //         secondWindow();
    //     gotoxy(5, 30);
    //     char ch;
    //     scanf("%c", &ch);
    //     cout << endl;
    // }
    return;
}

// min path according to time start

void Pathmin(float d, int e, int st, int inter)
{
    int t = e, s;
    static float dist = 0;
    dist += d;
    // clrscreen();

    gotoxy(16, 11);
```

```
    cout << "THE SHORTEST PATH IS : ";

    static int pos_x = 12;
    static int pos_y = 15;
    vector<int> path;
    path.push_back(t);
    while (t != st)
    {
        s = P[t].p.size();
        t = P[t].p[s - 1];
        path.push_back(t);
    }
    vector<int>::iterator i = path.end();
    string str;
    string color;
    if (!inter)
    {
        gotoxy(44, 13);
        cout << "****** LOADING ******";
        delay(2000);
        gotoxy(89, 13);
        cout << "****** ROUTE FOUND ******";
    }
    vector<int>::iterator i2 = path.end();
    i2--;
    int n_of_stations = 0;
    while (i != path.begin())
    {
        i--;
        color = station[*i].color;
        if (color == "BLUE")
            system("color 09");
        else if (color == "YELLOW")
            system("color 06");
        else if (color == "PINK")
            system("color 0D");
        else if (color == "RED")
            system("color 04");
        else if (color == "MAGENTA")
            system("color 05");
```

```
        else if (color == "VOILET")
            system("color 01");
        else if (color == "GREEN")
            system("color 02");
        else if (color == "AQUA")
            system("color 03");
        else if (color == "ORANGE")
            system("color 0C");

        if (i != i2)
        {
            if (station[*(i2)].color != station[*(i)].color)
            {
                pos_x = 8;
                pos_y++;
                gotoxy(pos_x, pos_y);
                cout << "{change from " << station[*i2].color << " to " <<
station[*i].color << "}";
                pos_x = 12;
                pos_y++;
                gotoxy(pos_x, pos_y);
            }
            i2--;
        }
        str = station[*i].name;
        if (pos_x + 5 + str.size() < 106)
        {
            gotoxy(pos_x, pos_y);
            cout << " ->> " << str;
            pos_x = pos_x + str.size() + 5;
        }
        else
        {
            pos_x = 12;
            pos_y++;
            gotoxy(pos_x, pos_y);
            cout << " ->> " << str;
            pos_x = pos_x + str.size() + 5;
        }
        n_of_stations++;
```

```
        delay(700);
    }
    if (!inter)
    {
        gotoxy(12, ++pos_y);
        cout << "INTERMEDIATE POINT";
        pos_y++;
        pos_x = 12;
    }

    if (inter)
    {
        delay(1000);
        gotoxy(72, 11);
        cout << "PATH LENGTH IS :";
        gotoxy(97, 11);
        cout << d << " KM";
        delay(1000);
        gotoxy(16, 10);
        cout << "AVERAGE TIME : ";
        gotoxy(39, 10);
        cout << timetaken(d) << " MIN";
        delay(1000);
        gotoxy(72, 10);
        cout << "AVERAGE FARE : ";
        gotoxy(94, 10);
        cout << " Rs. " << money(d);
        delay(1000);
        gotoxy(72, 12);
        cout << "NO OF STOPS : ";
        gotoxy(92, 12);
        cout << n_of_stations - 1;
    }
    else
    {
        delay(1000);
        gotoxy(72, 11);
        cout << "PATH LENGTH IS :";
        gotoxy(89, 11);
        cout << d << " KM,";
```

