

PROJECT REQUIREMENTS AND INSTRUCTIONS:

I. REQUIREMENTS

1. The program must use the following programming concepts:
 - a. Functions
 - b. Arrays
 - c. Structures
 - d. Pointers and/or Linked List
 - e. File Manipulation
2. The program must use an external file (.txt) as the storage of all the records that are added to the program.
3. The program must have a Main Menu wherein the user can select the operation he wants. Operations must be:
 - a. Add new record
 - b. Search record
 - c. Delete record
 - d. Display records
4. The program must display the name of your group and the names of the members of the group when the user exits from the program.
5. The program must have basic error trapping feature, like when a user enters an invalid input, or invalid form of data.

II. INSTRUCTIONS

1. Create a Student Information program.
2. The program will ask the user what he wants to do by providing him a menu that shows the operations that the program can do. Example:

```
Welcome to Group Pr0gr@mm@z Student Information System
```

```
What do you want to do?
```

1. Add New Record
2. Search Record
3. Display All Records
4. Display Specific Record
5. Delete Record
6. Exit

```
Please type your selection: __
```

3. When the user selects an operation from the list, the program will execute the selected operation.

4. If the user inputs a number that is not in the menu, e.g. 7, the program will tell the user that he entered a wrong input and will ask the user to enter another selection.
5. The program must store the following:
 - a. Student ID Number
 - b. Full Name
 - c. Birthday
 - d. Address
 - e. Gender
 - f. Degree Program
 - g. Year Level
6. Your program can have one or more ways of searching for a record such as searching by name, or by ID Number.
7. Create a video presentation which shows how your program is used and how it works. Any member or all members of the group can appear in the video. (This video presentation will be upload to your Google Drive or to any video streaming platform online).



FEU Institute of Technology

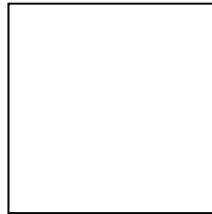
COLLEGE OF ENGINEERING • COLLEGE OF COMPUTER STUDIES

Department of Information Technology

CCS0007 – COMPUTER PROGRAMMING 2

AW-12

Final Project



Grade

Submitted by:

RIVAS, DAVE MAURECE P.



Submitted to:

Prof. Merlin A. Marfa

DOCUMENT CONTENTS:

I. INTRODUCTION

Provide some statement about your project or computer program.

This C++ program uses an easy-to-navigate menu system for people of all ages making it user friendly, allowing users to select their desired action by entering a number. This loop repeats until the user decides to quit by inputting '5'. Moreover, it acknowledges the developer's name at the end of the program.

II. OBJECTIVES

State the main objective of the project.

This project is a straightforward system for managing student records, created using C++. It offers essential features like adding, searching, deleting, and displaying student information. The program stores data in a text file called "records.txt" using file handling techniques.

Here's a brief rundown of what the program can do:

Add Record: Users can input details like student ID, full name, birthdate, address, gender, degree program, and year level to create a new student record.

Search Record: Users can find specific student records by entering their student ID. If found, the program displays all the details of the student.

Delete Record: Users have the option to remove a student record by providing the student ID. If the record exists, it gets deleted from the file.

Display Records: The program can show all the student records stored in the file, allowing users to view the entire list of students.

III. PROGRAM SCREEN SHOTS

Capture every screen of your running program and provide some explanation what is in the screen shot. See example below...

```
1. Add Record
2. Search Record
3. Delete Record
4. Display Records
5. Exit
Please enter your selection:
```

Figure 1. Main Screen

Figure 1 shows The main menu of the program, with the available options to choose from such as, add record, search record, delete record and display records.

```
Add new record
Enter Student ID: 2023
Enter Full Name: DAVE RIVAS
Enter Birthdate (mm/dd/yyyy): 04 14 05
Enter Address: PHILIPINES
Enter Gender (M/F): M
Enter Degree Program: BSIT
Enter Year Level (1/2/3/4/Irregular): 1
Record Added!
```

