JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, SECTOR-62 NOIDA

ALGORITHM AND PROBLEM SOLVING (15B17CI471)

AMRIT DAAN (FOOD WASTAGE MINIMISATION)

SUBMITTED BY:

SAUMIL GUPTA	22103179
TANUSH	22103157
KARAN ARORA	22103155
KARAN NAVEEN SOOD	22103180

SUBMITTED TO: DR ANITA SAHOO

PROBLEM STATEMENT:

Our project aims to combat the escalating rates of malnourishment and food insecurity in India by developing a technology-driven solution to reduce food wastage. By efficiently connecting surplus food from restaurants, grocery stores, and events with individuals and organizations in need, we seek to ensure that edible food reaches those facing food scarcity. Through this initiative, we aim to reduce malnutrition and prevent food-related fatalities in our communities.

MOTIVATION:

Our project is driven by a deep sense of compassion and empathy towards those experiencing food insecurity in India. It seeks to address this pressing issue by leveraging technology to reduce food wastage and redistribute surplus food to individuals and organizations in need. This endeavor is motivated by moral values such as social justice, responsibility, dignity, and solidarity, as well as the long-term goal of sustainability and collective action for positive societal change.

OBJECTIVE:

The project's objective is to tackle the rising challenges of malnutrition and food insecurity in India by employing technology-driven approaches to minimize food wastage and redistribute surplus food effectively. Through the connection of surplus food outlets such as restaurants,

grocery stores, and events with individuals and groups experiencing food scarcity, the project aims to guarantee that edible food reaches those who need it most. Ultimately, the aim is to mitigate malnutrition and avert food-related fatalities in communities throughout India.

REQUIREMENTS SPECIFICATION

Data Structures:

1. File Handling: File handling algorithms used for securely storing and verifying user passwords.

2. Graphs: Utilized for representing relationships between users, locations, and food items, enabling efficient matching and routing**.**

Algorithms:

1. **Maximum Bipartite Matching Algorithm:** Implementing algorithm to ensure such that maximum NGOs can be served.

2. **Graph Algorithms (Travelling Salesman Algorithm):** Utilized for optimizing pickup/delivery routes and minimizing travel distances between donors and recipients.

Technologies:

1. C++: Used for implementing the backend logic, including data management, algorithm development.

ALGORITHMS

1.**Travelling Salesman Problem (TSP):** The Travelling Salesman Problem (TSP) is an optimization problem. It involves finding the shortest possible route that visits each city exactly once and returns to the origin city. It could be implemented using various algorithms, including backtracking and dynamic programming.

2. **Maximum Bipartite Matching:** The Maximum Bipartite Matching problem deals with finding the largest possible set of mutually compatible pairs between two disjoint sets of vertices in a bipartite graph. In a bipartite graph, the vertices are divided into two independent sets, and edges only exist between vertices from different sets. Maximum Bipartite Matching has applications in various fields, including job assignment, resource allocation.

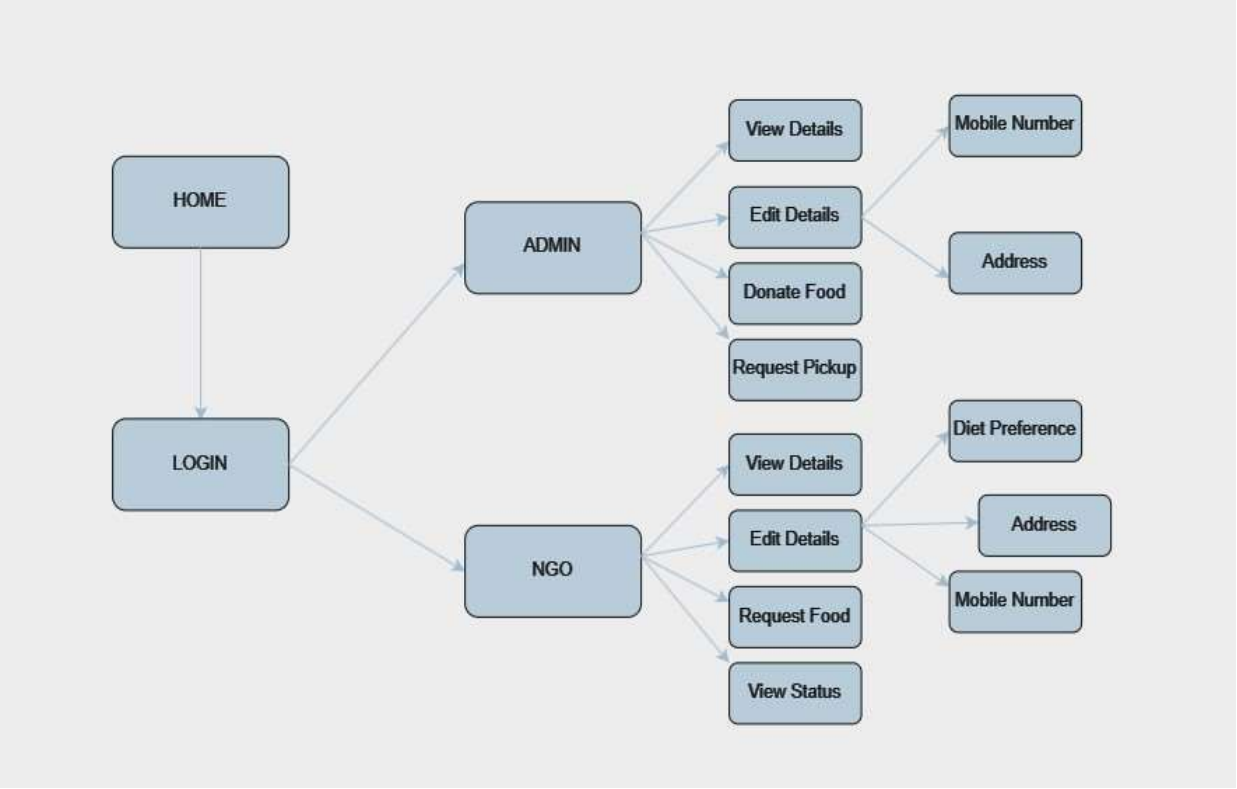
3. **Greedy Algorithms:** Greedy algorithms make locally optimal choices at each step with the

