OOPs Project (IE2)

# Input:

// Project Title: Botanical Garden Management System, Project Head Name: Swetank Kalyan Raut, Date of Completion: 27-10-23

/\*

\*Project Description:

    This project is a Botanical Garden Management System. It provides functionalities for adding, displaying, searching for, and deleting plant data in a garden. Users can also manage and update plant types and habitats in the garden. The program reads and writes data to a CSV file, which serves as a database for storing plant information. It allows users to interact with the garden's data and make updates as needed, providing a basic system for botanical garden management.

\*/

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

using namespace std;

class BotanicalG

{

private:

    vector<string> habitats;

    vector<string> plantTypes;

    string commonName;

    string scientificName;

    string lifeSpan;

    long PlantID;

public:

    BotanicalG(fstream &stream)

    {

        stream.seekg(0, ios::end);

        if (stream.tellg() == 0)

        {

            // The file is empty, so you can add your header and other data.

            stream.clear(); // Clear any error flags to write to an empty file

            stream << "Plant ID,"

                   << "Common Name,"

                   << "Scientific Name,"

                   << "Type of Plant,"

                   << "Habitat,"

                   << "Life Span\n";

        }

        else

        {

            // Move the file pointer back to the beginning of the file

            stream.seekg(0, ios::beg);

        }

    }

    void addAPlant(fstream &file)

    {

        string st;

        long ID;

        int noOfSpecies, n, r;

        cout << "Enter Common Name: ";

        cin.ignore();

        getline(cin, commonName);

        cout << "Enter Scientific Name: ";

        cin.ignore();

        getline(cin, scientificName);

        while (true)

        {

        go1:

            try

            {

                displayPlantTypes();

                cout << "Choice Type of Plant: ";

                cin >> st;

                n = stol(st); // Convert the string

            }

            catch (const invalid\_argument &e)

            {

                cerr << "Invalid input. Please enter a valid numeric value for Plant type." << endl;

                goto go1;

            }

            if (n <= plantTypes.size() && n > 0)

            {

            go2:

                try

                {

                    displayHabitats();

                    cout << "Choice Habitat by no: ";

                    cin >> st;

                    r = stol(st); // Convert the string

                }

                catch (const invalid\_argument &e)

                {

                    cerr << "Invalid input. Please enter a valid numeric value for Choice." << endl;

                    goto go2;

                }

                if (r <= habitats.size() && r > 0)

                    break;

                break;

            }

            else

                cout << "Enter a Valid choice!" << endl;

        }

        cout << "Enter Life Span of Plant: ";

        cin.ignore();

        getline(cin, lifeSpan);

        cout << "Enter, How many same Plant species are available?: ";

        cin >> noOfSpecies;

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

        {

        id:

            try

            {

                cout << "Enter Plant ID for plant " << i + 1 << ": ";

                cin >> st;

                ID = stol(st); // Convert the string to a long

            }

            catch (const invalid\_argument &e)

            {

                cerr << "Invalid input. Please enter a valid numeric value for Plant ID." << endl;

                goto id;

            }

            file << ID << ", " << commonName << ", " << scientificName

                 << ", " << plantTypes[n - 1] << ", " << habitats[r - 1] << ", " << lifeSpan << "\n";

        }

    }

    void searchAPlant(ifstream &file, long plantID)

    {

        string line;

        bool found = false;

        file.clear();

        while (getline(file, line))

        {

            vector<string> elements;

            elements.push\_back(line);

            if (elements.size() >= 1) // checks presence of line

            {

                try

                {

                    long id = stol(elements[0]);

                    if (id == plantID)

                    {

                        cout << "Plant found:" << endl;

                        cout << line << endl;

                        found = true;

                        return;

                    }

                }

                catch (const invalid\_argument &e)

                {

                }

            }

        }

        if (!found)

        {

            cout << "Plant not found." << endl;

        }

    }

    void deleteAPlantData(fstream &file, long plantID)

    {

        string line;

        vector<string> lines; // used as a vector to store line as an element

        bool found = false;

        file.clear();

        while (getline(file, line))

        {

            vector<string> elements;

            elements.push\_back(line);

            if (elements.size() >= 1)

            {

                try

                {

                    long id = stol(elements[0]);

                    if (id != plantID) // By doing this, it will skip that line only which has the matching ID

                    {

                        lines.push\_back(line);

                    }

                    else

                    {

                        found = true;

                    }

                }

                catch (const invalid\_argument &e)

                {

                }

            }

        }

        file.close();

        file.open("plant\_database.csv", ios::out | ios::trunc); // Previous data is deleted after using trunc

        file << "Plant ID,"

             << "Common Name,"

             << "Scientific Name,"

             << "Type of Plant,"

             << "Habitat,"

             << "Life Span\n";

        for (string &l : lines) // as lines have a copy of previous data, therefore copied in l

        {

            file << l << "\n";

        }

        if (!found)

        {

            cout << "Plant not found." << endl;

        }

        else

            cout << "Plant data deleted successfully." << endl;

    }

    void addHabitat()

    {

        string habitat;

        char choice;

        cout << "Enter The Available types of Habitat in Garden: " << endl;

        do

        {

            cout << "Enter: ";

            getline(cin, habitat);

            habitats.push\_back(habitat);

            cout << "Want to Continue(y/n): ";

            cin >> choice;

            cin.ignore();

        } while (choice == 'y' || choice == 'Y');

    }

    void addPlantType()

    {

        string plantType;

        char choice;

        cout << "Enter Required Types of Plant in Garden: " << endl;

        do

        {

            cout << "Enter: ";

            getline(cin, plantType);

            plantTypes.push\_back(plantType);

            cout << "Want to Continue(y/n): ";

            cin >> choice;

            cin.ignore();

        } while (choice == 'y' || choice == 'Y');

    }

    void update(double a)

    {

        int index;

        cout << "Enter the number of the plant type to update: ";

        cin >> index;

        if (index > 0 && index <= plantTypes.size())

        {

            string newPlantType;

            cout << "Enter the new plant type: ";

            cin.ignore();

            getline(cin, newPlantType);

            plantTypes[index - 1] = newPlantType;

            cout << "Plant type updated successfully.\n";

        }

        else

        {

            cout << "Invalid choice. Please select a valid option.\n";

        }

    }

    void update(int a)

    {

        int index;

        cout << "Enter the position number of the habitat to update: ";

        cin >> index;

        if (index > 0 && index <= habitats.size())

        {

            string newHabitat;

            cout << "Enter the new habitat: ";

            cin.ignore();

            getline(cin, newHabitat);

            habitats[index - 1] = newHabitat;

            cout << "Habitat updated successfully.\n";

        }

        else

        {

            cout << "Invalid choice. Please select a valid option.\n";

        }

    }

    void displayHabitats()

    {

        cout << "Habitats Menu:\n";

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

        {

            cout << i + 1 << ". " << habitats[i] << ", ";

        }

    }

    void displayPlantTypes()

    {

        cout << "Plant Types Menu:\n";

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

        {

            cout << i + 1 << ". " << plantTypes[i] << ", ";

        }

    }

};

class Display : public BotanicalG

{

    public:

    Display(fstream &File): BotanicalG(File) {}

    void displayWholeData(ifstream &file)

    {

        string line;

        while (getline(file, line))

        {

            cout << line << endl;

        }

    }

};

int main()

{

    int choice, a;

    string st;

go:

    fstream outFile("plant\_database.csv", ios::app);

    Display GData(outFile);

    GData.addHabitat();

    GData.addPlantType();

    outFile.close();

    while (true)

    {

        cout << "<-==========================================================================->" << endl;

        cout << "|| -:^\_^:- -:^\_^:- :: BOTANICAL GARDEN MANAGEMENT SYSTEM :: -:^\_^:- -:^\_^:- ||" << endl;

        cout << "<-==========================================================================->" << endl;

        cout << "\t\t\t1. Add a Plant" << endl;

        cout << "\t\t\t2. Display All Plants" << endl;

        cout << "\t\t\t3. Search for a Plant" << endl;

        cout << "\t\t\t4. Delete a Plant" << endl;

        cout << "\t\t\t5. Update a Plant Type" << endl;

        cout << "\t\t\t6. Update a Habitat" << endl;

        cout << "\t\t\t7. Clear All Data" << endl;

        cout << "\t\t\t8. Exit" << endl;

    back:

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

        cin >> st;

        try

        {

            choice = stol(st); // Convert the string to a value

        }

        catch (invalid\_argument &e)

        {

            cerr << "Invalid input. Please enter a valid numeric value for Menu Choice." << endl;

            goto back;

        }

        fstream outFile("plant\_database.csv", ios::app);

        ifstream inFile("plant\_database.csv");

        fstream inOutFile("plant\_database.csv", ios::in | ios::out);

        switch (choice)

        {

        case 1:

        {

            GData.addAPlant(outFile);

            break;

        }

        case 2:

        {

            GData.displayWholeData(inFile);

            break;

        }

        case 3:

        {

            a = 1;

            long searchID;

            while (a == 1)

            {

                try

                {

                    cout << "Enter the Plant ID to search: ";

                    cin >> st;

                    searchID = stol(st);

                    a = 0;

                }

                catch (const invalid\_argument &e)

                {

                    cerr << "Please Enter a Valid Plant ID." << endl;

                    a = 1;

                }

            }

            GData.searchAPlant(inFile, searchID);

            break;

        }

        case 4:

        {

            a = 1;

            long searchID;

            while (a == 1)

            {

                try

                {

                    cout << "Enter the Plant ID to delete: ";

                    cin >> st;

                    searchID = stol(st);

                    a = 0;

                }

                catch (const invalid\_argument &e)

                {

                    cerr << "Please Enter a Valid Plant ID." << endl;

                    a = 1;

                }

            }

            GData.deleteAPlantData(inOutFile, searchID);

            break;

        }

        case 5:

            GData.update(1.1);

            break;

        case 6:

            GData.update(1);

            break;

        case 7:

        {

            char ch;

            inOutFile.close();

            cout << "Do you Really want to delete all Data?(y/n)" << endl;

            cin >> ch;

            if (ch == 'y')

            {

                remove("plant\_database.csv");

                ofstream newFile("plant\_database.csv");

                newFile << "Plant ID,"

                        << "Common Name,"

                        << "Scientific Name,"

                        << "Type of Plant,"

                        << "Habitat,"

                        << "Life Span\n";

                newFile.close();

                cout << "All data cleared." << endl;

                cin.ignore();

                goto go;

            }

            break;

        }

        case 8:

            cout << "Exiting the program." << endl;

            return 0;

        default:

            cout << "Invalid choice. Please enter a valid option." << endl;

            break;

        }

    }

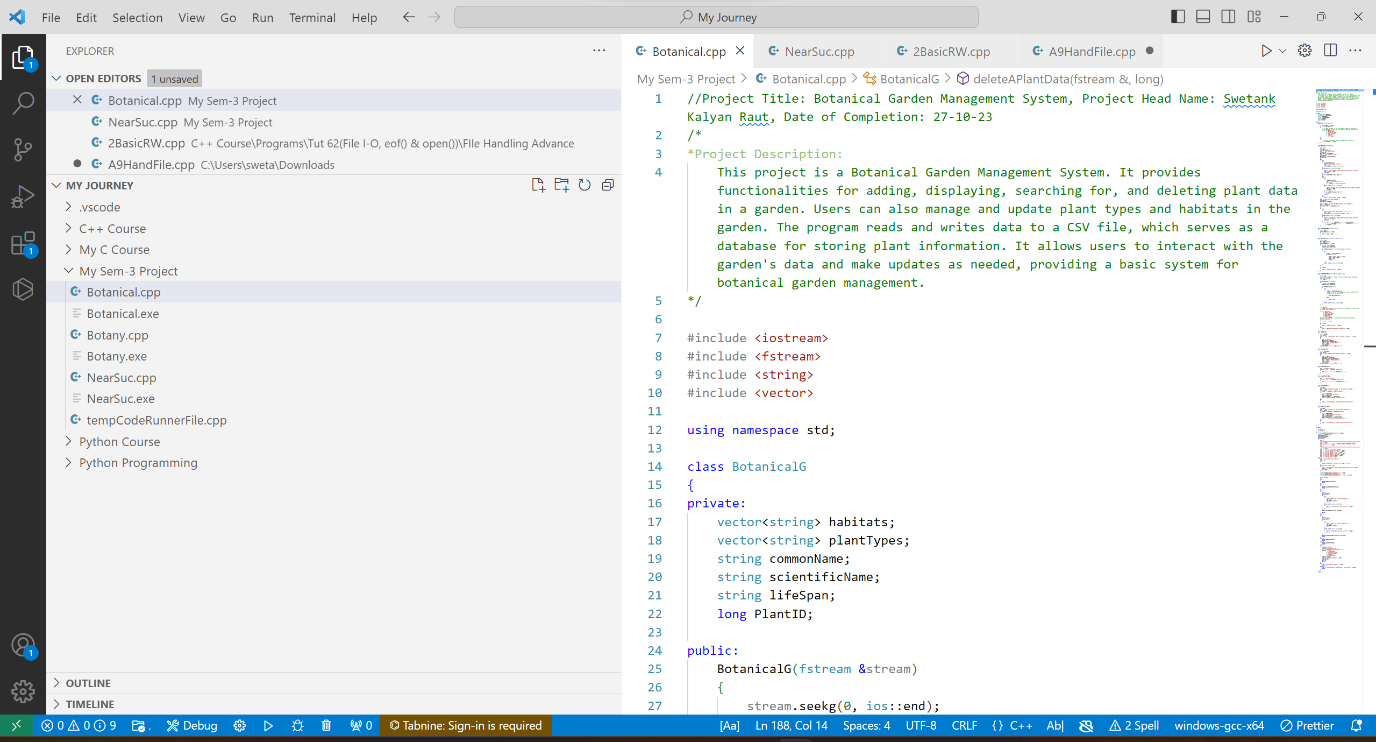
    return 0;