```
        delay(1000);
        gotoxy(16, 10);
        cout << "AVERAGE TIME : ";
        gotoxy(32, 10);
        cout << timetaken(d) << " MIN,";
        delay(1000);
        gotoxy(72, 10);
        cout << "AVERAGE FARE : Rs. ";
        gotoxy(91, 10);
        cout << money(d) << ",";
        delay(1000);
        gotoxy(72, 12);
        cout << "NO OF POINTS : ";
        gotoxy(89, 12);
        cout << n_of_stations - 1 << ",";

        delay(3000);
        /////// print min here

    }
    if (inter)
    {
        delay(2500);
        gotoxy(44, 9);
        cout << "WANT TO SEARCH AGAIN ?  ";
        string choice;
        pos_x = 12;
        pos_y = 15;
        // dekhna padega
        gotoxy(68, 9);
        cin >> choice;
        choice = makecapital(choice);
        if (choice == "Y" || choice == "YES")
            secondWindow();
        gotoxy(5, 30);
        char ch;
        scanf("%c", &ch);
        cout << endl;
    }
    return;
```

```
}

// min path according to time end

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

int minTime(float time[], bool sptSet[])
{
    float min = INT_MAX;
    int min_index;

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

    return min_index;
}

void printSolution(float dist[], int n, int src, int temp, int inter)
{
    Details(temp);
    Path(dist[temp], temp, src, inter);
}

void printminSolution(float dist[], int n, int src, int temp, int inter)
{
    Details(temp);
    Pathmin(dist[temp], temp, src, inter);
}
```

```
void dijkstra(float graph[V][V], int src, int targ, int inter)
{
    float dist[V];
    bool sptSet[V];
    for (int i = 0; i < V; i++)
        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] != INT_MAX && dist[u]
+ graph[u][v] < dist[v])
            {
                dist[v] = dist[u] + graph[u][v];
                P[v].p.push_back(u);
            }
    }
    printSolution(dist, V, src, targ, inter);
}

void dijkstra__time(float graph_t[V][V], int src, int targ, int inter)
{
    float time[V];
    bool sptSet[V];
    for (int i = 0; i < V; i++)
        time[i] = INT_MAX, sptSet[i] = false;
    time[src] = 0;
    for (int count = 0; count < V - 1; count++)
    {
        int u = minTime(time, sptSet);
        sptSet[u] = true;
        for (int v = 0; v < V; v++)
            if (!sptSet[v] && graph_t[u][v] && time[u] != INT_MAX &&
time[u] + graph_t[u][v] < time[v])
            {
                time[v] = time[u] + graph_t[u][v];
                P[v].p.push_back(u);
            }
```

```
    }
    printminSolution(time, V, src, targ, inter);
}

int lcs(string X, string Y)
{
    int m = X.length();
    int n = Y.length();
    int L[m + 1][n + 1];
    int i, j;

    for (i = 0; i <= m; i++)
    {
        for (j = 0; j <= n; j++)
        {
            if (i == 0 || j == 0)
                L[i][j] = 0;

            else if (X[i - 1] == Y[j - 1])
                L[i][j] = L[i - 1][j - 1] + 1;

            else
                L[i][j] = max(L[i - 1][j], L[i][j - 1]);
        }
    }
    return L[m][n];
}
int sameMatch(string s)
{
    int max = 0, maxi = -1;
    for (int i = 0; i < 248; i++)
    {
        int val = lcs(s, station[i].name);
        if (val > max)
        {
            maxi = station[i].code;
            max = val;
        }
    }
    if (max < station[maxi].name.length() / 2)
```

```
        return -1;
    return maxi;
}
void menu()
{
    clrscreen();
    delay(90);
    gotoxy(48, 19);
    cout << "Select Mode of Travel" << endl;
    gotoxy(16, 23);
    cout << "0.Road" << endl;
    gotoxy(72, 23);
    cout << "1.Metro" << endl;
    gotoxy(48, 21);
    int choice;
    cin >> choice;
    delay(90);
    switch (choice)
    {
    case 0:
        secondWindow();
    case 1:
        secondWindow();
    }
}
void take_input()
{
    char ch;
    // scanf("%c",&ch);
    string start_s, end_s, inter_s, line, col;
    gotoxy(16, 3);
    cout << "ENTER THE STARTING POINT:";
    gotoxy(20, 5);
    // fflush();
    getline(cin, start_s);
    gotoxy(72, 3);
    cout << "ENTER THE DESTINATION POINT:";
    gotoxy(76, 5);
    getline(cin, end_s);
    gotoxy(42, 6);
```

```
    cout << "ENTER THE INTERMEDIATE POINT:";
    gotoxy(46, 8);
    getline(cin, inter_s);

    // for(int i=0;i<V;i++)
    // cout<<station[i].name<<" "<<station[i].code<<"
"<<station[i].color<<endl;

    start_s = makecapital(start_s);
    end_s = makecapital(end_s);
    inter_s = makecapital(inter_s);

    int startcode, endcode, intercode = -1;
    int startflag = 0, endflag = 0, interflag1 = 0, interflag2 = 0;
    if (!(inter_s == "NO" || inter_s == "N"))
        interflag1 = 1;
    startcode = sameMatch(start_s);
    endcode = sameMatch(end_s);
    intercode = sameMatch(inter_s);
    if (startcode == -1)
        startflag = 1;
    if (endcode == -1)
        endflag = 1;
    if (endcode == -1)
        interflag2 = 1;
    /*for(int i=0;i<248;i++)
    {
        if(station[i].name==start_s)
        {
            startcode=station[i].code;
            startflag=1;
        }
        if(station[i].name==end_s)
        {
            endcode=station[i].code;
            endflag=1;
        }
        if(interflag1&&station[i].name==inter_s)
        {
          intercode=station[i].code;
```

```
        interflag2=1;
      }
  }*/
  int fault = 0;
  if (startflag == 1)
  {
      gotoxy(42, 10);
      cout << "WRONG STARTING NAME ENTERED";
      fault = 1;
  }
  if (endflag == 1)
  {
      gotoxy(40, 11);
      cout << "WRONG DESTINATION NAME ENTERED";
      fault = 1;
  }
  if (interflag1 && interflag2 == 1)
  {
      gotoxy(39, 12);
      cout << "WRONG INTERMEDIATE NAME ENTERED";
      fault = 1;
  }
  // ch;
  scanf("%c", &ch);
  if (fault)
  {
      secondWindow();
      return;
  }
  if (interflag1)
  {
      dijkstra(graph, startcode, intercode, 0);

      dijkstra(graph, intercode, endcode, 1);

      dijkstra(graph_t, startcode, intercode, 0);

      dijkstra(graph_t, intercode, endcode, 1);
  }
  else
```

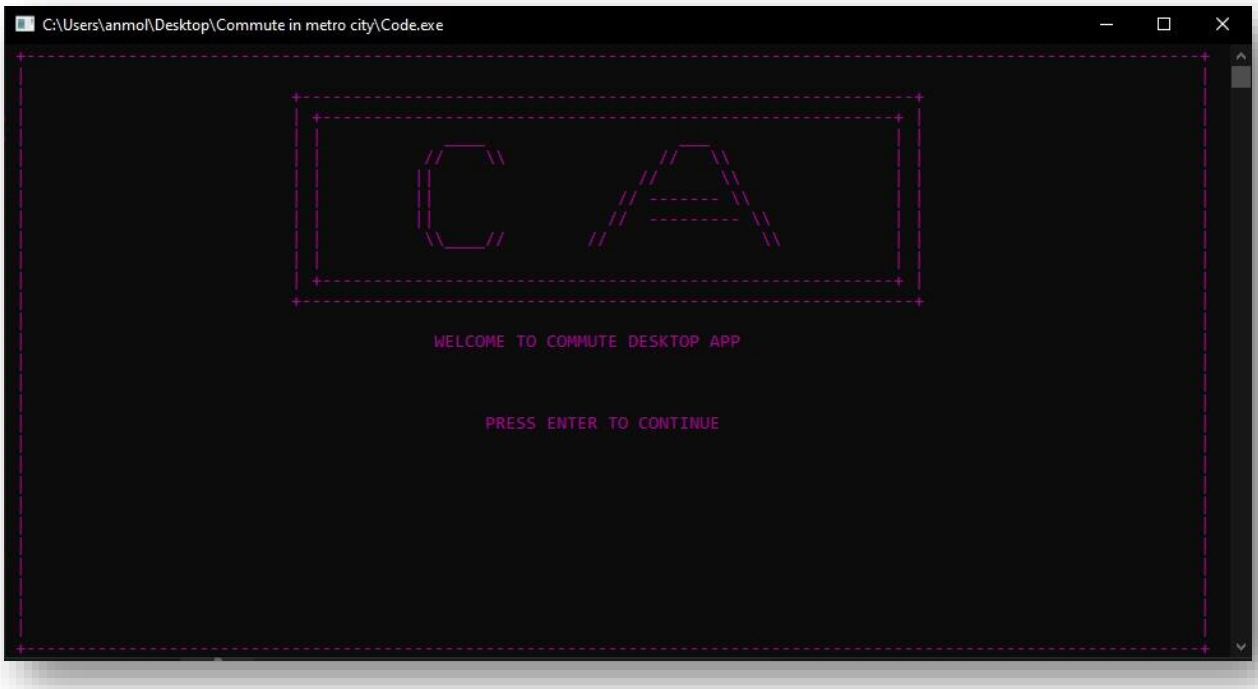
```
    {
        dijkstra(graph, startcode, endcode, 1);