Figure 2. Add Record

Figure 2 shows the function of the add record and its usability requiring the user to enter His/Her information such as, Student ID, Full Name, Birthdate, Address, Gender, Degree program and Year level.

```
Search Record
Enter Student ID to search: 2023

Record Found:
Student ID: 2023
Full Name: DAVE RIVAS
Birthday: 04 14 05
Address: PHILIPINES
Gender: M
Degree Program: BSIT
Year Level (1/2/3/4/Irregular): 1
-----
```

Figure 3. Search Record

Figure 3 shows the function of the Search Record which the system prompts the user to enter the Student ID to find or gather the stored data.

```
1. Add Record
2. Search Record
3. Delete Record
4. Display Records
5. Exit
Please enter your selection: 3
Enter Student ID to delete: 2023
Record deleted successfully.
```

Figure 4. Delete Record

Figure 4 shows the function of the Delete Record which the system prompts the user to enter the Student ID to find or gather the stored data hence this time deleting it from the record database.

```
1. Add Record
2. Search Record
3. Delete Record
4. Display Records
5. Exit
Please enter your selection: 4
Student ID: 2023
Full Name: DAVE RIVAS
Birthday: 04 14 05
Address: PHILIPINES
Gender: M
Degree Program: BSIT
Year Level (1/2/3/4/Irregular): 1
```

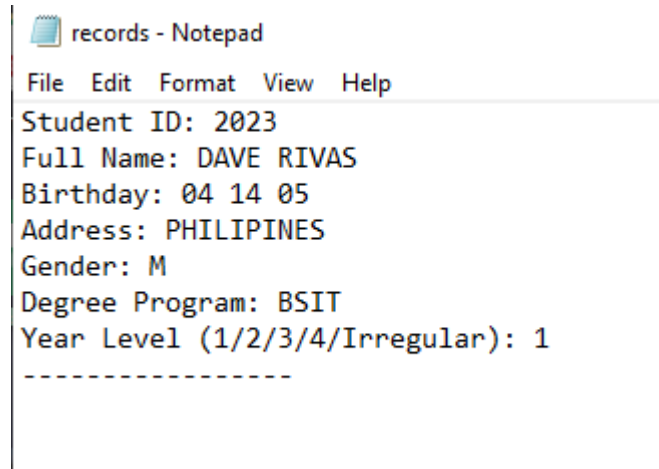
Figure 5. Display Record

Figure 5 shows the function of the Display Record, Displaying the input data of the add record function, this time showing the stored users data information such as, Student ID, Full Name, Birthdate, Address, Gender, Degree program and Year level.

```
1. Add Record
2. Search Record
3. Delete Record
4. Display Records
5. Exit
Please enter your selection: 5
Group: Solo
Members: Dave Maurece P. Rivas
-----
```

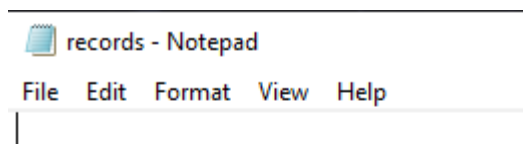
Figure 6. Exit Program

Figure 6 shows the option for Exiting the Program
This program once exiting shows the name of the developer of the code or the program itself.



```
records - Notepad
File Edit Format View Help
Student ID: 2023
Full Name: DAVE RIVAS
Birthday: 04 14 05
Address: PHILIPINES
Gender: M
Degree Program: BSIT
Year Level (1/2/3/4/Irregular): 1
-----
```

Figure 7. Records stored before deleting the record



```
records - Notepad
File Edit Format View Help
```

Figure 8. Records Deleted after deleting the record

PROGRAM DEMONSTRATION VIDEO: [CCS007L FINAL PROJECT RECORDING](#)

IV. SOURCE CODE

Paste the program codes with proper comments for each module and some explanation statements. **Use Courier New as your font for your codes, and Arial for the explanation.**

```

#include <iostream>
#include <fstream>
#include <string>
#include <algorithm>
using namespace std;

struct Student {
    string studentID;
    string fullname;
    string birthday;
    string address;
    char gender;
    string degreeProgram;
    string yearLevel;
    Student* next;
};

const string FILE_NAME = "records.txt";

void addRecord() {
    Student* newStudent = new Student();
    ofstream file(FILE_NAME.c_str(), ios::app);

    cout<<"Add new record"<<endl;
    cout<<"Enter Student ID: ";
    cin>>newStudent->studentID;
    cin.ignore();

    cout<<"Enter Full Name: ";
    getline(cin, newStudent->fullname);
    transform(newStudent->fullname.begin(), newStudent-
>fullname.end(), newStudent->fullname.begin(), ::toupper);

    cout<<"Enter Birthdate (mm/dd/yyyy): ";
    getline(cin, newStudent->birthday);
    transform(newStudent->birthday.begin(), newStudent-
>birthday.end(), newStudent->birthday.begin(), ::toupper);

    cout<<"Enter Address: ";

```



```

        getline(cin, newStudent->address);
        transform(newStudent->address.begin(), newStudent-
>address.end(), newStudent->address.begin(), ::toupper);

        cout<<"Enter Gender (M/F): ";
        cin>>newStudent->gender;
        newStudent->gender = toupper(newStudent->gender);
        cin.ignore();

        cout<<"Enter Degree Program: ";
        getline(cin, newStudent->degreeProgram);
        transform(newStudent->degreeProgram.begin(), newStudent-
>degreeProgram.end(), newStudent->degreeProgram.begin(),
::toupper);

        cout<<"Enter Year Level (1/2/3/4/Irregular): ";
        cin>>newStudent->yearLevel;
        transform(newStudent->yearLevel.begin(), newStudent-
>yearLevel.end(), newStudent->yearLevel.begin(), ::toupper);
        cin.ignore();

        newStudent->next = NULL;

        if (file.is_open()) {
            file<<"Student ID: "<<newStudent->studentID<<endl;
            file<<"Full Name: "<<newStudent->fullname<<endl;
            file<<"Birthday: "<<newStudent->birthday<<endl;
            file<<"Address: "<<newStudent->address<<endl;
            file<<"Gender: "<<newStudent->gender<<endl;
            file<<"Degree Program: "<<newStudent-
>degreeProgram<<endl;
            file<<"Year Level (1/2/3/4/Irregular): "<<newStudent-
>yearLevel<<endl;
            file<<"-----"<<endl;
            cout<<"Record Added!"<<endl;
            file.close();
        } else {
            cout<<"Error in file creation!"<<endl;
        }

        delete newStudent;

```

```

}

void searchRecord() {
    string searchID, line;
    bool found = false;

    cout<<"Search Record"<<endl;
    cout<<"Enter Student ID to search: ";
    cin>>searchID;
    transform(searchID.begin(), searchID.end(),
searchID.begin(), ::toupper);
    cin.ignore();

    ifstream file(FILE_NAME.c_str());
    if (file.is_open()) {
        while (getline(file, line)) {
            if (line.find("Student ID: " + searchID) !=
string::npos) {
                found = true;
                cout<<"\nRecord Found:\n";
                cout<<line<<endl;
                for (int i = 0; i < 6; ++i) {
                    getline(file, line);
                    cout<<line<<endl;
                }
                cout<<"-----"<<endl;
                break;
            }
        }
        if (!found) {
            cout<<"No record found with Student ID:
"<<searchID<<endl;
        }
        file.close();
    } else {
        cout<<"Unable to open file."<<endl;
    }
}

void displayRecords() {
    ifstream inFile(FILE_NAME.c_str());

```

```

    if (!inFile) {
        cout<<"Error: Unable to open file for reading."<<endl;
        return;
    }

    Student student;
    string line;
    while (getline(inFile, line)) {
        if (line.find("Student ID: ") != string::npos) {
            cout<<line<<endl;
            for (int i = 0; i < 6; ++i) {
                getline(inFile, line);
                cout<<line<<endl;
            }
            cout<<"-----"<<endl;
        }
    }
    inFile.close();
}

void deleteRecord(string id) {
    ifstream inFile(FILE_NAME.c_str());
    ofstream tempFile("temp.txt");

    if (!inFile || !tempFile) {
        cout<<"Error: Unable to open file for reading or
writing."<<endl;
        return;
    }

    string line;
    bool found = false;
    while (getline(inFile, line)) {
        if (line.find("Student ID: " + id) != string::npos) {
            for (int i = 0; i < 7; ++i) {
                getline(inFile, line);
            }
            found = true;
        } else {
            tempFile<<line<<endl;
        }
    }
}

```

```

    }
    inFile.close();
    tempFile.close();

    remove(FILE_NAME.c_str());
    rename("temp.txt", FILE_NAME.c_str());

    if (found) {
        cout<<"Record deleted successfully."<<endl;
    } else {
        cout<<"No record found with Student ID: "<<id<<endl;
    }
}

int main() {
    int choice;
    do {
        cout <<"1. Add Record\n2. Search Record\n3. Delete
Record\n4. Display Records\n5. Exit"<<endl;
        cout<<"Please enter your selection: ";
        cin>>choice;
        switch (choice) {
            case 1:
                addRecord();
                break;
            case 2:
                searchRecord();
                break;
            case 3:
                {
                    string id;
                    cout<<"Enter Student ID to delete: ";
                    cin>>id;
                    deleteRecord(id);
                }
                break;
            case 4:
                displayRecords();
                break;
            default:
                break;
        }
    }
}

```

```

        }
    } while (choice != 5);

    cout<<"Group: Solo"<<endl;
    cout<<"Members: Dave Maurece P. Rivas"<<endl;

    return 0;
}

```

V. SUMMARY AND DISCUSSION

Write a summary about the program, the program codes, and the things that you have learned from doing this program and ultimately from this course.

This program is a student record management system created in C++. It allows users to add, search, delete, and display student records stored in a text file named "records.txt". Users can input details such as student ID, name, birthdate, address, gender, degree program, and year level for record addition. The program provides a menu-driven interface where users can choose their desired operation by entering a corresponding numeric choice. The main loop continues until the user chooses to exit by entering '5'. Additionally, the program acknowledges the name of the solo developer, Dave Maurece P. Rivas, who created it.

From developing this program, here are the things that i have learned, I gained a deeper understanding of file handling in C++, specifically how to read from and write to text files. Additionally, I learned about dynamic memory allocation and the use of structures to organize data efficiently. Moreover, the implementation of a menu-driven interface enhanced my comprehension of control flow and user interaction in C++ programming. Overall, this project provided valuable insights into various aspects of C++ programming, reinforcing fundamental concepts and enhancing my problem-solving skills. I also have learned that this course have a broad and a wide section of learning and it makes us as an individual want to learn or explore more new things not only about this course but also in the modern world of computer technologies, programming etc. The code debugging for example is a technical skill that is learned through this course as well as more other things to learn if you explore the world of programming deeply.

VI. RUBRICS

| TRAIT | Exceptional | Acceptable | Amateur | Unsatisfactory |
|--------------------------------|--|--|---|---|
| Specifications (20) | The program works and meets all of the specifications. | The program works and produces the correct results and displays them correctly. It also meets most of the other specifications. | The program produces correct results but does not display them correctly. | The program is producing incorrect results. |
| Readability (20) | The code is exceptionally well organized and very easy to follow. | The code is fairly easy to read. | The code is readable only by someone who knows what it is supposed to be doing. | The code is poorly organized and very difficult to read. |
| Reusability (20) | The code could be reused as a whole or each routine could be reused. | Most of the code could be reused in other programs. | Some parts of the code could be reused in other programs. | The code is not organized for reusability. |
| Documentation (30) | The documentation is well written and clearly explains what the code is accomplishing and how. | The documentation consists of embedded comment and some simple header documentation that is somewhat useful in understanding the code. | The documentation is simply comments embedded in the code with some simple header comments separating routines. | The documentation is simply comments embedded in the code and does not help the reader understand the code. |
| Efficiency (10) | The code is extremely efficient without sacrificing readability and understanding. | The code is fairly efficient without sacrificing readability and understanding. | The code is brute force and unnecessarily long. | The code is huge and appears to be patched together. |