hope of finding a global optimum solution. At each decision point, a greedy algorithm selects the best available option without considering the overall future consequences. Examples of greedy algorithms include Prim's algorithm for minimum spanning trees and Dijkstra's algorithm for single-source shortest paths.

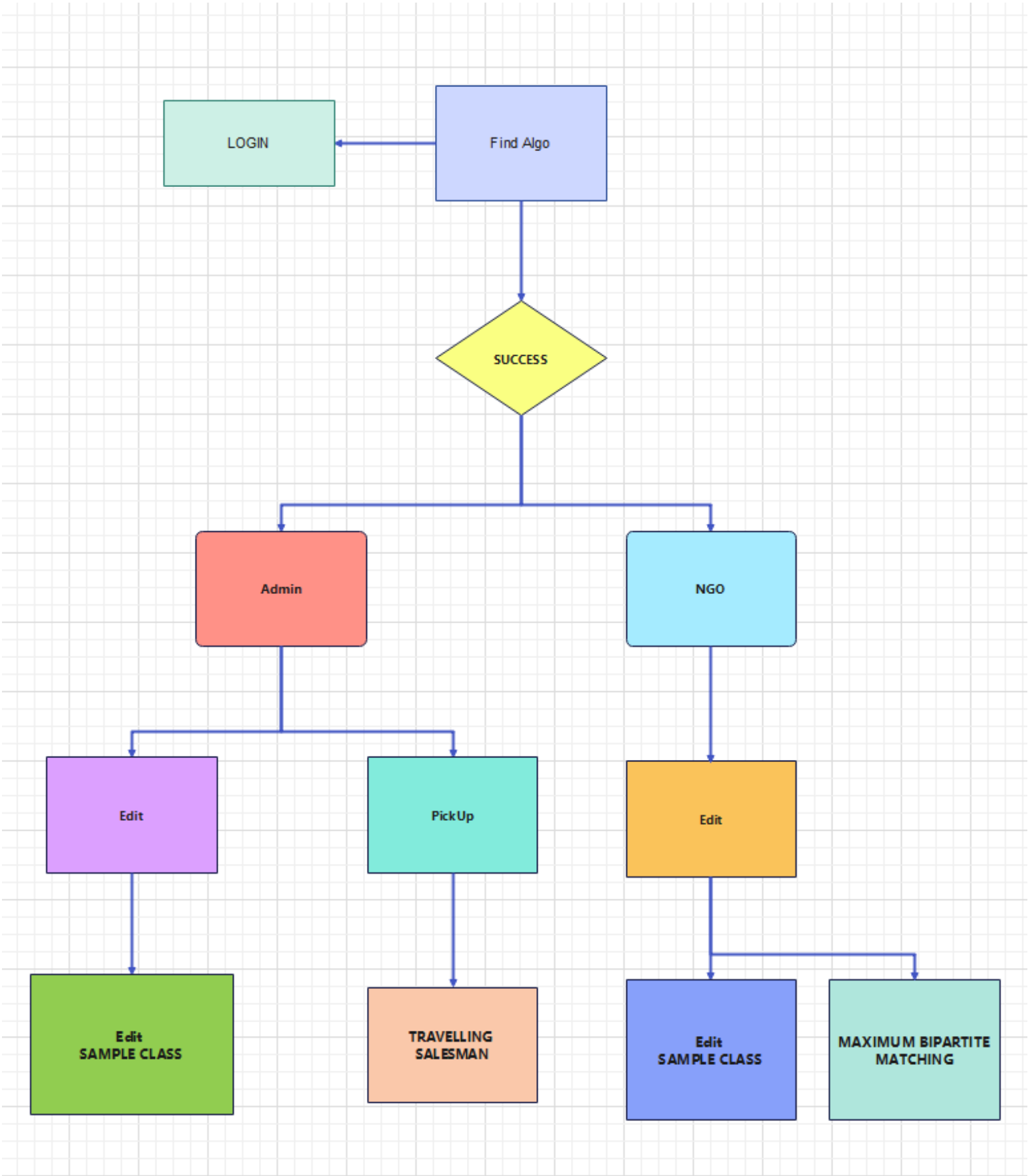
4. **Graphs:** Graphs are mathematical structures consisting of vertices (nodes) and edges (connections) between them. They are versatile data structures used to model relationships and connections between objects. Graphs can represent a wide range of real-world scenarios, including social networks, transportation networks.

5. **File Handling:** File handling refers to the process of reading from and writing to files on a computer's storage system. In programming, file handling involves operations such as opening, reading, writing, and closing files.