}

# Output:

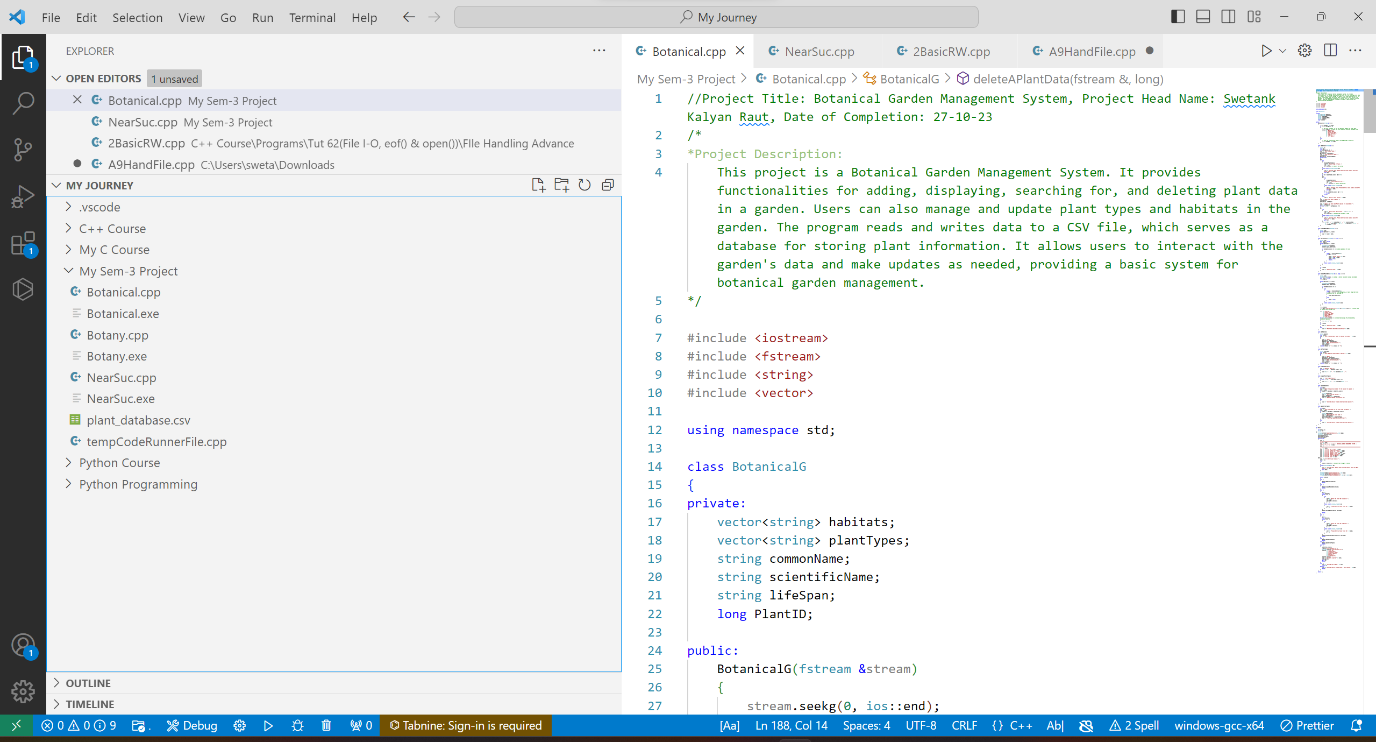
## On Console and CSV:

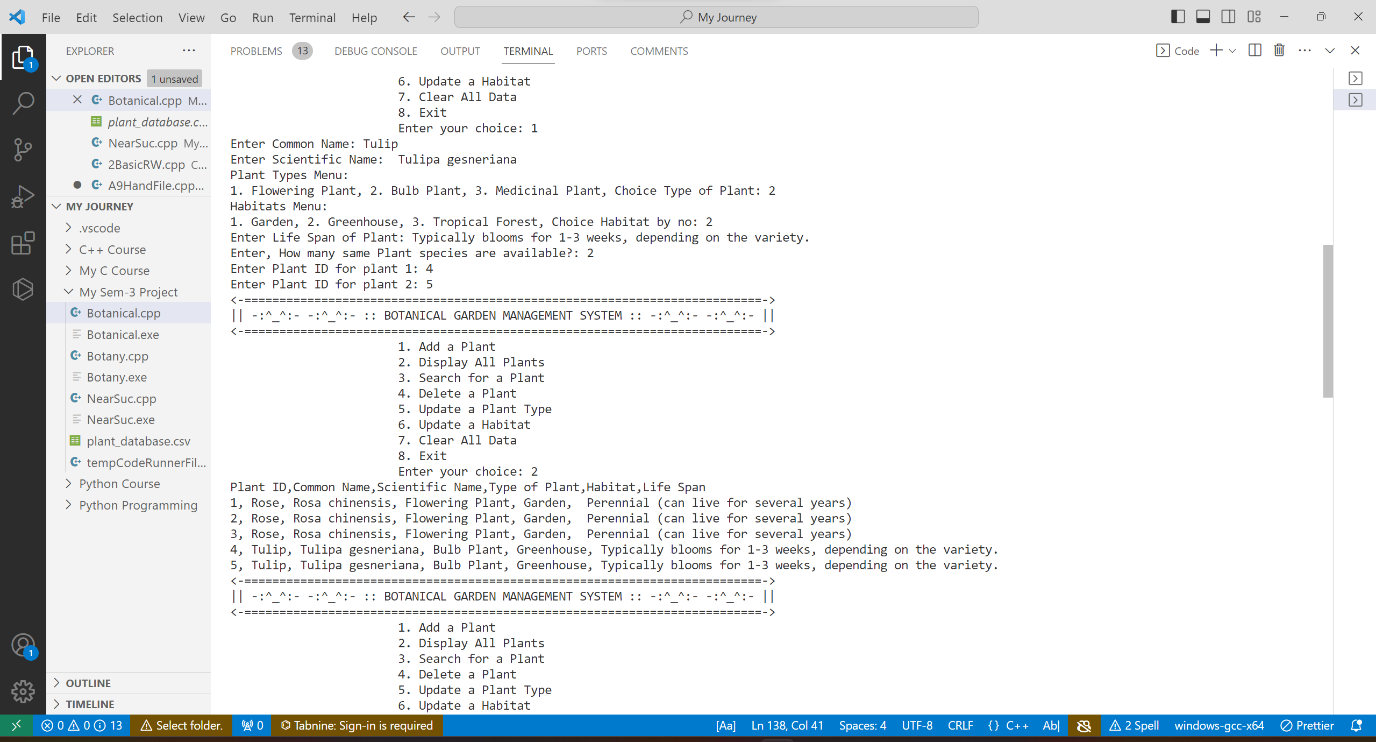
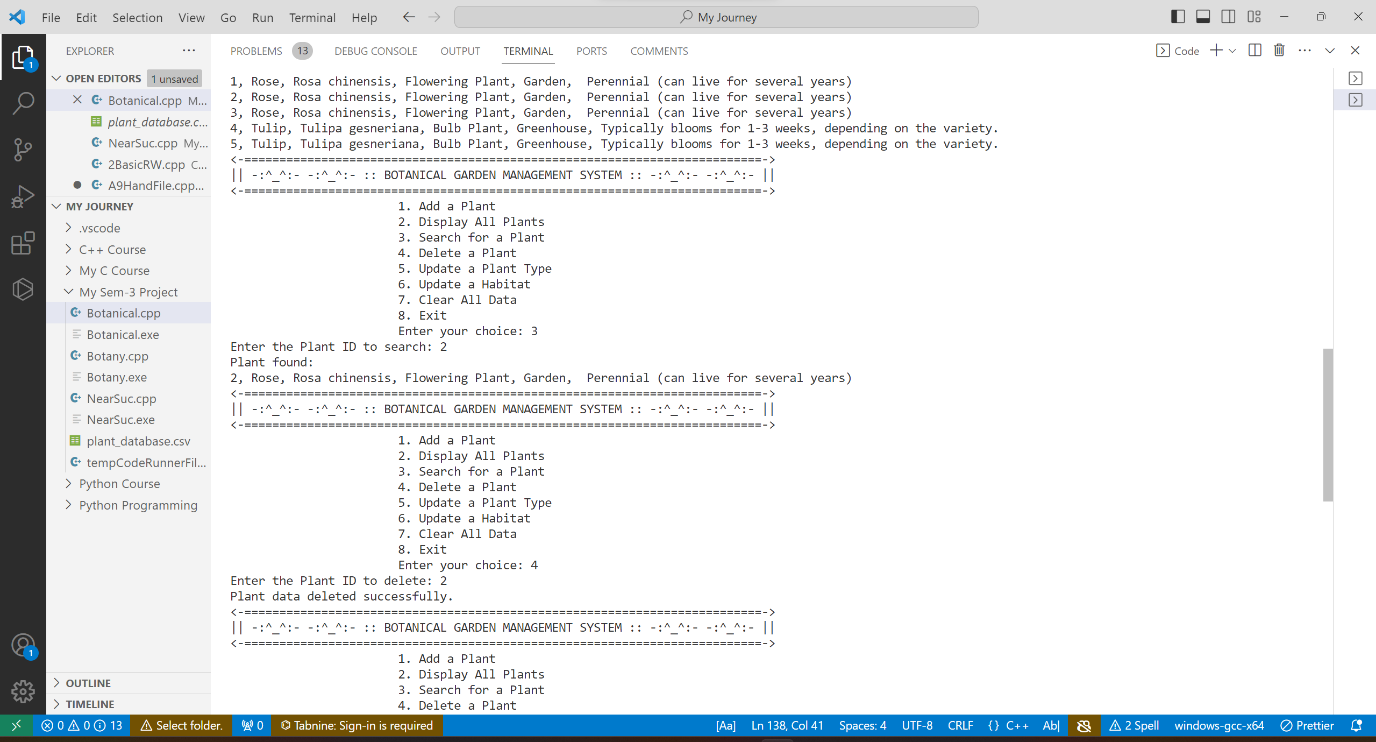
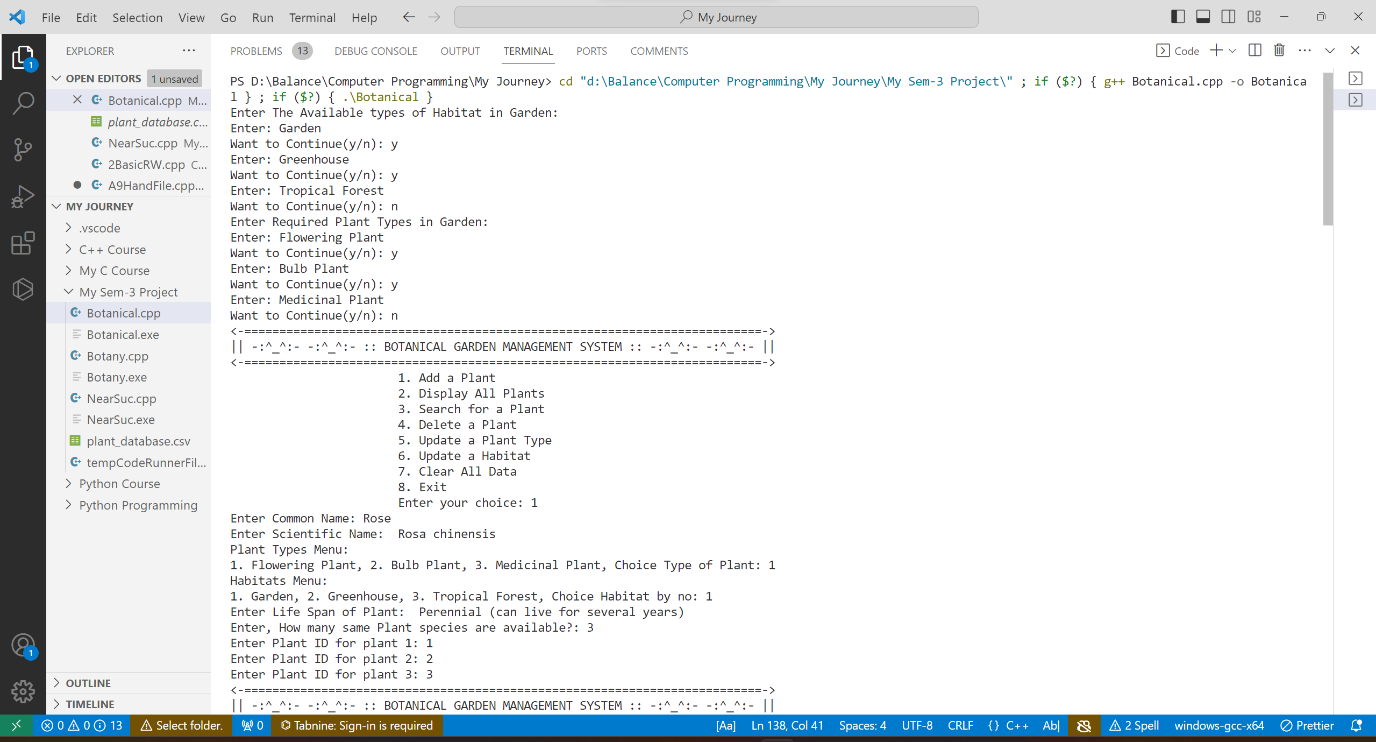
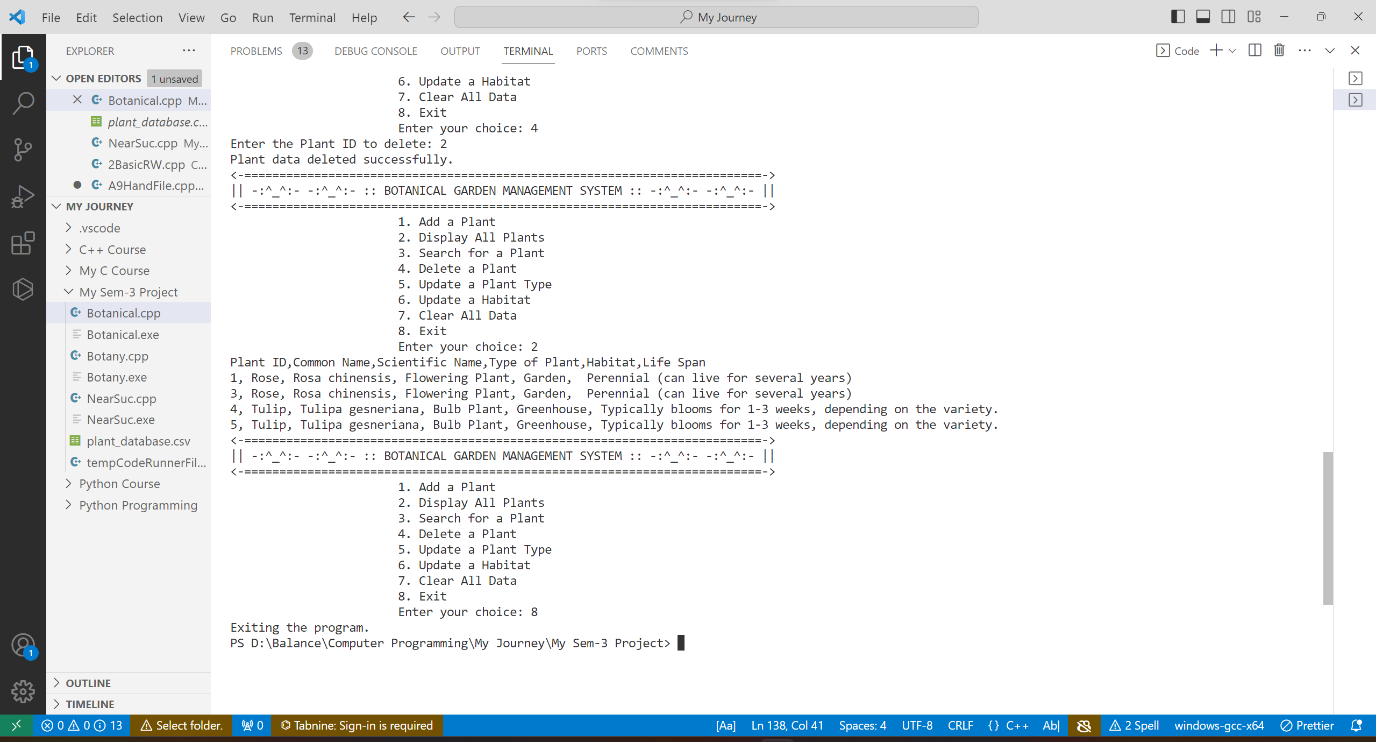
### Before Execution of program:

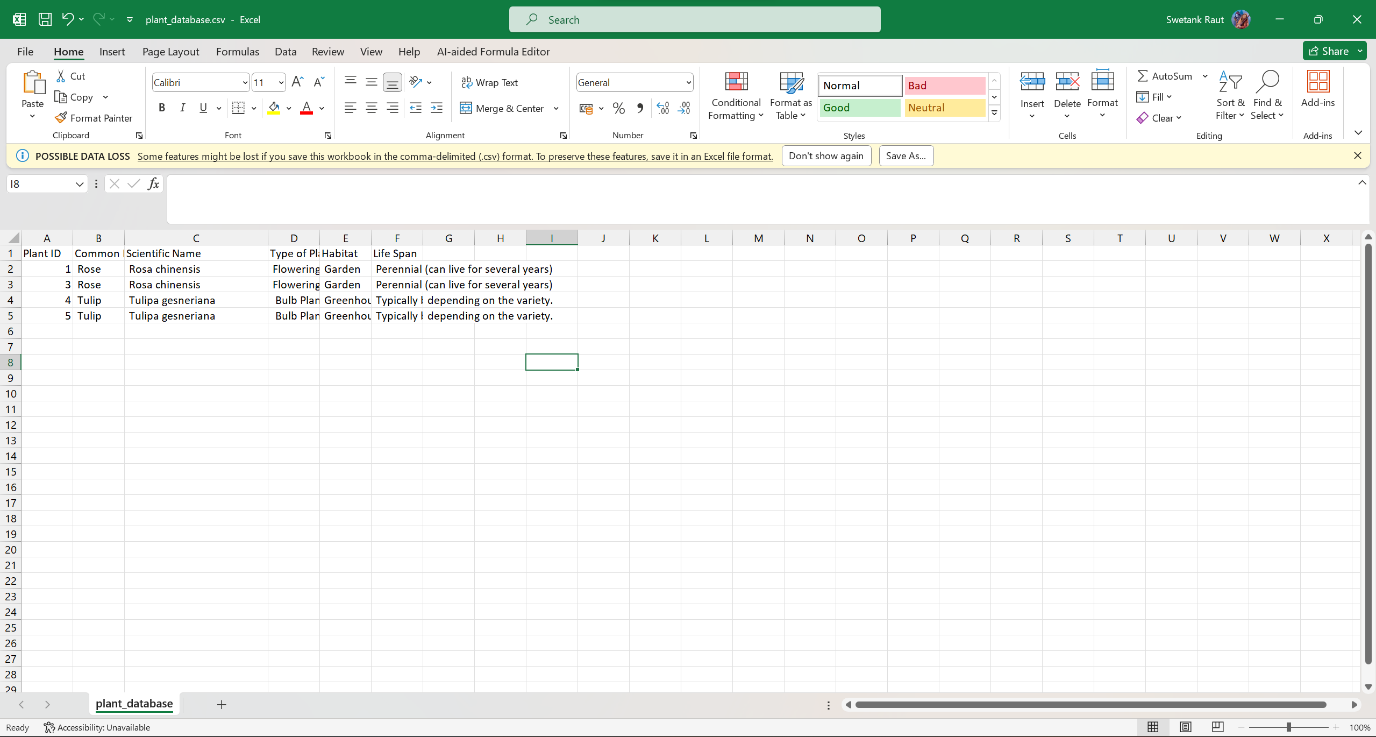


### After Execution:

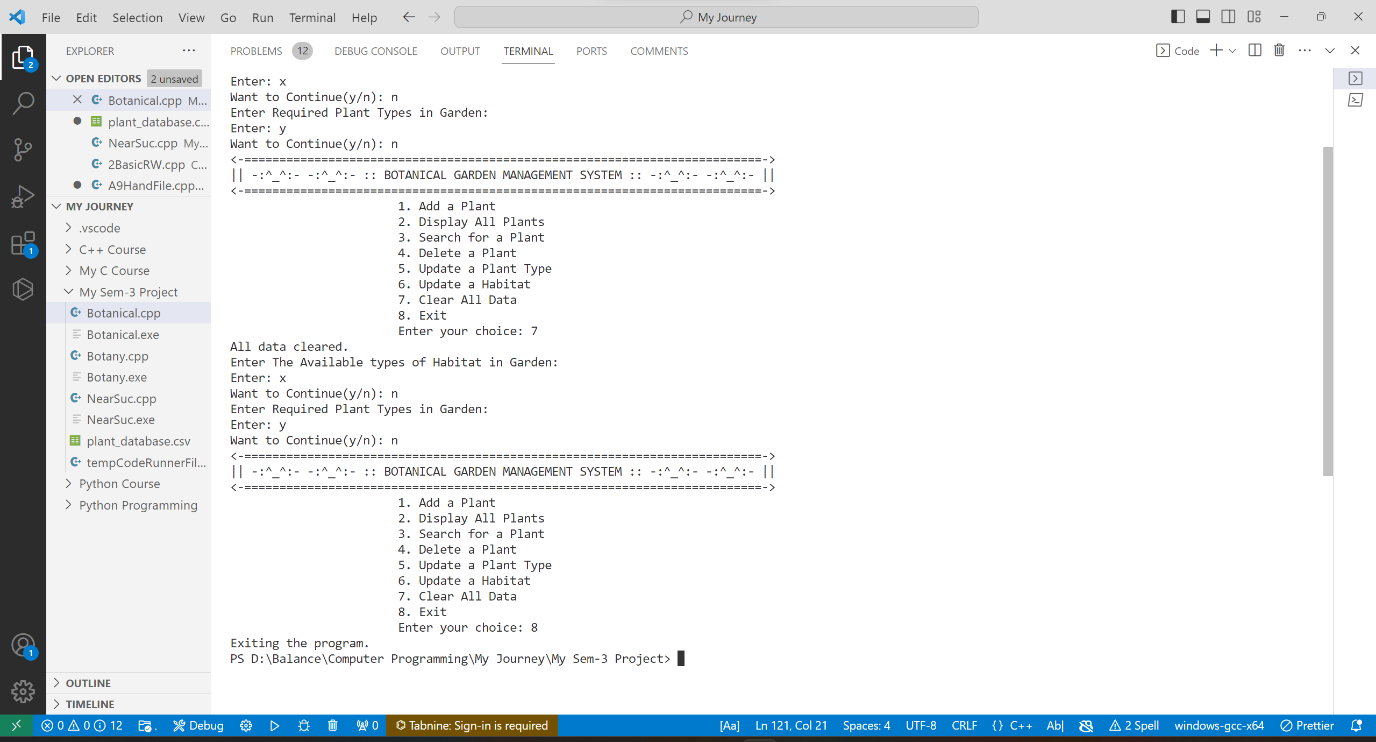
* Created plant\_data.csv:



* Inside plant\_data.csv: 

### Clearing all data:



* After Deletion inside plat\_data.csv:

