Assignment #3

Q1

Code:

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

class User;

template <typename T>

class DataHolder{

    T data;

    public:

    DataHolder(T d):data(d){}

    void setData(T d){data=d;}

    T getData(){return data;}

};

template <typename T>

User\* findUser(User\*\* users,int count,T value,T(User::\*getter)()const)

{

    for (int i = 0; i < count; i++)

    {

        if ((users[i]->\*getter)()==value)

        {

            return users[i];

        }

    }

    return nullptr;

};

template <typename T>

void addToDynamicArray(T\*& arr,int& count,int& capacity,T item){

    if (count>=capacity)

    {

        capacity\*=2;

        T\* temp=new T[capacity];

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

        {

            temp[i]=arr[i];

        }

        delete[] arr;

        arr=temp;

    }

    arr[count]=item;

    count++;

};

class User

{

protected:

    string name;

    string id;

    string location;

    bool activeCard;

    bool isLogIn;

public:

    User(string n, string i, string l) : name(n), id(i), location(l), activeCard(false) {}

    virtual void payFee() = 0;

    virtual void logIn() = 0;

    virtual ~User() {}

    bool isCardActive()const{return activeCard;}

    string getLocation()const{return location;}

    string getName()const{return name;}

    string getId()const{return id;}

};

class Student : public User

{

    int attendanceCount;

    string password;

public:

    Student(string n, string i, string l, string p) : User(n, i, l), attendanceCount(0), password(p) {}

    void logIn()

    {

        string pass;

        cout << "\n Welcome Student : " << name;

        cout << "\n Enter Password To Login : ";

        getline(cin, pass);

        if (pass == password)

        {

            isLogIn = true;

            cout << "\n Login successfully ";

        }

        else

        {

            cout << "\n Credential Failed";

        }

    }

    void payFee()

    {

        if (isLogIn)

        {

            activeCard = true;

            cout << "\n Student Pay per semester";

        }

        else

        {

            cout << "\n Log In Required";

        }

    }

    void tap()

    {

        if (isLogIn)

        {

            if (activeCard)

                attendanceCount++;

        }

        else

        {

            cout << "\n Log In Required";

        }

    }

};

class Teacher : public User

{

    int pin;

public:

    Teacher(string n, string i, string l, int p) : User(n, i, l), pin(p) {}

    void logIn()

    {

        int p;

        cout << "\n Welcome Teacher : " << name;

        cout << "\n Enter Your pin to LogIn  : ";

        cin >> p;

        cin.ignore();

        if (p == pin)

        {

            isLogIn = true;

            cout << "\n Log In Successfully ";

        }

        else

        {

            cout << "\n Invalid credentials";

        }

    }

    void payFee()

    {

        if (isLogIn)

        {

            cout << "\n Teacher Pay per month";

            activeCard = true;

        }

        else

        {

            cout << "\n Log In Required";

        }

    }

};

class Staff : public User

{

    int otp;

public:

    Staff(string n, string i, string l, int o) : User(n, i, l), otp(o) {}

    void logIn()

    {

        int o;

        cout << "\n Welcome Staff " << name;

        cout << "\n Enter opt to LogIn : ";

        cin >> o;

        cin.ignore();

        if (o == otp)

        {

            isLogIn = true;

            cout << "\n Log In Successfully ";

        }

        else

        {

            cout << "\n Invalid credentials";

        }

    }

    void payFee()

    {

        if (isLogIn)

        {

            cout << "\n Staff Pay per month";

            activeCard = true;

        }

        else

        {

            cout << "\n Log In Required";

        }

    }

};

class Route{

    int routeNumber;

    string stop;

    public:

    Route(int r, string s) : routeNumber(r), stop(s) {}

    string getStop() { return stop; }

    int getRouteNumber() { return routeNumber; }

    bool operator==(Route& r){return routeNumber==r.routeNumber && stop==r.stop;}

};

class Routes{

    Route\*\* routes;

    int routeCount;

    int routeCapacity;

    User\*\* users;

    int userCount;

    int userCapacity;

    public:

    Routes():userCount(0),routeCount(0),userCapacity(10),routeCapacity(10){

        routes=new Route\*[routeCapacity];

        users=new User\*[userCapacity];