CASE DIAGRAM



DATA FLOW DIAGRAM



IMPLEMENTATION

```
// FINAL PROJECT
// FOOD DONATION APP
#include <bits/stdc++.h>
using namespace std;

class graph
{
    public:
    string src;
    string dest;
    int distance;
    map<string, list<pair<string, int>>> m;
    graph(){}
    graph(string x,string y,int z)
    {
        src=x;
        dest=y;
        distance=z;
    }
    void add(string x,string y,int z)
    {
        // UNDIRECTED GRAPH
        m[x].push_back({y,z});
        m[y].push_back({x,z});
    }
};

class ngo
{
    public:
    string name;
    int quantity;
    string type;
    ngo(){}
    ngo(string x,int y,string z)
    {
        name=x;
        quantity=y;
        type=z;
    }
};

bool compare(ngo n1,ngo n2)
{
    // SORT IN ASCENDING ORDER
    return n1.quantity<n2.quantity;
}
```

```
class sample
{
    public:
    sample(){}
    // FINDING ALORITHM
    void findalgo(string id,string password,string s,int &ans,vector<string>*v)
    {
        ifstream fptr;
        fptr.open(s);
        string s1, s2, s3, s4, s5,s6;
        bool found = false;
        while (getline(fptr, s1, ' ') && !found)
        {
            getline(fptr, s2, ' ');
            getline(fptr, s3, ' ');
            getline(fptr, s4, ' ');
            getline(fptr, s5, ' ');
            getline(fptr, s6, '\n');
            if (s1 == id && s2==password)
            {
                found = true;
                v->push_back(s1);
                v->push_back(s2);
                v->push_back(s3);
                v->push_back(s4);
                v->push_back(s5);
                v->push_back(s6);
            }
        }
        fptr.close();
        if(found==0)
        {
            ans=0;
        }
        else
        {
            ans=1;
        }
    }
    void edit(string username,string password,string s,int field,string newData)
    {
        int ss;
        sample A;
        vector<string> *v = new vector<string>();
        A.findalgo(username, password,s, ss, v);
        (*v)[field] = newData;
        ifstream fin(s);
        ofstream fout("temp.txt");
        string line;
        while (getline(fin, line))
        {
            if (line.find(username) != string::npos)
            {
```

```
                fout << (*v)[0];
                for (int i = 1; i < v->size(); ++i)
                {
                    fout << " " << (*v)[i];
                }
                fout << endl;
            }
            else
            {
                fout << line << endl;
            }
        }
        fin.close();
        fout.close();
        remove(s.c_str());
        rename("temp.txt", s.c_str());
        cout << "RECORDS UPDATED SUCCESSFULLY " << endl;
        delete v;
    }
};

void tsp(graph &g, map<string, bool> &visited, string currPos, int n, int count,
int cost, int &ans, vector<string> &path, vector<string> &bestPath)
{
    if (count == n)
    {
        for (auto &edge : g.m[currPos])
        {
            if (edge.first == "Noida" && cost + edge.second < ans)
            {
                ans = cost + edge.second;
                bestPath = path;
                bestPath.push_back("Noida");
            }
        }
        return;
    }

    for (auto i : g.m[currPos])
    {
        string city = i.first;
        int dist = i.second;
        if (!visited[city])
        {
            visited[city] = true;
            path.push_back(city);
            tsp(g, visited, city, n, count + 1, cost + dist, ans, path,
bestPath);
            // BACKTRACK
            path.pop_back();
            visited[city] = false;
        }
    }
```

```
}

class admin
{
    public:
    admin()
    {
        // LOGIN WINDOW FOR AUTHENTICATION OF USER
        mainmenu:
        cout << " ................. " <<endl;
        cout  << "   LOGIN WINDOW : " << endl;
        cout << " ................. " << endl<< endl;

        cout << "ENTER USERNAME : ";
        string username;
        cin >> username;
        cout << "ENTER PASSWORD : ";
        string password;
        cin>>password;
        // CODE FOR 4 DIGIT CAPTCHA GENERATION
        // SECOND FACTOR AUTHENTICATION
        srand(time(0));
        int ans = 0;
        int CAPTCHA_LENGTH = 4;
        int captcha = rand() % (9000) + 1000;
        cout << endl<< endl<< "ENTER CAPTCHA " << captcha << " : ";
        cin >> ans;
        sample s;
        vector<string> *v = new vector<string>();
        int as;
        s.findalgo(username,password,"DonorDetails.TXT",as,v);
        delete v;
        while (as!=1)
        {
            cout << endl<< "INCORRECT USERNAME OR PASSWORD OR CAPTCHA " << endl;
            cout << "TRY AGAIN !!! " << endl;
            goto mainmenu;
        }
        cout << endl<<endl<< "LOGIN SUCCESSFUL !!" << endl;
        cout << "ACCESS GRANTED !!!" << endl<< endl;
        while(true)
        {
            m:
            cout<<endl<<endl<<"            ******************************"<<end
l;
            cout<<"             ENTER 1 TO VIEW PROFILE"<<endl;
            cout<<"             ENTER 2 TO EDIT PROFILE"<<endl;
            cout<<"             ENTER 3 TO DONATE FOOD"<<endl;
            cout<<"             ENTER 4 FOR PICKUP "<<endl;
            cout<<"             ENTER 5 TO GO BACK "<<endl;
            cout<<"             ENTER YOUR CHOICE : ";
            int choice;
            cin>>choice;
```

```
            cout << "            ******************************" << endl<<
endl;
            switch(choice)
            {
                case 1:
                {
                    sample s;
                    vector<string> *v = new vector<string>();
                    int aa;
                    s.findalgo(username,password,"DonorDetails.TXT",aa,v);
                    cout<<"------------------------------ ";
                    cout<<endl<<"DISPLAYING DETAILS : "<<endl;
                    cout<<"------------------------------ "<<endl;
                    cout<<"FIRST NAME         : "<<(*v)[0]<<endl;
                    cout<<"AGE                : "<<(*v)[2]<<endl;
                    cout<<"MOBILE NUMBER      : "<<(*v)[3]<<endl;
                    cout<<"QUANTITIES DONATED : "<<(*v)[4]<<endl;
                    cout<<"ADDRESS            : "<<(*v)[5]<<endl;
                    cout<<"------------------------------ "<<endl;
                    delete v;
                    break;
                }
                case 2:
                {  cout<<"---------------------------------- "<<endl;
                    cout<<"ENTER 1 TO UPDATE MOBILE NUMBER "<<endl;
                    cout<<"ENTER 2 TO UPDATE ADDRESS "<<endl;
                    cout<<"ENTER 3 TO GO BACK "<<endl;
                    int choice;
                    cout<<"ENTER YOUR CHOICE : ";
                    cin>>choice;

                    switch(choice)
                    {
                        case 1:
                        {
                            string mobile;
                            cout<<"ENTER UPDATED MOBILE NUMBER : ";
                            cin>>mobile;

                            sample sss;
                            sss.edit(username,password,"DonorDetails.TXT",3,mobi
le);
                            break;
                        }
                        case 2:
                        {
                            string address;
                            cout<<"ENTER UPDATED ADDRESS : ";
                            cin>>address;
                            sample sss;
                            sss.edit(username,password,"DonorDetails.TXT",5,addr
ess);
                            break;
```

```
                    }
                    case 3:
                    {
                        goto m;
                    }
                    default:
                    {
                        cout<<"INVALID CHOICE "<<endl;
                    }
                }
                     cout<<"---------------------------------- ";
                break;
            }
            case 3:
            {
                int quantity;
                cout<<"------------------------------------ "<<endl;
                cout << "ENTER QUANTITY OF FOOD TO BE DONATED : ";
                cin >> quantity;
                string type;
                cout << "ENTER TYPE OF FOOD(VEG/NON VEG) : ";
                cin >> type;
                int ss;
                sample s;
                vector<string> *v = new vector<string>();
                s.findalgo(username, password, "DonorDetails.TXT", ss, v);
                ofstream fptr;
                if(!fptr)
                {
                    cout<<"ERROR IN OPENING FILE "<<endl;
                }
                fptr.open("DONATES.TXT",ios::app);
                fptr<<username<<" "<<(*v)[5]<<" "<<quantity<<"
"<<type<<endl;
                fptr.close();

                int itemsDonated = stoi((*v)[4]);
                itemsDonated += quantity;
                (*v)[4] = to_string(itemsDonated);
                ifstream fin("DonorDetails.TXT");
                ofstream fout("temp.txt");
                string line;
                while (getline(fin, line))
                {
                    if (line.find(username) != string::npos)
                    {
                        fout << (*v)[0];
                        for (int i = 1; i < v->size(); ++i)
                        {
                            fout << " " << (*v)[i];
                        }
                        fout << endl;
                    }
```

```
                    else
                    {
                        fout << line << endl;
                    }
                }
                fin.close();
                fout.close();
                remove("DonorDetails.TXT");
                rename("temp.txt", "DonorDetails.TXT");
                cout << "RECORDS UPDATED SUCCESSFULLY " << endl;
                delete v;
                cout<<"------------------------------------- "<<endl;
                break;
            }
            case 4:
            {
                // GLOBAL MAP
                graph mapss;
                mapss.add("Noida","Delhi",50);
                mapss.add("Noida","Bangalore",2000);
                mapss.add("Noida","Chennai",3000);
                mapss.add("Delhi","Chennai",2500);
                mapss.add("Bangalore","Chennai",300);

               ifstream fptr;
               fptr.open("DONATES.TXT");
                if(!fptr)
                {
                    cout<<"ERROR IN OPENING FILE "<<endl;
                }
                string s1, s2, s3, s4;
                vector<string> *v = new vector<string>();
                while (getline(fptr, s1, ' '))
                {
                    getline(fptr, s2, ' ');
                    getline(fptr, s3, ' ');
                    getline(fptr, s4, '\n');
                    v->push_back(s2);
                }
                fptr.close();


                // DELIVERY VAN NEEDS TO VISIT ALL LOCATIONS GIVEN IN V
VECTOR ACC TO TSP PROBLEM
                v->push_back("Noida");
                // CONSTRUCTING THE GRAPH
                graph g;
                bool a=true;
                for (int i = 0;i<v->size(); i++)
                {
                    for (int j=i + 1; j<v->size(); j++)
                    {
                        // CHECK IF THIS VERTEX EXISTS IN THE GRAPH
```

```
                        if (mapss.m.find((*v)[i]) != mapss.m.end())
                        {
                            for (auto &edge : mapss.m[(*v)[i]])
                            {
                                if (edge.first == (*v)[j])
                                {
                                    g.add((*v)[i], (*v)[j], edge.second);
                                    break;
                                }
                            }
                        }
                    }
                    // DELIVERY VAN WILL START FROM NOIDA AND BRING BACK FOOD TO
NOIDA(DELIVERY CENTER)
                    string src="Noida";
                    string dest="Noida";
                    map<string, bool> vv;
                    for (auto &it : g.m)
                    vv[it.first] = false;

                    vv["Noida"] = true;
                    int ans = INT_MAX;
                    vector<string> path, bestPath;
                    tsp(g, vv, "Noida", g.m.size(), 1, 0, ans, path, bestPath);
                    if(!bestPath.empty())
                    {
                        cout<<"Noida -> ";
                        for(int i=0;i<bestPath.size();i++)
                        {
                            cout<<bestPath[i]<<"-> ";
                        }
                        cout<<endl<<"MINIMUM DISTANCE IS "<<ans<<" KMS"<<endl;
                    }
                    else
                    {
                        cout<<"NO ROUTE FOUND "<<endl;
                    }
                    delete v;
                    break;
                }
                case 5:
                {
                    return;
                }
                default:
                {
                    cout<<"INVALID CHOICE "<<endl;
                }
            }
        }
    }
};
```

```
class user
{
    public:
    user()
    {
        // LOGIN WINDOW FOR AUTHENTICATION OF USER
        mainmenu:
        cout << " ................. " <<endl;
        cout << endl<< "   LOGIN WINDOW : " << endl;
         cout << " ................. " <<endl<<endl;
        cout << "ENTER USERNAME : ";
        string username;
        cin >> username;
        cout << "ENTER PASSWORD : ";
        string password;
        cin>>password;
        // CODE FOR 4 DIGIT CAPTCHA GENERATION
        // SECOND FACTOR AUTHENTICATION
        srand(time(0));
        int ans = 0;
        int CAPTCHA_LENGTH = 4;
        int captcha = rand() % (9000) + 1000;
        cout << endl<< endl<< "ENTER CAPTCHA " << captcha << " : ";
        cin >> ans;
        sample s;
        vector<string> *v = new vector<string>();
        int as;
        s.findalgo(username,password,"NgoDetails.TXT",as,v);
        while (as!=1)
        {
            cout << endl<< "INCORRECT USERNAME OR PASSWORD OR CAPTCHA " << endl;
            cout << "TRY AGAIN !!! " << endl;
            goto mainmenu;
        }
        cout << endl<<endl<< "LOGIN SUCCESSFUL !!" << endl;
        cout << "ACCESS GRANTED !!!" << endl<< endl;
        while(true)
        {
            ssss:
            cout<<endl<<endl<<"            ******************************"<<end
l;
            cout<<"             ENTER 1 TO VIEW PROFILE"<<endl;
            cout<<"             ENTER 2 TO EDIT PROFILE"<<endl;
            cout<<"             ENTER 3 TO DEMAND FOOD"<<endl;
            cout<<"             ENTER 4 TO VIEW STATUS "<<endl;
            cout<<"             ENTER 5 TO GO BACK "<<endl;
            cout<<"             ENTER YOUR CHOICE : ";
            int choice;
            cin>>choice;
               cout<<endl<<"            ******************************"<<endl;
            switch(choice)
```

```
            {
                case 1:
                {
                    sample s;
                    vector<string> *v = new vector<string>();
                    int aa;
                    s.findalgo(username,password,"NgoDetails.TXT",aa,v);
                        cout<<"------------------------------ ";
                    cout<<endl<<"DISPLAYING DETAILS : "<<endl;
                   cout<<"------------------------------ "<<endl;
                    cout<<"NGO NAME            : "<<(*v)[0]<<endl;
                    cout<<"MOBILE NUMBER       : "<<(*v)[2]<<endl;
                    cout<<"ADDRESS             : "<<(*v)[3]<<endl;
                    cout<<"NO OF CHILDREN      : "<<(*v)[4]<<endl;
                    cout<<"DIETARY PREFERENCE  : "<<(*v)[5]<<endl;
                      cout<<"------------------------------ ";
                    delete v;
                    break;
                }
                case 2:
                {    cout<<"------------------------------------ "<<endl;
                    cout<<"ENTER 1 TO UPDATE MOBILE NUMBER "<<endl;
                    cout<<"ENTER 2 TO UPDATE ADDRESS "<<endl;
                    cout<<"ENTER 3 TO UPDATE DIETARY PREFERENCE "<<endl;
                    cout<<"ENTER 4 TO GO BACK "<<endl;
                    int choice;
                    cout<<"ENTER YOUR CHOICE : ";
                    cin>>choice;
                    switch(choice)
                    {
                        case 1:
                        {
                            string mobile;
                            cout<<"ENTER UPDATED MOBILE NUMBER : ";
                            cin>>mobile;
                            sample sss;
                            sss.edit(username,password,"NgoDetails.TXT",2,mobile
);
                            break;
                        }
                        case 2:
                        {
                            string address;
                            cout<<"ENTER UPDATED ADDRESS : ";
                            cin>>address;
                            sample sss;
                            sss.edit(username,password,"NgoDetails.TXT",3,addres
s);
                            break;
                        }
                        case 3:
                        {
                            string diet;
```

```
                        cout<<"ENTER UPDATED DIETARY PREFERNCE : ";
                        cin>>diet;
                        sample sss;
                        sss.edit(username,password,"NgoDetails.TXT",5,diet);
                        break;
                    }
                    case 4:
                    {
                        goto ssss;
                    }
                    default:
                    {
                        cout<<"INVALID CHOICE "<<endl;
                    }
                }
                    cout<<"------------------------------------ "<<endl;
                break;
            }
            case 3:
            { cout<<"------------------------------------ "<<endl;
                int quantity;
                cout << "ENTER QUANTITY OF FOOD NEEDED : ";
                cin >> quantity;
                string type;
                cout << "ENTER TYPE OF FOOD(VEG/NON VEG) : ";
                cin >> type;
                int ss;
                sample s;
                vector<string> *v = new vector<string>();
                s.findalgo(username, password, "NgoDetails.TXT", ss, v);
                ofstream fptr;
                if(!fptr)
                {
                    cout<<"ERROR IN OPENING FILE "<<endl;
                }
                fptr.open("DEMANDS.TXT",ios::app);
                fptr<<username<<" "<<quantity<<" "<<type<<endl;
                fptr.close();
                cout<<"REQUEST STORED SUCCESSFULLY !!! "<<endl;
                 cout<<"------------------------------------ "<<endl;
                delete v;
                break;
            }
            case 4:
            {
                // OPEN FILE DONATES.TXT AND READS NO OF VEG AND NON VEG
FOOD
                ifstream fptr;
                fptr.open("DONATES.TXT");
                if(!fptr)
                {
                    cout<<"ERROR IN OPENING FILE "<<endl;
                }
```

```
string s1, s2, s3, s4;
int vegQuantity=0;
int nonvegQuantity=0;
while (getline(fptr, s1, ' '))
{
    getline(fptr, s2, ' ');
    getline(fptr, s3, ' ');
    getline(fptr, s4, '\n');
    if (s4=="VEG")
    {
        vegQuantity+=stoi(s3);
    }
    else if(s4=="NONVEG")
    {
        nonvegQuantity+=stoi(s3);
    }
}
fptr.close();

ifstream f("DEMANDS.TXT");
if(!f)
{
    cout<<"ERROR IN OPENING FILE "<<endl;
}
vector<ngo>*vegg= new vector<ngo>();
vector<ngo>*nonvegg= new vector<ngo>();
string a1,a2,a3;
while (getline(f, a1, ' '))
{
    getline(f, a2, ' ');
    getline(f, a3, '\n');
    if (a3=="VEG")
    {
        vegg->push_back(ngo{a1,stoi(a2), a3});
    }
    else if(a3=="NONVEG")
    {
        nonvegg->push_back(ngo{a1,stoi(a2), a3});
    }
}
sort(vegg->begin(),vegg->end(),compare);
sort(nonvegg->begin(),nonvegg->end(),compare);
int position=0;
// APPLYING GREEDY ALGORITHM
// ASSIGNNING FOOD TO NGOS WITH MINIMUM DEMANDS
int vegPosition=0;
int nonvegPosition=0;
for(int i=0;i<vegg->size();i++)
{
    if(vegQuantity-(*vegg)[i].quantity >=0)
    {
        vegPosition+=1;
        vegQuantity -(*vegg)[i].quantity;
```

```
                        }
                        else
                        {
                            break;
                        }
                    }

                    for(int i=0;i<nonvegg->size();i++)
                    {
                        if(nonvegQuantity - (*nonvegg)[i].quantity >=0)
                        {
                            nonvegPosition+=1;
                            nonvegQuantity-=(*nonvegg)[i].quantity;
                        }
                        else
                        {
                            break;
                        }
                    }
                     cout<<"------------------------------------- "<<endl;
                    cout<<"VEG FOOD ALLOTED TO : "<<endl;
                    for(int i=0;i<vegPosition;i++)
                    {
                        cout<<(*vegg)[i].name<<endl;
                    }
                    cout<<"NON VEG FOOD ALLOTED TO : "<<endl;
                    for(int i=0;i<nonvegPosition;i++)
                    {
                        cout<<(*nonvegg)[i].name<<endl;
                    }

                    f.close();
                    delete vegg;
                    delete nonvegg;
                    break;
                }
                case 5:
                {
                    return;
                }
                default:
                {
                    cout<<"INVALID CHOICE "<<endl;
                }
                 cout<<"------------------------------------- "<<endl;
            }
        }
    }
};

int main()
{
    system("color 1F");
```

