**DSA PROJECT**

**REPORT**

**TITLE**

**VACCINE MANAGEMENT SYSTEM**

**Submitted By:** **Submitted To:**

21104001: Sarthak Gupta Dr. Apeksha Aggarwal

21104011: Manya Manish Srivastava

21104018: Aditya Lohia

21104020: Aryan Arora

**BATCH-B14**

**ABSTRACT**

This is a fully functional C++ software based on the Vaccine Management System. The data contains the name of the persons who have taken the vaccine and also tells the current status of vaccine availability. Individuals can register their details and plan their testing time slot on the User Module side of the system. Previously the task of handling the vaccination data was very difficult, so there was a need for software that can handle all the vaccination data.

Therefore the Vaccine Management System was designed. After the release of this system, the stress and workload of employees were absolutely finished. It was also time-wasting for the employees to handle the whole vaccination data with the help of a notebook. But now it hardly takes 5 to 10 minutes to search the vaccination status of a particular person.

**Data Structures used**

* Graph
* Vector
* Map
* Queue

**Some of The Important Functions used**

* Add centre():-Allow us to add the details of new recovery centre
* Add user(): Allows us to add details of a new registered user
* Get centre():-Shows the nearest available recovery centre
* Book slot(): Provides us with the facility to book vaccination slot
* Get stats(): Gives us the information on total availability of slots

**Code:**

#include <bits/stdc++.h>

using namespace std;

class User

{

private:

string name;

char gender;

string preferred\_vaccine\_type;

int bookingDate;

public:

User(string n, char g, string pvt){

name = n;

gender = g;

preferred\_vaccine\_type = pvt;

bookingDate = -1;

canBook = true;

}

bool canBook;

string getName();

string getPreferredVaccineType();

int getBookings();

void addBooking(int date);

};

string User::getName()

{

return name;

}

string User::getPreferredVaccineType()

{

return preferred\_vaccine\_type;

}

int User::getBookings(){

return bookingDate;

}

void User::addBooking(int date){

bookingDate = date;

}

class Centre

{

private:

int covaxInventory;

int coviShieldInventory;

int covaxSlots[31][3];

int coviShieldSlots[31][3];

vector<vector<int>> bedsMatrix;

public:

string cid;

Centre(string cid, int a,int b,int c, vector<vector<int>> d)

{

this->cid = cid;

covaxInventory = a;

coviShieldInventory = b;

for(int i=0;i<31;i++) //days of the month

{

for(int j=0;j<3;j++) //slots(morning....)

{

covaxSlots[i][j]=c;

coviShieldSlots[i][j]=c;

}

}

bedsMatrix = d;

}

Centre(){}

bool checkCovaxSlots(int i,int j);

vector<int> bookings;

bool checkCoviSheildSlots(int i,int j);

int getcovaxInventory();

int getcoviShieldInventory();

void bookCovax(int uid,int i,int j);

void bookCoviShield(int uid,int i,int j);

int computeSafetyQuotient(vector<vector<int>> bedsMatrix) const;

int getSafetyQuotient() const;

bool operator > (const Centre& c) const {

if(getSafetyQuotient() == c.getSafetyQuotient()) {

return bookings.size() < c.bookings.size();

}

return getSafetyQuotient() > c.getSafetyQuotient();

}

};

bool Centre::checkCovaxSlots(int i,int j)

{

if(covaxSlots[i][j]>0&&covaxInventory>0)

return false; //slots availaible

return true; //not available

}

bool Centre::checkCoviSheildSlots(int i,int j)

{

if(coviShieldSlots[i][j]>0&&coviShieldInventory>0)

return false;

return true;

}

int Centre::getcovaxInventory()

{

return covaxInventory;

}

int Centre::getcoviShieldInventory()

{

return coviShieldInventory;

}

void Centre::bookCovax(int uid,int i,int j)

{

covaxInventory--;

bookings.push\_back(uid);

covaxSlots[i][j]--;

}

void Centre::bookCoviShield(int uid,int i,int j)

{

coviShieldInventory--;

bookings.push\_back(uid);

coviShieldSlots[i][j]--;

}

int Centre::computeSafetyQuotient(vector<vector<int>> bedsMatrix) const

{

int a[4][2] = {{-1,0},{1,0},{0,1},{0,-1}}; //bfs

int n = bedsMatrix.size();

int m = bedsMatrix[0].size();

bool flag = false;

queue<pair<int,int>> pendingCells;

int time\_stamp = 0;

for(int i=0; i<n; i++)

for(int j=0; j<m; j++)

if(bedsMatrix[i][j]==2)

pendingCells.push({i,j}); //first we pushed locations where [i][j] ==2 as the are more prone

pendingCells.push({-1,-1}); // to denote one completion round

while(1){

pair<int,int> cur\_cell = pendingCells.front(); //fn of queue

if(cur\_cell.first==-1 && cur\_cell.second==-1)

break;

pendingCells.pop();

for(int i = 0; i < 4; i++){

int newX = cur\_cell.first + a[i][0]; //to move adjacent to x cells

int newY = cur\_cell.second + a[i][1];

if(newX>=0 && newX<n && newY>=0 && newY<m /\* checked if indices are valid and within dimensions \*/

&& bedsMatrix[newX][newY]==1) {

flag = true;

pendingCells.push({newX,newY});

bedsMatrix[newX][newY]=2; //marking infected

}

}

cur\_cell = pendingCells.front();

if(cur\_cell.first==-1 && cur\_cell.second==-1){

if(flag==true)

{

time\_stamp++;

flag=false;

}

pendingCells.pop();

pendingCells.push({-1,-1});

}

}

for(int i=0; i<n; i++)

for(int j=0; j<m; j++)

if(bedsMatrix[i][j]==1)

return -1;

return time\_stamp;

}

int Centre::getSafetyQuotient() const {

return computeSafetyQuotient(bedsMatrix);

}

/\* ########################### MAIN METHOD ##############################\*/

vector<User> users; //users type vector

map<string, Centre> centres; //string to centre map

void intro();

void timing\_details();

void addUser();

void addCentre();

void bookslot();

void getstats();

void getCentres();

int main()

{

intro();

system("cls");

while(1)

{

//system("cls");

cout<<"\n\n\n \t";

for(int i=1;i<43;i++)

{

cout<<"=";

}

cout<<"\n\t\t||\t FEATURES ||\n";

cout<<" \t";

for(int i=1;i<43;i++)

{

cout<<"=";

}

cout<<"\n\n\t\tPress 1 for Vaccination Timings";

cout<<"\n\n\t\tPress 2 for Adding a Recovery Centre";

cout<<"\n\n\t\tPress 3 for Adding a User";

cout<<"\n\n\t\tPress 4 for Get Centres";

cout<<"\n\n\t\tPress 5 for Booking a Slot";

cout<<"\n\n\t\tPress 6 for Getting Stats";

cout<<"\n\n\t\tPress 7 for exiting\n";

cout<<"\n\n\t\tEnter choice: ";

int ch;

cin>>ch;

//system("cls");

if(ch==1)

timing\_details();

else if(ch==2)

addCentre();

else if(ch==3)

addUser();

else if(ch==4)

getCentres();

else if(ch==5)

bookslot();

else if(ch==6)

getstats();

else

{

system("cls");

cout<<"\n\n\t Thanks for Visiting"<<endl;

cout<<"\n\n\tJaan Hai Toh Jahan Hai!!";

break;

}

}

}

void getstats()

{

system("cls");

map<string, Centre>::iterator it;

for (it = centres.begin(); it != centres.end(); it++)

{

cout<<"\n\n Recovery Center : "<<it->first<<endl;

Centre center = it->second;

cout<<"\n Covax available : "<<center.getcovaxInventory();

cout<<"\n Covishield available : "<<center.getcoviShieldInventory();

cout<<"\n Users : \n";

for(int i=0;i<center.bookings.size();i++) // to display name of all users

{

cout<<users[center.bookings[i]].getName()<<endl;

}

}

cin.get();

//system("cls");

}

void addUser()

{

system("cls");

string name,pvt;

char gender;

cout<<"\n Enter Name : ";

cin>>name;

cout<<"\n Enter Gender(M/F/T/O) : ";

cin>>gender;

while(1)

{

cout<<"\n Enter Preferred Vaccine Type (COVAX or COVISHIELD) : ";

cin>>pvt;

if(pvt == "COVISHIELD" || pvt=="COVAX" || pvt=="covax" || pvt=="covishield" || pvt=="Covishield" || pvt=="Covax")

break;

cout<<"\n Invalid Input! Try again \n";

}

int id = users.size()+1;

User user = User(name,gender,pvt);

users.push\_back(user);

cout<<"\n User Registered with ID : "<<id<<endl;

//system("cls");

}

void addCentre()

{

system("cls");

string cid;

int covax,covishield,maxCapacity;

while(1)

{

cout<<"\n Enter Center ID : ";

cin>>cid;

if(centres.find(cid)!=centres.end())

{

cout<<"\n Center ID already exist! Try again.\n";

}

else

{

break;

}

}

cout<<"\n Enter COVAX inventory available : ";

cin>>covax;

cout<<"\n Enter COVISHIELD inventory available : ";

cin>>covishield;

cout<<"\n Enter Max Capacity Per Slot : ";

cin>>maxCapacity;

int R;int C;

cout<<"\n Enter dimensions of Recovery Center (m x n) : "; //2d matrix (2,3)

cin>>R>>C;

vector<vector<int>> bedsMatrix;

int i,j;

cout<<"\n Enter m x n values from (0,1,2) such that : ";

cout<<"\n 0: Empty ward";

cout<<"\n 1: Wards have uninfected patients";

cout<<"\n 2: Wards have infected patients\n\n ";

for(i=0;i<R;i++)

{

vector<int> temp;

for(j=0;j<C;j++)

{

int k;

cin>>k;

temp.push\_back(k);

}

bedsMatrix.push\_back(temp);

}

Centre center = Centre(cid,covax,covishield,maxCapacity,bedsMatrix);

centres.insert({cid,center});

//system("cls");

}

void bookslot()

{

system("cls");

int uid;

string cid;

cout<<"\n Enter UserID : ";

cin>>uid;

if(uid>users.size())

{

cout<<"\n Invalid ID";

return;

}

cout<<"\n Enter CenterID : ";

cin>>cid;

if(centres.find(cid)==centres.end()) //used find fn to return iterator to the object

{

cout<<"\n Invalid ID";

return;

}

uid--;

int bookingDate = users[uid].getBookings(); //invoked getbookings using user[uid]

if(!users[uid].canBook) //if false canbook

{

cout<<"\n Cant Book More Slots !";

return;

}

cout<<"\n Enter booking date : ";

int date;

cin>>date;

if(date-bookingDate<10&&bookingDate!=-1) //to check if user can book date having a minimum gap of 10 days

{

cout<<"\n Minimun gap should be 10!";

return;

}

int slot;

while(1)

{

cout<<"\n Choose slot (1 for morning, 2 for afternoon and 3 for evening) : ";

cin>>slot;

if(slot>0&&slot<=3)

break;

cout<<"\n Invalid Input. try again.";

}

Centre center = centres.find(cid)->second;//to find if center id is valid

string pvt = users[uid].getPreferredVaccineType();

if(pvt == "COVAX" || pvt=="covax" || pvt=="Covax")

{

if(center.getcovaxInventory()<=0)

{

cout<<"\n Covax Not available";

return;

}

if(center.checkCovaxSlots(date-1,slot-1)) //check n 2d array slots (boolean)

{

cout<<"\n Slot not available.";

return;

}

else //booking will be made

{

center.bookCovax(uid,date-1,slot-1);

if(bookingDate==-1)

users[uid].addBooking(date);

else

users[uid].canBook = false;

}

}

else

{

if(center.getcoviShieldInventory()<=0)

{

cout<<"\n Covishield Not available";

return;

}

if(center.checkCoviSheildSlots(date-1,slot-1))

{

cout<<"\n Slot not available.";

return;

}

else

{

center.bookCoviShield(uid,date-1,slot-1);

if(bookingDate==-1)

users[uid].addBooking(date);

else

users[uid].canBook = false;

}

}

centres[cid]=center;

}

//bool compare(Centre const c1, Centre const c2) {

// return c1.getSafetyQuotient() > c2.getSafetyQuotient();

//}

// Displays centres where user preferred vaccine type is available in decreasing order of safetyQuotient

// If SafetyQuotient of two centres is same, then priority is given to centres having lesser no. of bookings

void getCentres()

{

int uid;

cout<<"\n Enter UserID : ";

cin>>uid;

if(uid-1 >= users.size())

{

cout<<"\n Invalid ID, Enter again!!!";

return;

}

string pvt = users[uid-1].getPreferredVaccineType();

map<string, Centre>::iterator it;

vector<Centre> userCenters;

//fetching all centres having user's preferred vaccine

for (it = centres.begin(); it != centres.end(); it++) {

Centre center = it->second;

transform(pvt.begin(), pvt.end(), pvt.begin(), ::toupper); //to push those cenrers in which vaccine is available

if((pvt.compare("COVAX") == 0 && center.getcovaxInventory() > 0) || (pvt.compare("COVISHIELD") == 0 && center.getcoviShieldInventory() > 0)) {

userCenters.push\_back(center);

}

}

sort(userCenters.begin(), userCenters.end(), greater<Centre>()); //here > operator is used for sorting here > condition also implemented

cout<<"\n######## DISPLAYING AVAILABLE CENTRES AROUND YOU ########\n";

cout<<"\n CENTER ID \t Safety Quotient";

for(int i = 0; i < userCenters.size(); i++) {

cout<<"\n "<<userCenters[i].cid << "\t\t\t" <<userCenters[i].getSafetyQuotient();

}

cin.get();

}

void timing\_details()

{

for(int i=1;i<54;i++)

{

cout<<"=";

}

cout<<"\n| VACCINATION TIMINGS |\n";

for(int i=1;i<54;i++)

{

cout<<"=";

}

cout<<"\n|->->->->->TIMING : 10.00 AM TO 05.00 PM<-<-<-<-<-<-|";

cout<<"\n| |";

cout<<"\n| # STAY SAFE STAY VACCINATED |";

cout<<"\n| |";

cout<<"\n| # GAP BETWEEN 1ST DOSE AND 2ND DOSE IS 60 DAYS |";

cout<<"\n| |\n";

for(int i=1;i<54;i++)

{

cout<<"=";

}

}

void intro()

{

cout << "\n\n\n \t";

for(int i=1;i<59;i++)

{

cout<<"=";

}

cout << "\n\t\t||\t Welcome to Vaccination Portal ||\n";

cout << " \t";

for(int i=1;i<59;i++)

{

cout<<"=";

}

cout << "\n\t\t\t|| By: Sarthak, Manya, Aditya, Aryan ||\n";

cout<<"\t\t\t";

for(int i=1;i<44;i++)

{

cout<<"=";

}

cout << "\n\t\t\t|| Enrollment No.:21104001, 21104011, 21104018, 21104020 ||\n";

cout<<"\t\t\t";

for(int i=1;i<44;i++)

{

cout<<"=";

}

cout << "\n\t\t\t||\t Batch:B14 ||\n";

cout<<"\t\t\t";

for(int i=1;i<44;i++)

{

cout<<"=";

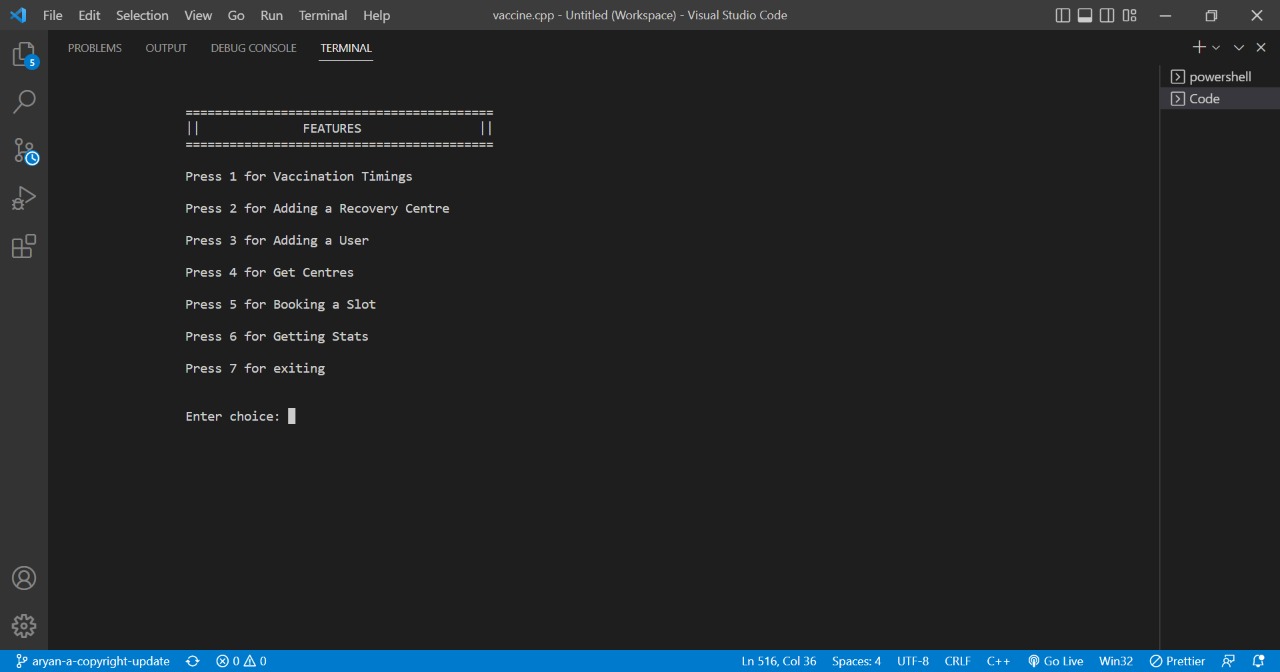
}

cin.get();

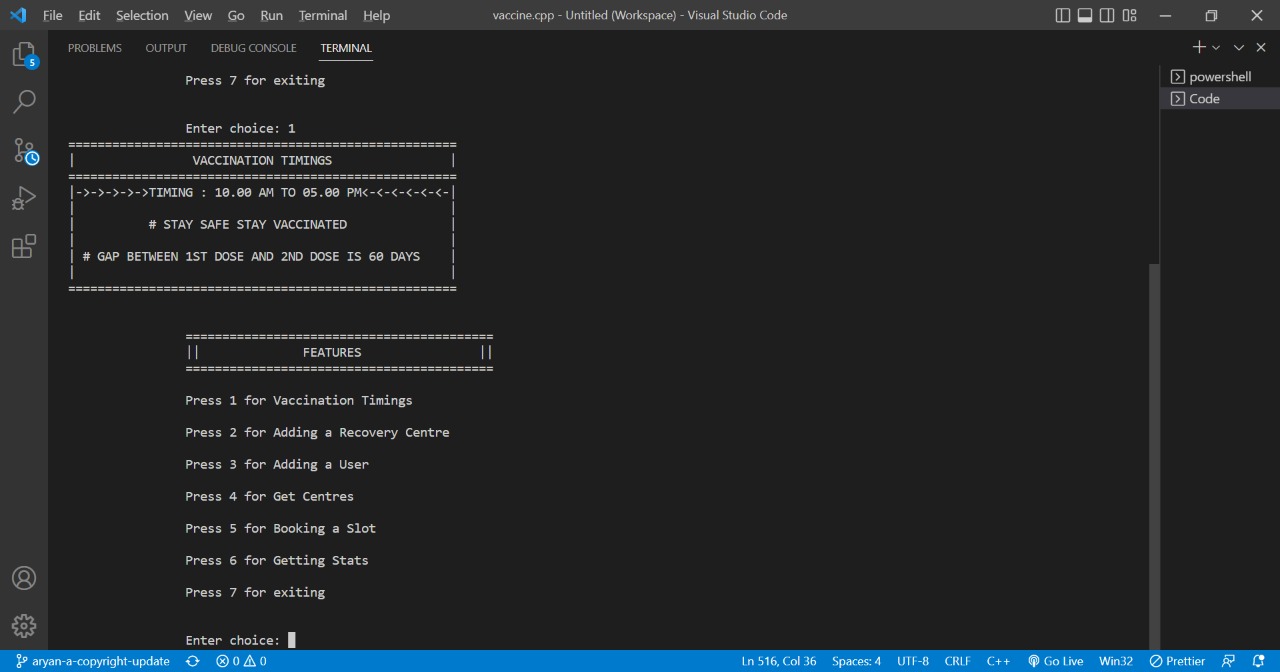
}

**Output:**

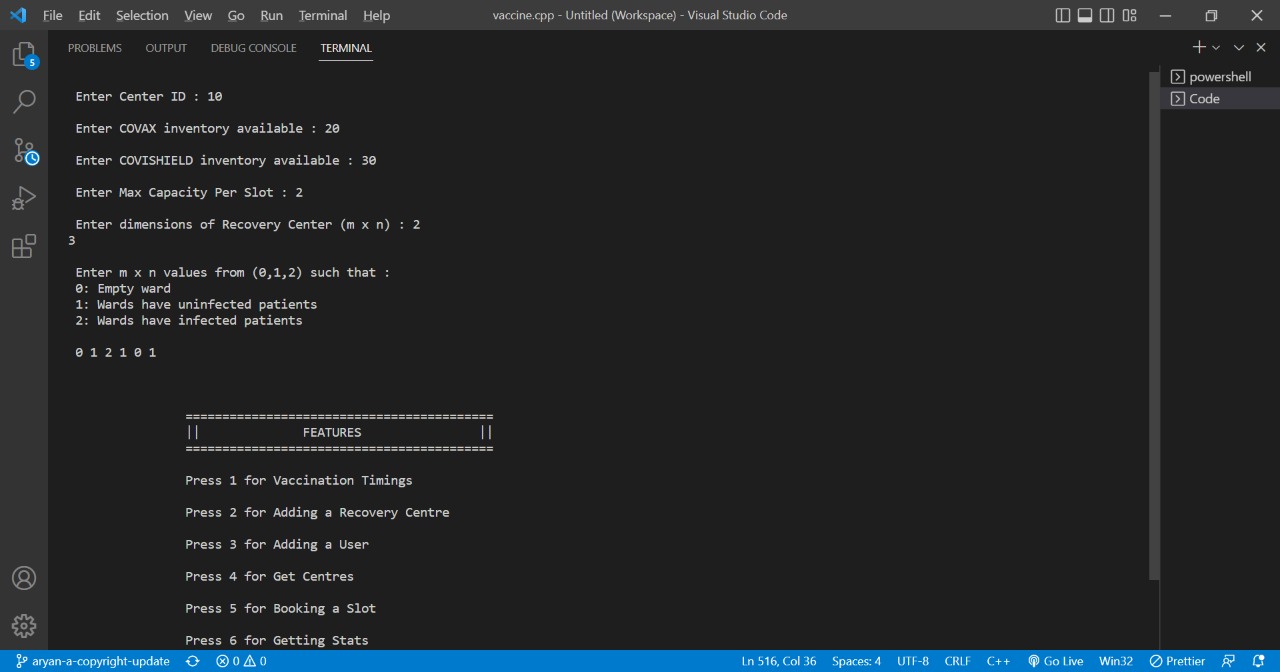
1. Menu:



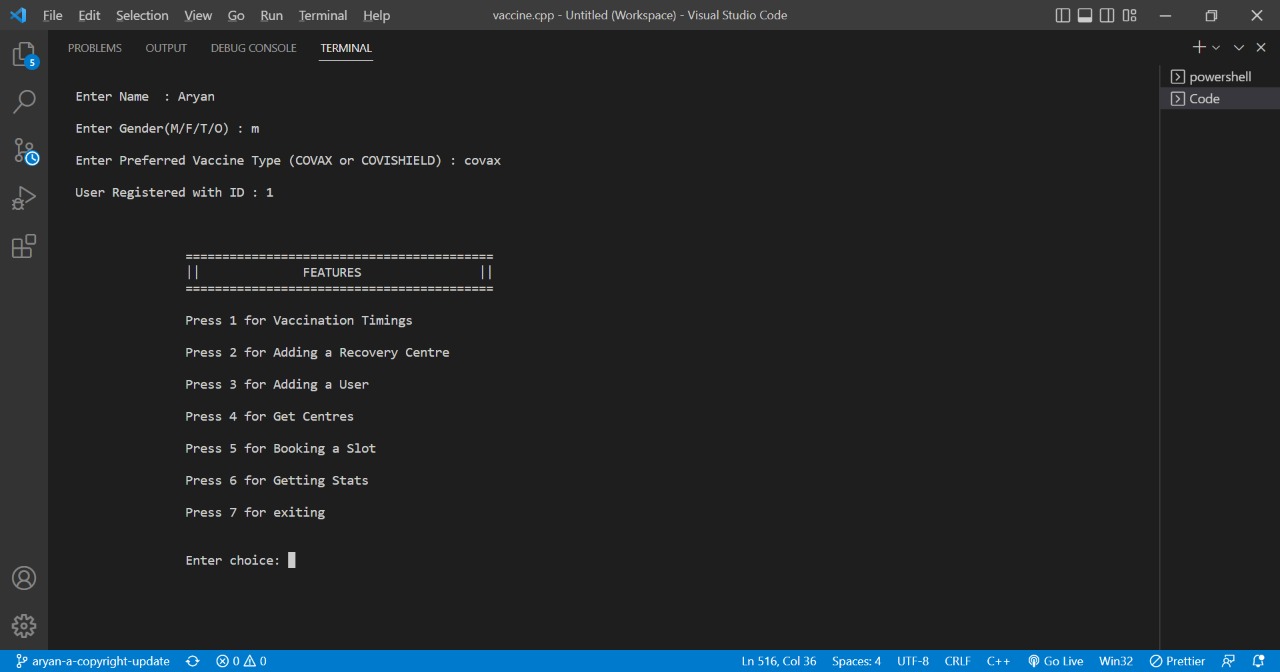
2)Vaccination Timings:



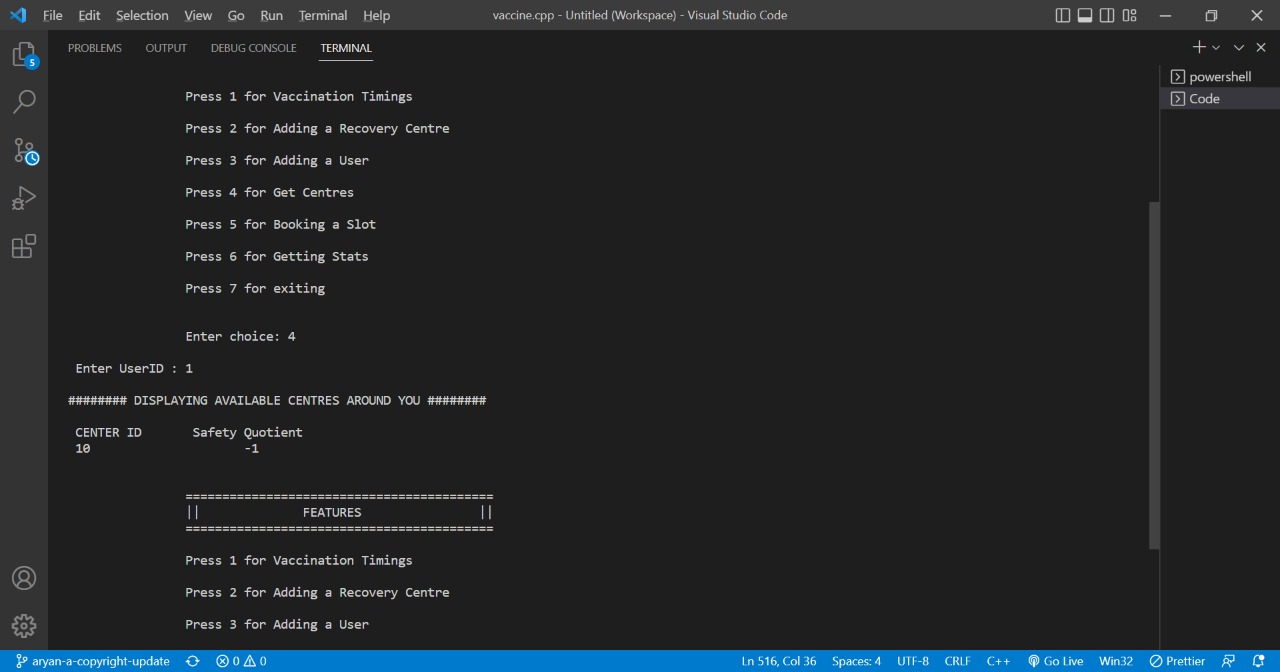
3) Adding a Centre:



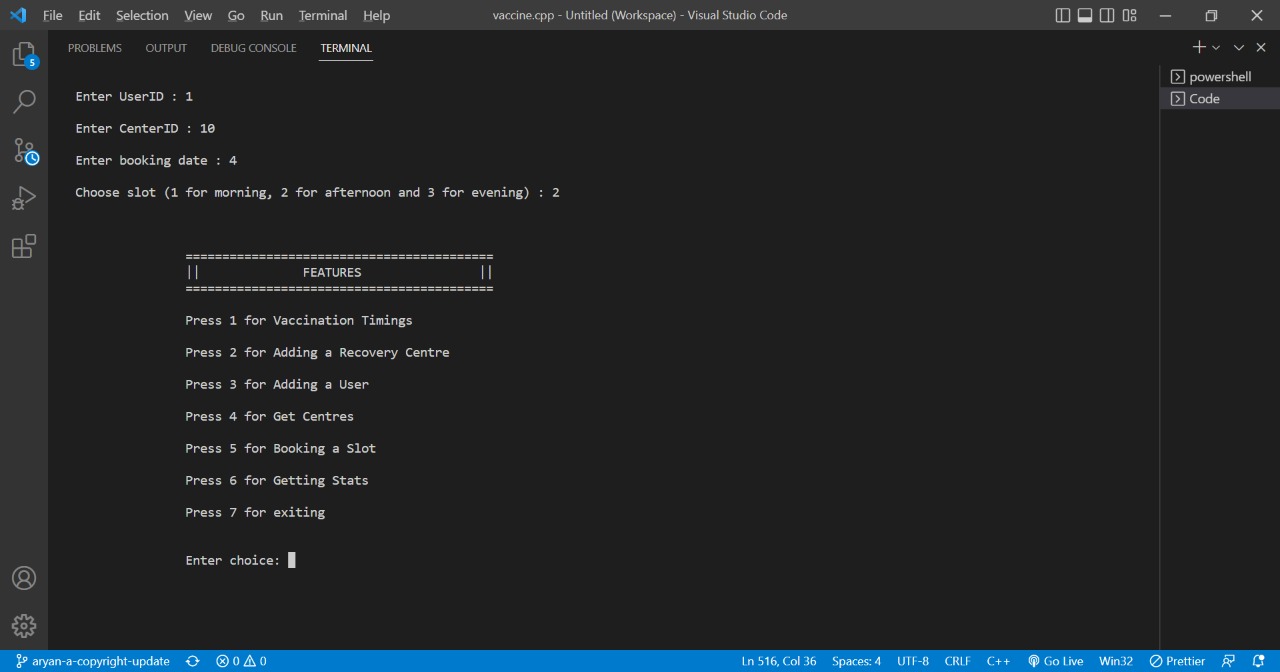
4) Adding a User:



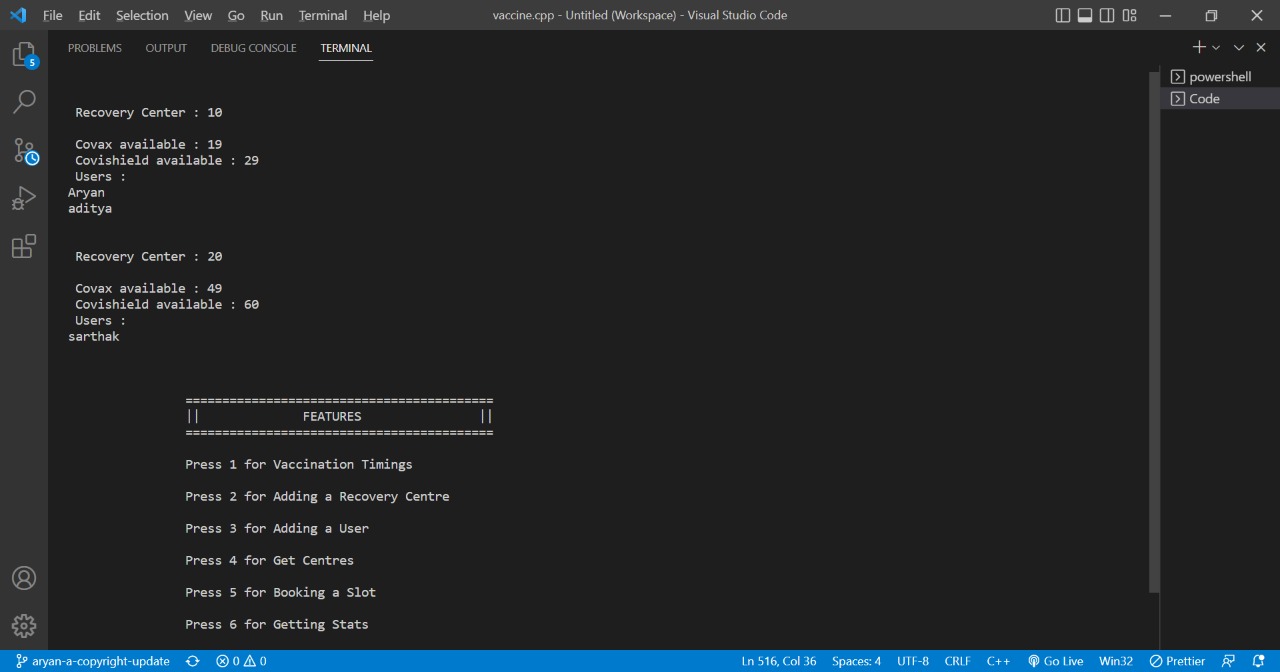
5) Get Nearby Centres:



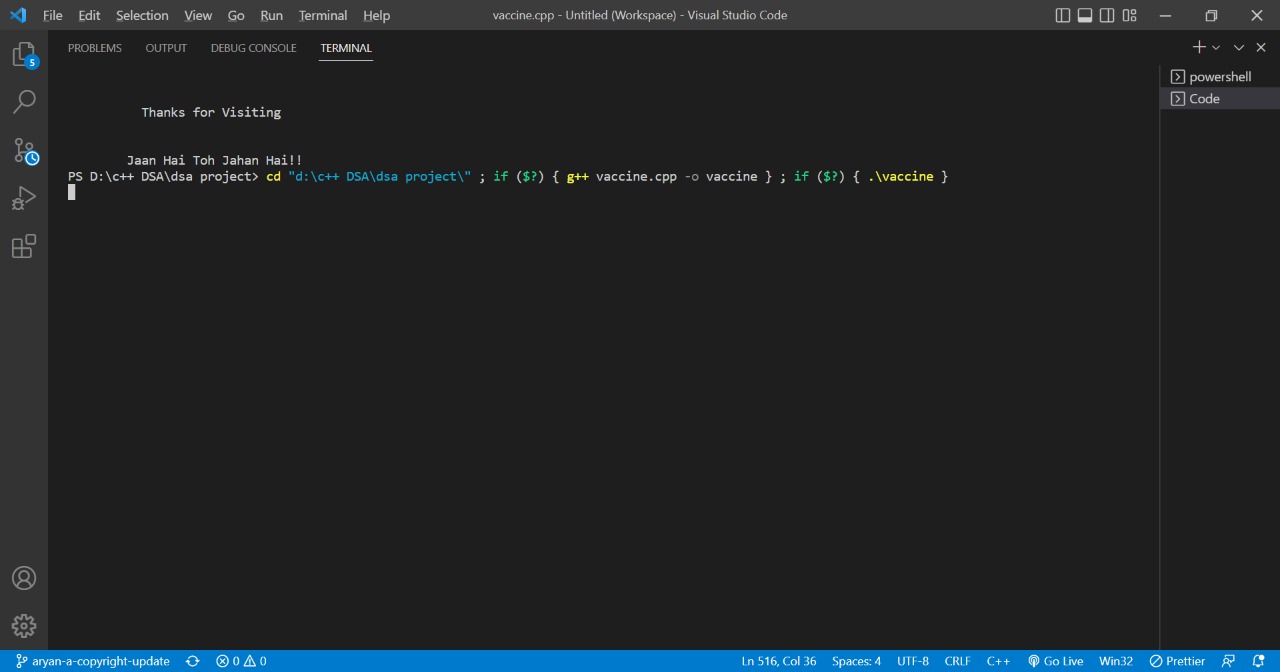
6) Booking Slot:



7) Getting Stats:



8) Exit:



**Conclusion:**

This data structure and algorithm project which uses the principles of Graph, Map, Queue, Oops and Vector namely ‘Vaccine Management System’ can be utilized by users and health care staff equivalently. Individuals can register their details and plan their testing time slot on the User Module side of the system. Previously the task of handling the vaccination data was very difficult, so there was a need for software that can handle all the vaccination data. Therefore the Vaccine Management System was designed. After the release of this system, the stress and workload of employees were absolutely finished. It was also time-wasting for the employees to handle the whole vaccination data with the help of a notebook. But now it hardly takes 5 to 10 minutes to search the vaccination status of a particular person.

**References:**

<https://www.geeksforgeeks.org/>

<https://www.javatpoint.com/>

<https://www.geeksforgeeks.org/the-c-standard-template-library-stl/>

<https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/>