        dijkstra(graph_t, startcode, endcode, 1);
    }
}

void logo(int x, int y)
{
    gotoxy(x, y);
    printf("      ____                    ___  ");
    gotoxy(x, y + 1);
    printf("    //    \\\\                //   \\\\");
    gotoxy(x, y + 2);
    printf("   ||                     //      \\\\");
    gotoxy(x, y + 3);
    printf("   ||                   // ------- \\\\");
    gotoxy(x, y + 4);
    printf("   ||                  //  --------- \\\\");
    gotoxy(x, y + 5);
    printf("    \\\\____//         //               \\\\");
}

void secondWindow()
{
    clrscreen();
    gotoxy(48, 19);
    system("color 0C");
    delay(90);
    take_input();
}

void UI()
{
    system("color 0A");
    drawbox(1, 0, 117, 29, 4);
    system("color 0C");
    delay(90);
    system("color 0A");
    delay(90);
```

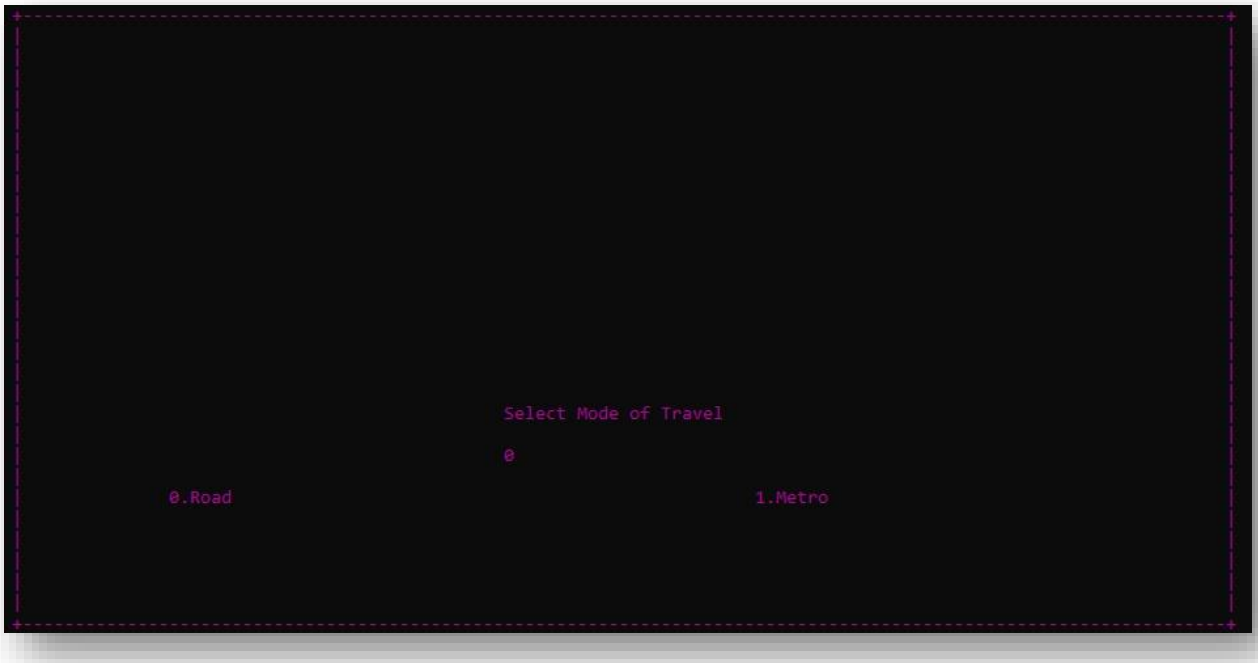
```
    logo(37, 4);
    system("color 0C");
    delay(90);
    drawbox(30, 3, 87, 11, 4);
    drawbox(28, 2, 89, 12, 4);
    system("color 0C");
    delay(90);
    system("color 0A");
    delay(90);
    gotoxy(42, 14);
    cout << "WELCOME TO COMMUTE DESKTOP APP";
    system("color 0C");
    delay(90);
    system("color 0D");
    delay(90);
    gotoxy(47, 18);
    printf("PRESS ENTER TO CONTINUE");
    char ch;
    scanf("%c", &ch);
    menu();
}

int main()
{
    int temp, n1, n2;
    float dis;
    ifstream fin, fin2;
    fin.open("node_values_new.txt");
    fin2.open("node_values_with_speed.txt");
    /*      while (fin) {

            // Read a Line from File
            getline(fin, line);

            // Print line in Console
            cout << line << endl;
        }
    */
    for (int i = 0; i < 248; i++)
    {
```

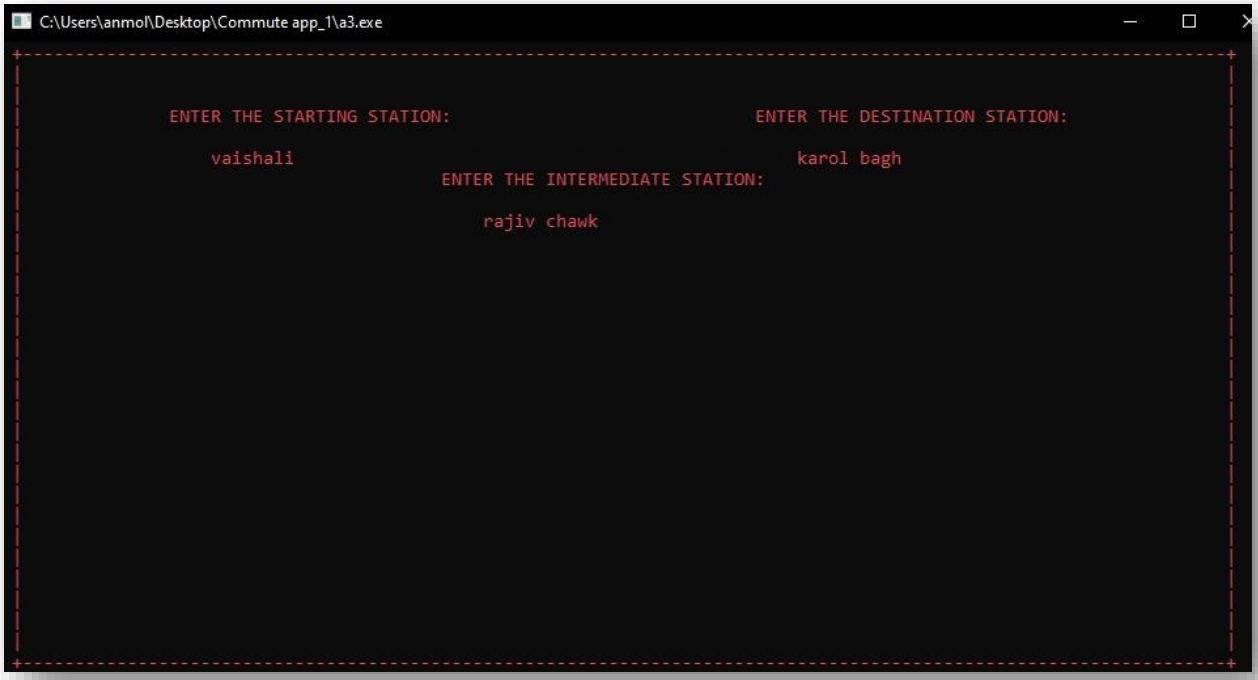
```
        for (int j = 0; j < 248; j++)
            graph[i][j] = 0;
    }

    for (int i = 0; i < 248; i++)
    {
        for (int j = 0; j < 248; j++)
            graph_t[i][j] = 0;
    }

    for (int i = 1; i <= V; i++)
    {
        fin >> temp;
        fin >> n1;
        // cout<<temp<<" "<<n1<<endl;
        for (int j = 0; j < temp; j++)
        {
            fin >> n2;
            fin >> dis;
            // cout<<" "<<n2<<" "<<dis<<endl;
            graph[n1 - 1][n2 - 1] = dis;
        }
        // cout<<endl;
    }

    for (int i = 1; i <= V; i++)
    {
        fin2 >> temp;
        fin2 >> n1;
        // cout<<temp<<" "<<n1<<endl;
        for (int j = 0; j < temp; j++)
        {
            fin2 >> n2;
            fin2 >> dis;
            // cout<<" "<<n2<<" "<<dis<<endl;
            graph_t[n1 - 1][n2 - 1] = dis;
        }
        // cout<<endl;
    }
```

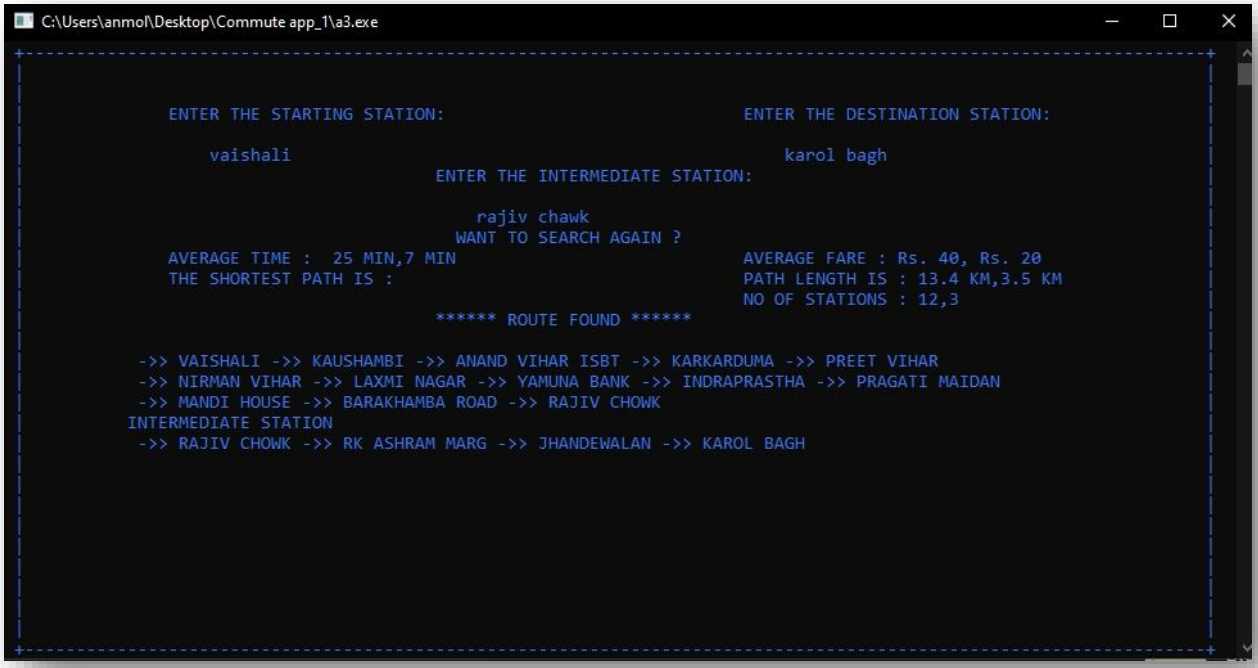
```
    string line, col;
    ifstream code;
    ifstream color;
    code.open("stationcodes.txt");
    color.open("stationcolorcodes.txt");
    for (int i = 0; i < V; i++)
    {
        getline(code, line);
        station[i].name = line;
        station[i].code = i;
        getline(color, col);
        station[i].color = col;
    }
    // Writing the matrix
    /*ofstream matrix;
    matrix.open("matrix.txt");
    matrix<<"  ";
    for(int i=0;i<V;i++)
        matrix<<i<<" ";
    matrix<<endl;
    for(int i=0;i<V;i++)
    {
        for(int j=0;j<V;j++)
        {
            if(j==0)
                matrix<<i<<" ";
            matrix<<graph[i][j]<<" ";
        }
        matrix<<endl;
    }*/
    UI();
    // secondWindow();
    // take_input(graph);
    /*Uncomment UI for accessing the introductory slide
      uncomment secondWindow and comment UI for debugging from the input
part*/
}
```

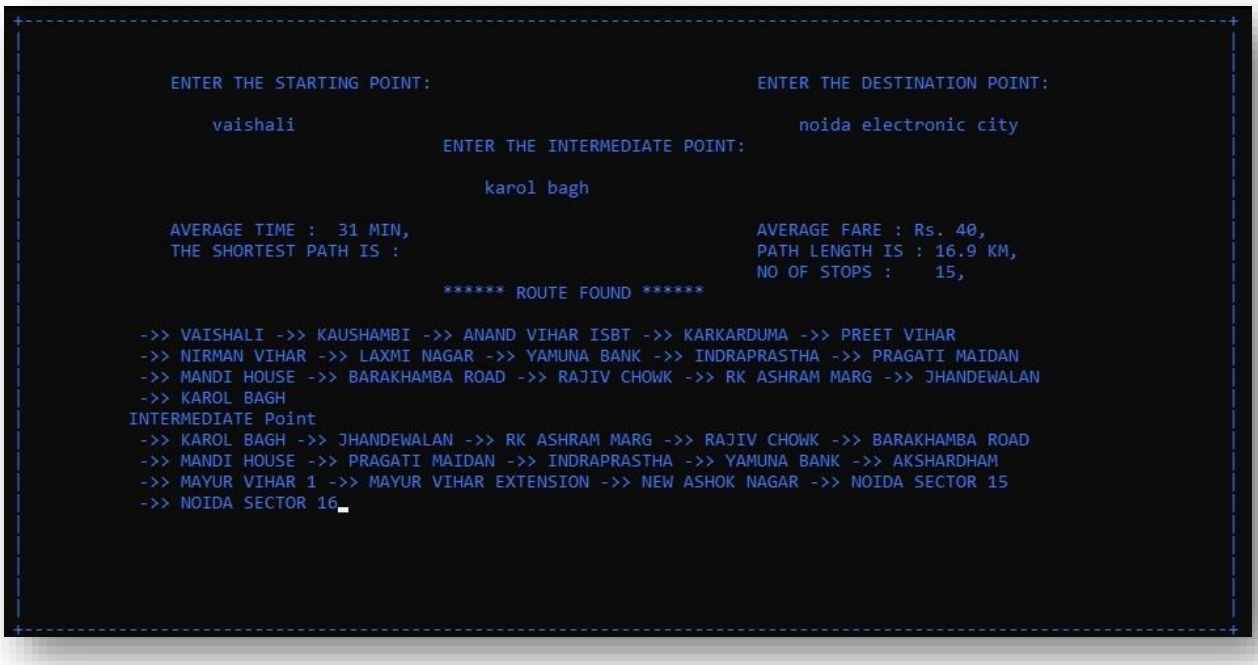
RESULTS


C:\Users\anmol\Desktop\Commute in metro city\Code.exe
WELCOME TO COMMUTE DESKTOP APP
PRESS ENTER TO CONTINUE


Select Mode of Travel
0
0.Road
1.Metro


C:\Users\anmol\Desktop\Commute app_1\a3.exe
ENTER THE STARTING STATION:
vaishali
ENTER THE DESTINATION STATION:
karol bagh
ENTER THE INTERMEDIATE STATION:
rajiv chawk


C:\Users\anmol\Desktop\Commute app_1\a3.exe
ENTER THE STARTING STATION:
vaishali
ENTER THE DESTINATION STATION:
karol bagh
ENTER THE INTERMEDIATE STATION:
rajiv chawk
WANT TO SEARCH AGAIN ?
AVERAGE TIME : 25 MIN,7 MIN
AVERAGE FARE : Rs. 40, Rs. 20
THE SHORTEST PATH IS :
PATH LENGTH IS : 13.4 KM,3.5 KM
NO OF STATIONS : 12,3
****** ROUTE FOUND ******
->> VAISHALI ->> KAUSHAMBI ->> ANAND VIHAR ISBT ->> KARKARDUMA ->> PREET VIHAR
->> NIRMAN VIHAR ->> LAXMI NAGAR ->> YAMUNA BANK ->> INDRAPRASTHA ->> PRAGATI MAIDAN
->> MANDI HOUSE ->> BARAKHAMBA ROAD ->> RAJIV CHOWK
INTERMEDIATE STATION
->> RAJIV CHOWK ->> RK ASHRAM MARG ->> JHANDEWALAN ->> KAROL BAGH


ENTER THE STARTING POINT:
vaishali
ENTER THE DESTINATION POINT:
noida electronic city
ENTER THE INTERMEDIATE POINT:
karol bagh
AVERAGE TIME : 31 MIN,
THE SHORTEST PATH IS :
AVERAGE FARE : Rs. 40,
PATH LENGTH IS : 16.9 KM,
NO OF STOPS : 15,
****** ROUTE FOUND ******
->> VAISHALI ->> KAUSHAMBI ->> ANAND VIHAR ISBT ->> KARKARDUMA ->> PREET VIHAR
->> NIRMAN VIHAR ->> LAXMI NAGAR ->> YAMUNA BANK ->> INDRAPRASTHA ->> PRAGATI MAIDAN
->> MANDI HOUSE ->> BARAKHAMBA ROAD ->> RAJIV CHOWK ->> RK ASHRAM MARG ->> JHANDEWALAN
->> KAROL BAGH
INTERMEDIATE Point
->> KAROL BAGH ->> JHANDEWALAN ->> RK ASHRAM MARG ->> RAJIV CHOWK ->> BARAKHAMBA ROAD
->> MANDI HOUSE ->> PRAGATI MAIDAN ->> INDRAPRASTHA ->> YAMUNA BANK ->> AKSHARDHAM
->> MAYUR VIHAR 1 ->> MAYUR VIHAR EXTENSION ->> NEW ASHOK NAGAR ->> NOIDA SECTOR 15
->> NOIDA SECTOR 16

Time and Space Complexity Analysis-

Time Complexity of Dijkstra's Algorithm is O (v^2) but with min-priority queue it drops down to O (v+ E l o g v) .

Space complexity of Dijkstra's algorithm is O(v^2) where v denotes the number of vertices (or nodes) in the graph.

FUTURE APPLICATION

Hoping to feed live data to relate to traffic delay and find the shortest paths in according to this projects algorithm·

Utilization of the given algorithm to cover different metro cities – We will cover different metro cities around the country with two major operations on

1) To calculate fare between two metro stations
2) To calculate shortest distance between two operating metro stations to help save time for commuting within the city

· We will make use of our project to employ a working web application which will help all the internet users to productively use our application to find both fare and shortest distance between starting point and their respective destination.

CONCLUSION

This project covers all the prospects that would help users to find the shortest and reasonable route to reach their destination. This project will utilize the algorithm to calculate fare and also travel time which will help save time and provide convenience to the users to commute between different metro stations.