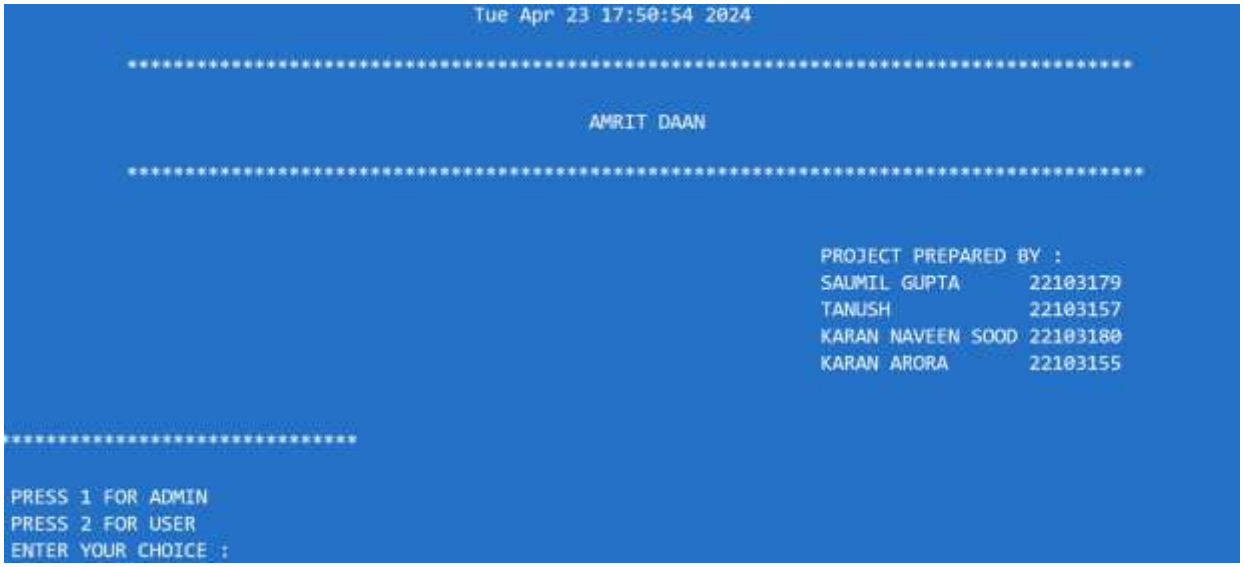
```
    while (true)
    {
        time_t now = time(0);     //  GET CURRENT DATE AND TIME WITH RESPECT TO
SYSTEM
        char *date = ctime(&now); // CONVERT IT INTO STRING
        cout << endl<< endl<< endl<< "                                        "
<< date << endl;
        cout <<
"           **********************************************************************
******************" << endl<< endl;
        cout << "                                                    AMRIT
DAAN                            " << endl<< endl;
        cout <<
"           **********************************************************************
********************" << endl<< endl<< endl;
        cout <<
"                                                                   PROJECT
PREPARED BY : " << endl;
        cout <<
"                                                                   SAUMIL
GUPTA      22103179 " << endl;
        cout <<
"                                                                   TANUSH
          22103157 " << endl;
        cout <<
"                                                                   KARAN
NAVEEN SOOD 22103180 " << endl;
        cout <<
"                                                                   KARAN
ARORA       22103155 " << endl;
        cout << endl<< endl;
         cout << "*******************************" << endl<< endl;
        cout << " PRESS 1 FOR ADMIN " << endl;
        cout << " PRESS 2 FOR USER " << endl;
        cout << " ENTER YOUR CHOICE : ";

        int key;
        cin >> key;
        cout << "*******************************" << endl<< endl;
        switch (key)
        {
            case 1:
            {
                admin a;
                break;
            }
            case 2:
            {
                user u;
                break;
            }
            case 3:
            {
```

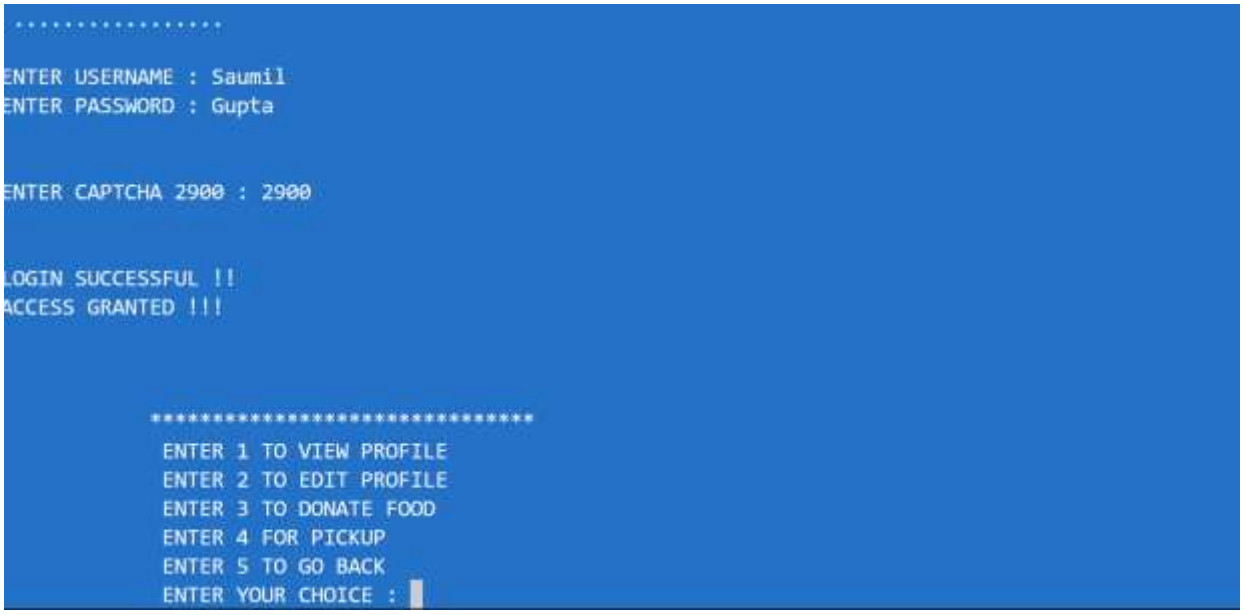
```
                exit(0);
                break;
            }
            default:
            {
                cout << "INVALID CHOICE " << endl;
                break;
            }
        }
    }
    return 0;
}
```

TEST CASES

The application has been tested across various conditions, consistently producing correct results. It has also been rigorously evaluated for both common cases and edge cases, passing all situations successfully.

RESULT


Tue Apr 23 17:50:54 2024
AMRIT DAAN
PROJECT PREPARED BY :
SAUMIL GUPTA 22103179
TANUSH 22103157
KARAN NAVEEN SOOD 22103180
KARAN ARORA 22103155
PRESS 1 FOR ADMIN
PRESS 2 FOR USER
ENTER YOUR CHOICE :


ENTER USERNAME : Saumil
ENTER PASSWORD : Gupta
ENTER CAPTCHA 2900 : 2900
LOGIN SUCCESSFUL !!
ACCESS GRANTED !!!
ENTER 1 TO VIEW PROFILE
ENTER 2 TO EDIT PROFILE
ENTER 3 TO DONATE FOOD
ENTER 4 FOR PICKUP
ENTER 5 TO GO BACK
ENTER YOUR CHOICE :

```
------------------------------
DISPLAYING DETAILS :
------------------------------
FIRST NAME         : Saumil
AGE                : 19
MOBILE NUMBER      : 7896789688
QUANTITIES DONATED : 0
ADDRESS            : Mumbai
------------------------------


          ******************************
           ENTER 1 TO VIEW PROFILE
           ENTER 2 TO EDIT PROFILE
           ENTER 3 TO DONATE FOOD
           ENTER 4 FOR PICKUP
           ENTER 5 TO GO BACK
           ENTER YOUR CHOICE :
```

```
-----------------------------------
ENTER 1 TO UPDATE MOBILE NUMBER
ENTER 2 TO UPDATE ADDRESS
ENTER 3 TO GO BACK
ENTER YOUR CHOICE : 1
ENTER UPDATED MOBILE NUMBER : 9898989898
RECORDS UPDATED SUCCESSFULLY
-----------------------------------


          ******************************
           ENTER 1 TO VIEW PROFILE
           ENTER 2 TO EDIT PROFILE
           ENTER 3 TO DONATE FOOD
           ENTER 4 FOR PICKUP
           ENTER 5 TO GO BACK
           ENTER YOUR CHOICE : 1
          ******************************
```

```
          ******************************
           ENTER 1 TO VIEW PROFILE
           ENTER 2 TO EDIT PROFILE
           ENTER 3 TO DONATE FOOD
           ENTER 4 FOR PICKUP
           ENTER 5 TO GO BACK
           ENTER YOUR CHOICE : 3
          ******************************

--------------------------------------
ENTER QUANTITY OF FOOD TO BE DONATED : 10
ENTER TYPE OF FOOD(VEG/NON VEG) : VEG
RECORDS UPDATED SUCCESSFULLY
--------------------------------------
```

```
          ******************************
           ENTER 1 TO VIEW PROFILE
           ENTER 2 TO EDIT PROFILE
           ENTER 3 TO DONATE FOOD
           ENTER 4 FOR PICKUP
           ENTER 5 TO GO BACK
           ENTER YOUR CHOICE : 4
          ******************************

Noida -> Chennai-> Noida->
MINIMUM DISTANCE IS 6000 KMS
```

```
ENTER USERNAME : HumanityExists
ENTER PASSWORD : hello


ENTER CAPTCHA 3681 : 3681


LOGIN SUCCESSFUL !!
ACCESS GRANTED !!!



          ******************************
           ENTER 1 TO VIEW PROFILE
           ENTER 2 TO EDIT PROFILE
           ENTER 3 TO DEMAND FOOD
           ENTER 4 TO VIEW STATUS
           ENTER 5 TO GO BACK
           ENTER YOUR CHOICE :
```

```
            ENTER 4 TO VIEW STATUS
            ENTER 5 TO GO BACK
            ENTER YOUR CHOICE : 1

           ******************************
--------------------------------
DISPLAYING DETAILS :
--------------------------------
NGO NAME             : HumanityExists
MOBILE NUMBER        : 78555622
ADDRESS              : Bhopal
NO OF CHILDREN       : 16
DIETARY PREFERENCE   : NONVEG
--------------------------------
```

```
            ENTER 4 TO VIEW STATUS
            ENTER 5 TO GO BACK
            ENTER YOUR CHOICE : 2

           ******************************
---------------------------------------
ENTER 1 TO UPDATE MOBILE NUMBER
ENTER 2 TO UPDATE ADDRESS
ENTER 3 TO UPDATE DIETARY PREFERENCE
ENTER 4 TO GO BACK
ENTER YOUR CHOICE : 1
ENTER UPDATED MOBILE NUMBER : 999999999
RECORDS UPDATED SUCCESSFULLY
---------------------------------------
```

```
            ENTER 3 TO DEMAND FOOD
            ENTER 4 TO VIEW STATUS
            ENTER 5 TO GO BACK
            ENTER YOUR CHOICE : 3

           ******************************
---------------------------------------
ENTER QUANTITY OF FOOD NEEDED : 10
ENTER TYPE OF FOOD(VEG/NON VEG) : VEG
REQUEST STORED SUCCESSFULLY !!!
---------------------------------------


           ******************************
            ENTER 1 TO VIEW PROFILE
```

```
           ******************************
            ENTER 1 TO VIEW PROFILE
            ENTER 2 TO EDIT PROFILE
            ENTER 3 TO DEMAND FOOD
            ENTER 4 TO VIEW STATUS
            ENTER 5 TO GO BACK
            ENTER YOUR CHOICE : 4

           ******************************
---------------------------------------
VEG FOOD ALLOTED TO :
HumanityExists
NON VEG FOOD ALLOTED TO :
```

CONCLUSION

In conclusion, our food waste reduction project embodies the spirit of "Amrit Daan" - the donation of food is the most profound gift one can offer. By employing technology, we are developing a platform that not only minimizes food wastage but also fosters a sense of community. Together, we can continue to make a meaningful impact in reducing food insecurity and ensuring that no one goes hungry. Let us embrace the aim of "Amrit Daan" and work towards a future where every individual has access to nutritious food paving the way for a healthier and more equitable society.

FUTURE SCOPE:

1.Mobile Application: Enhancing scalability by transitioning to a mobile app platform, enabling broader access and user engagement.

2.Optimization Techniques: Implementing performance optimizations to enhance program speed and efficiency, ensuring smoother user experience and resource utilization.

3. Graphical Interface: Introduce a visually appealing graphical interface to improve user engagement and usability, enhancing the overall user experience.

REFERENCES:

1. https://www.geeksforgeeks.org/greedy-algorithms/
2. https://www.javatpoint.com/dijkstras-algorithm
3. https://www.geeksforgeeks.org/graph-and-its-representations/