    }

    ~Routes() {

        for (int i = 0; i < routeCount; i++) delete routes[i];

        delete[] routes;

        for (int i = 0; i < userCount; i++) delete users[i];

        delete[] users;

    }

    User\*\* getUsers() { return users; }

    int getUserCount() { return userCount; }

    void saveUserToFile(const string &fileName){

        ofstream us(fileName);

        if(!us)

        throw runtime\_error(" \n Failed to open file : "+fileName);

        us<<userCount<<endl;

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

        {

            us <<  users[i]->getName() << " " << users[i]->getId() << " " << users[i]->getLocation() << endl;

        }

        us.close();

    }

    void saveRoutesToFile(const string &filename) {

        ofstream fout(filename);

        if (!fout) throw runtime\_error("Failed to open file for writing: " + filename);

        fout << routeCount << endl;

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

            fout << routes[i]->getRouteNumber() << " " << routes[i]->getStop() << endl;

        }

        fout.close();

    }

    void addUser(User\* u,const string &filename){

        addToDynamicArray(users,userCount,userCapacity,u);

        cout<<"\n User added successfully";

        saveUserToFile(filename);

    }

    void addRoute(Route\* r,const string &filename) {

        addToDynamicArray(routes,routeCount,routeCapacity,r);

        cout<<"\n Route Added ";

        saveRoutesToFile(filename);

    }

    void assignRoutes() {

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

            for (int j = 0; j < userCount; j++) {

                if (users[j]->isCardActive() && users[j]->getLocation() == routes[i]->getStop()) {

                    cout << "\n" << users[j]->getName() << " assigned to Route " << routes[i]->getRouteNumber();

                }

            }

        }

    }

};

int main() {

    Routes system;

    string userFile = "users.txt";

    string routeFile = "routes.txt";

    while (true) {

        cout << "\n\n====== FAST TRANSPORT SYSTEM ======";

        cout << "\n1. Add Student";

        cout << "\n2. Add Teacher";

        cout << "\n3. Add Staff";

        cout << "\n4. Add Route";

        cout << "\n5. Assign Routes";

        cout << "\n6. Find User";

        cout << "\n7. Exit";

        cout << "\nChoose an option: ";

        int choice;

        cin >> choice;

        cin.ignore();

        try {

            if (choice == 1) {

                string name, id, location, password;

                cout << "Enter Student Name: ";

                getline(cin, name);

                cout << "Enter ID: ";

                getline(cin, id);

                cout << "Enter Location: ";

                getline(cin, location);

                cout << "Enter Password: ";

                getline(cin, password);

                system.addUser(new Student(name, id, location, password), userFile);

            }

            else if (choice == 2) {

                string name, id, location;

                int pin;

                cout << "Enter Teacher Name: ";

                getline(cin, name);

                cout << "Enter ID: ";

                getline(cin, id);

                cout << "Enter Location: ";

                getline(cin, location);

                cout << "Enter PIN: ";

                cin >> pin;

                cin.ignore();

                system.addUser(new Teacher(name, id, location, pin), userFile);

            }

            else if (choice == 3) {

                string name, id, location;

                int otp;

                cout << "Enter Staff Name: ";

                getline(cin, name);

                cout << "Enter ID: ";

                getline(cin, id);

                cout << "Enter Location: ";

                getline(cin, location);

                cout << "Enter OTP: ";

                cin >> otp;

                cin.ignore();

                system.addUser(new Staff(name, id, location, otp), userFile);

            }

            else if (choice == 4) {

                int routeNumber;

                string stop;

                cout << "Enter Route Number: ";

                cin >> routeNumber;

                cin.ignore();

                cout << "Enter Stop Name: ";

                getline(cin, stop);

                system.addRoute(new Route(routeNumber, stop), routeFile);

            }

            else if (choice == 5) {

                system.assignRoutes();

            }else if (choice==6)

            {

                string name;

                cout<<"\n Enter Name To Find : ";

                getline(cin,name);

                User\* u = findUser(system.getUsers(), system.getUserCount(), name, &User::getName);

                if (u!=nullptr)

                {

                    cout << "\nFound: " << u->getName() << " at " << u->getLocation();

                }else{

                     cout << "\nUser not found.";

                }

            }

            else if (choice == 7) {

                cout << "\nExiting...\n";

                break;

            }

            else {

                cout << "\nInvalid Option. Try Again.";

            }

        } catch (exception &e) {

            cerr << "\nException: " << e.what() << endl;

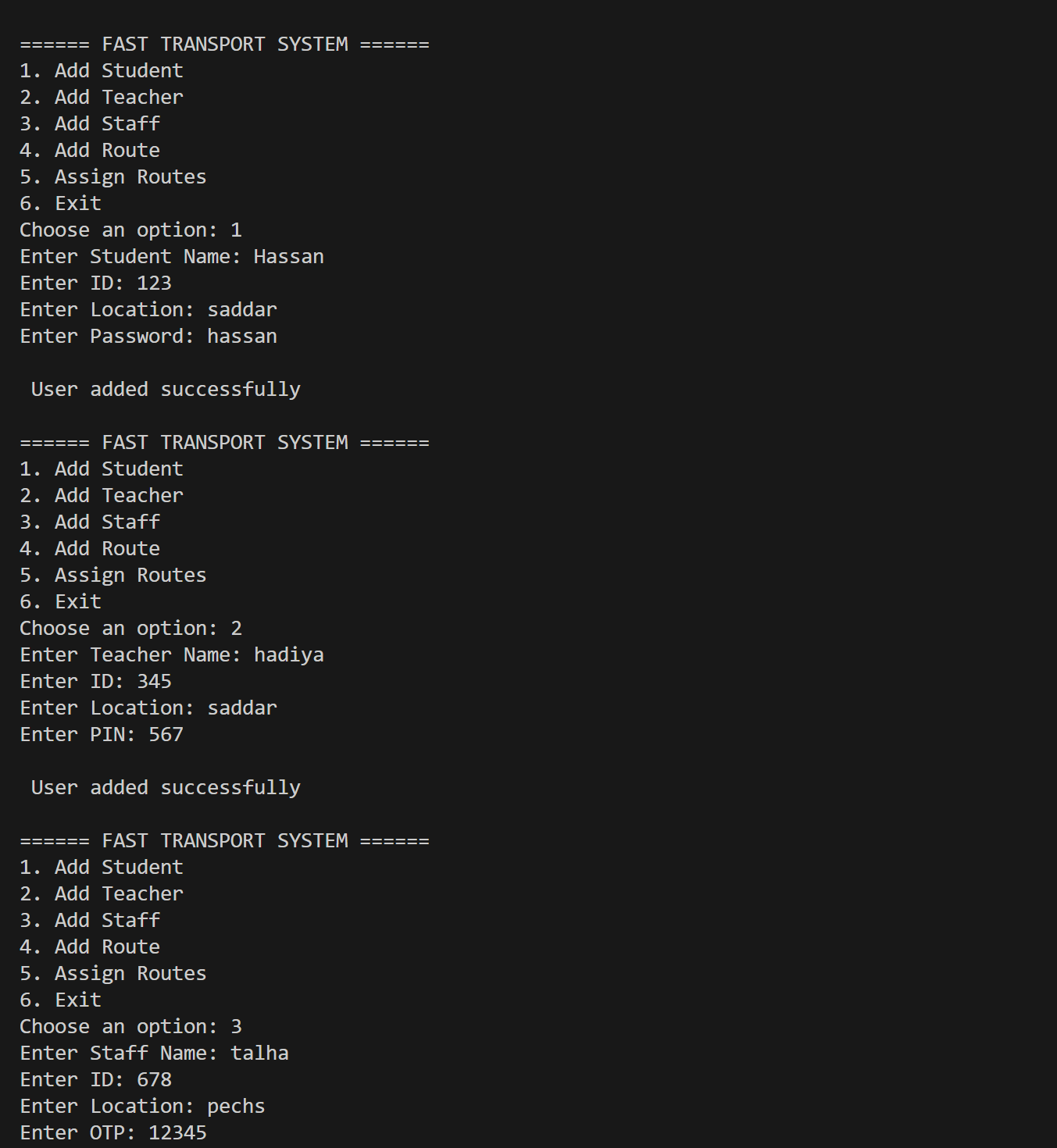
        }

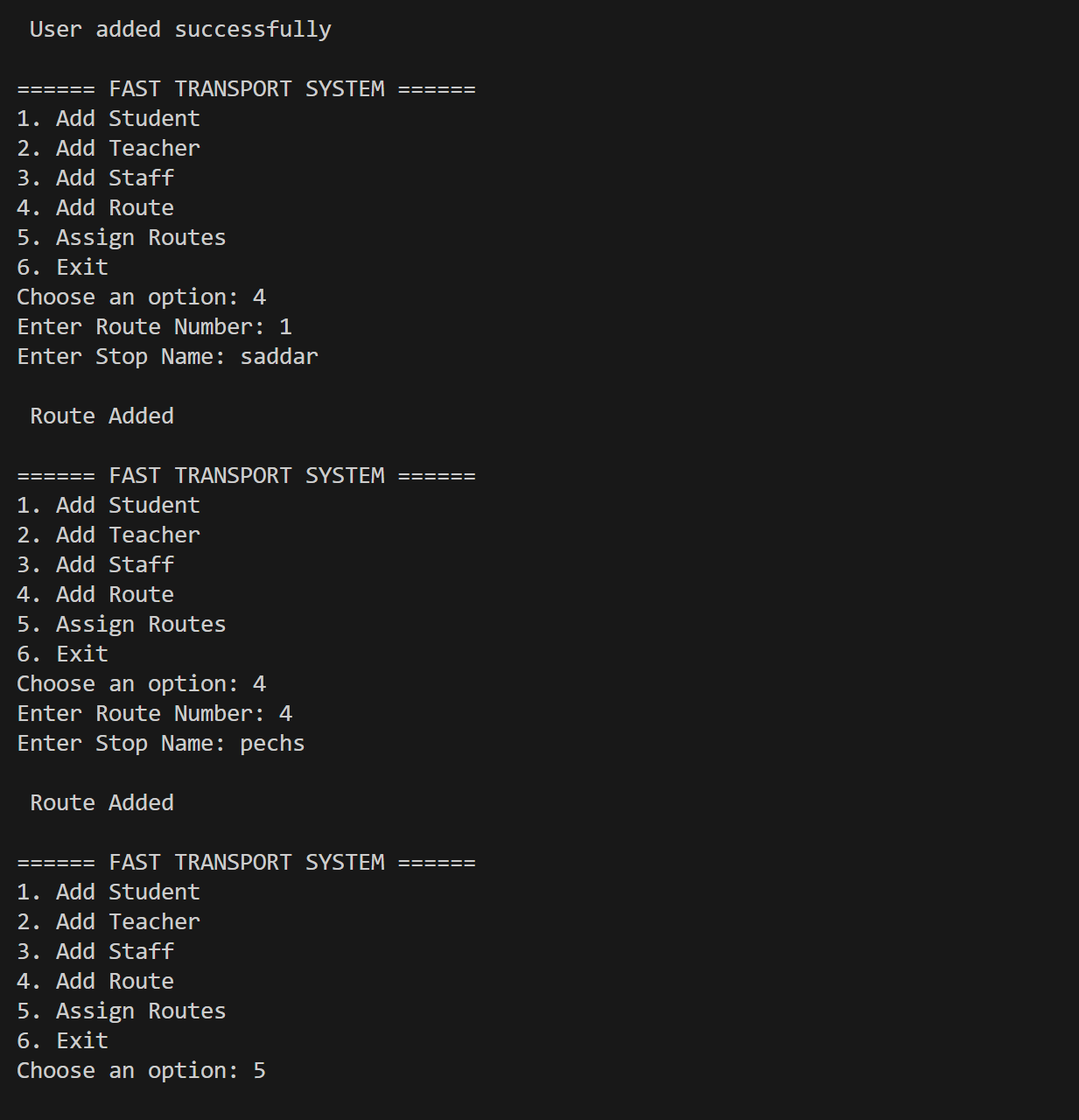
    }

    return 0;

}

Output:





A screenshot of a computer program

AI-generated content may be incorrect.

User.txt:

A screenshot of a computer

AI-generated content may be incorrect.

Routes.txt:

