



DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

PROJECT REPORT ON

STUDENT MANAGEMENT SYSTEM

CSE 104 || SEC: DG
DATE OF SUBMISSION: 13-05-2022

SUBMITTED BY

SUBMITTED TO

MD. SAIFUL ISLAM RIMON
STU. ID: 213002039

MD GULZAR HUSSAIN
LECTURER,
ELECTRICAL AND ELECTRONIC
ENGINEERING.
(PROJECT SUPERVISOR)



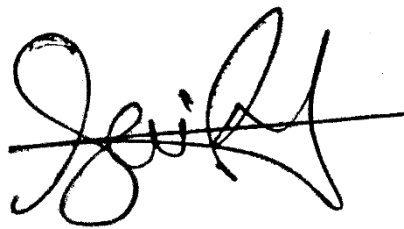
GREEN UNIVERSITY OF BANGLADESH
DHAKA-1216, BANGLADESH

DECLARATION

GREEN UNIVERSITY OF BANGLADESH

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Here, I am **MD SAIFUL ISLAM RIMON** (213002039) confirm that this report and the work presented in it are my own achievement. I have read and understood the penalties associated with plagiarism.

A handwritten signature in black ink, appearing to read 'Saiful', with a horizontal line crossing through the middle of the letters.

MD SAIFUL ISLAM RIMON

13-05-2022

Date:

CERTIFICATION

GREEN UNIVERSITY OF BANGLADESH

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

This is to certify that this project is fully adequate in scope and quality as an undergraduate project work.

.....

MD GULZAR HUSSAIN

LECTURER,

ELECTRICAL AND ELECTRONIC ENGINEERING.

(PROJECT SUPERVISOR)

DATE

ABSTRACT

STUDENT MANAGEMENT SYSTEM

The idea of developing a student management system is to improve communication between professors and parents. The student management system can be used by teachers, parents, and students. It aids in the tracking of a student's development so that the optimal decisions for the student's learning path may be made. Many educational institutions currently employ computer systems to arrange a student's data. The software assists the admissions department in keeping track of students' information and admissions. The software also keeps track of changes in the profiles of qualified students as well as their performances.

TABLE OF CONTENT

CONTEXT	PAGE
DECLARATION -----	02
CERTIFICATION -----	03
ABSTRACT -----	04
LIST OF FIGURES -----	07
CHAPTER 1. INTRODUCTION -----	08
1.1 WHAT IS STUDENT MANAGEMENT SYSTEM -----	08
1.2 WHAT DOES STUDENT MANAGEMENT SHOULD DO	08
1.3 WHAT DOES STUDENT MANAGEMENT SHOULD	
INCLUDE -----	09
1.4 AIM AND OBJECTIVES -----	10
CHAPTER 2. PROJECT IMPLEMENTATION -----	11
2.1 C PROGRAM -----	11
2.2 CODE -----	12
2.3 OUTPUT -----	26
CHAPTER 3. MATERIALS AND PROCEDURES-----	31
3.1 MATERIALS-----	31
3.2 PROCEDURES-----	31
CHAPTER 4. TERMINOLOGY -----	33
4.1 IDE (INTEGRATER DEVELOPMENT ENVIRONMENT)	33
4.2 CODE -----	33
4.3 BUILD AND RUN -----	34
4.3 DATABASE -----	34
4.3 ENCRYPTION -----	35
CHAPTER 5. FEATURES AND LIMITATIONS -----	36
5.1 FEATURES -----	36
5.2 LIMITATIONS -----	36
CHAPTER 6. PERFORMANCE EVALUATION -----	37
6.1 PERFORMANCE EVALUATION -----	37

6.2 RESULT AND DISCUSSION -----	37
CHAPTER 7. CONCLUSION -----	38
7.1 CONCLUSION-----	38
7.2 RECOMMENDATION AND FUTURE PLAN-----	38
REFERENCES -----	39

LIST OF FIGURES

1. *Figure 2.1: IEEE-the best 10 top programming language in 2018.*
2. *Figure 2.3.1: Starting interface asking password to give access in it.*
3. *Figure 2.3.2: Landing page of the app after logging*
4. *Figure 2.3.3: Option choosing page*
5. *Figure 2.3.4: Student adding page*
6. *Figure 2.3.5: Student information modifying page*
7. *Figure 2.3.2: Landing page of the app after logging*
8. *Figure 2.3.3: Option choosing page*
9. *Figure 2.3.5: Student information modifying page*
10. *Figure 2.3.2: Landing page of the app after logging*
11. *Figure 2.3.4: Student adding page*
12. *Figure 2.3.6: Showing all student information in one page*
13. *Figure 2.3.7: Showing individual view of student's information*
14. *Figure 2.3.8: Student information deleting page*
15. *Figure 2.3.9: Password changing page*
16. *Figure 2.3.10: Encrypted database*

CHAPTER 1: INTRODUCTION

1.1 WHAT IS STUDENT MANAGEMENT SYSTEM?

A Student Management System is also known as a Student Information System (SIS). These systems work to coordinate scheduling and communications between faculty regarding students. This system exists to simplify information tracking for both parents and administrative staff.

1.2 WHAT DOES STUDENT MANAGEMENT SHOULD DO?

Student-centric management systems work to make the management of information more accessible. Below is a list of tracking pieces you might have on your information management system:

- Health information
- Schedules
- Grade book information
- Behavior data
- Age
- Transcript information
- Grade level

The focus of this sort of management system is that it is student-centric. This software differs from your other type of management system: school management systems.

1.3 WHAT SHOULD A STUDENT MANAGEMENT SHOULD INCLUDE?

For an ideal *Student Management System* there is something need which are obvious.

- Student information
- Student Schedule Management
- Grade Book and Transcript Information
- Student Health Information
- An Easy Interface
- Easy adding system
- Easy modifying system
- Easy deleting system
- Viewing option
- Security
- Database system

1.4 Aim and Objectives

The Aim of this project is to design and implement a student management system that is able to perform the tasks which are included to the system..

The Objectives of the project are as follows:

- Implement the security first.
- Implement all the features as far as can possible.
- Implement an easy operating user interface.
- Encrypted Database system.

CHAPTER 2: PROJECT IMPLEMENTATION

For developing this project, I have used C as programming language. Let's discuss about it.

2.1 C PROGRAM

C is a general-purpose programming language that is extremely popular, simple, and flexible to use. It is a structured programming language that is machine-independent and extensively used to write various applications, Operating Systems like Windows, and many other complex programs like Oracle database, Git, Python interpreter, and more.

It is said that 'C' is a god's programming language. One can say, C is a base for the programming. If you know 'C,' you can easily grasp the knowledge of the other programming languages that uses the concept of 'C'

It is essential to have a background in computer memory mechanisms because it is an important aspect when dealing with the C programming language.

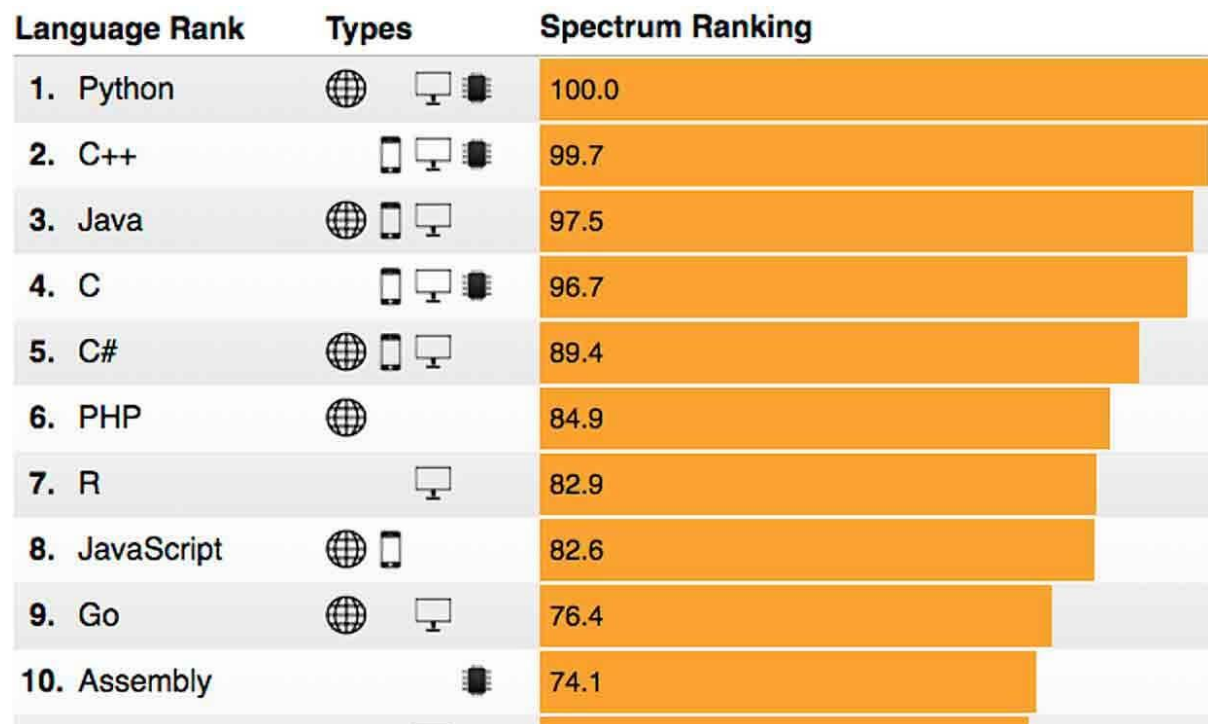


Figure 2.1: IEEE-the best 10 top programming language in 2018

2.2 Code

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<math.h>
#include <windows.h>

#define Student struct Stud

void add(FILE * fp);
void modify(FILE * fp);
void display(FILE * fp);
void Individual(FILE *fp);
void password();
FILE * del(FILE * fp);
void printChar(char ch,int n);
void title();
FILE *tp;

void gotoxy(int x,int y)
{
    COORD CRD;
    CRD.X = x;
    CRD.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),CRD);
}

struct pass
{
    char pass[25];
}pa;

struct Stud
```

```

{
    char name[100];
    char dept[50];
    int roll;
    float sgpa[12];
    float cgpa;
};

int main()
{
    int ch,id,k,i;
    char c,pas[50];
    SetConsoleTitle("Student Management System | GREEN UNIVERSITY OF
BANGLADESH | 213002039");
    FILE * fp;
    Student s;
    int option;
    char another;

    if((fp=fopen("db.txt","rb+"))==NULL)
    {
        if((fp=fopen("db.txt","wb+"))==NULL)
        {
            printf("Can't create or open Database.");
            return 0;
        }
    }

    system("color 9f");
    gotoxy(42,8);
    printf("LOGIN(If 1st login press ENTER)");
    gotoxy(42,10);
    printf("_____");

```

```

gotoxy(42,11);
printf("\tEnter password :      |");
gotoxy(42,12);
printf("|_____|");
//printf("\n\t\t\t\t\t");
gotoxy(65,11);
while( k<10)
{
    pas[k]=getch();
    char s=pas[k];
    if(s==13)
        break;
    else printf("*");
    k++;
}
pas[k]='\0';
tp=fopen("F:/Password.txt","r+");
fgets(pa.pass,25,tp);
if(strcmp(pas,pa.pass)==0)
{
    system("cls");
    gotoxy(10,3);
    printf("<<<< Loading Please Wait >>>>");
    for(i=0; i<5; i++)
    {
        printf("\t(*_*)");
        Sleep(500);
    }
    printf("\n\n\n\n\n\t\t\t\t\t * * * * *");
    printf("\n\n\t\t\t\t\t *          *");
    printf("\n\n\t\t\t\t\t *    Welcome    *");
    printf("\n\n\t\t\t\t\t *          *");
    printf("\n\n\t\t\t\t\t * * * * *");

```

```

        printf("\n\n\n\n\n\t\t\t\t\tPress any key to continue..... ");
        getch();

title();
printf("\n\n\t\t\t\t\tLab Final || CSE104");
printf("\n\n\t\t\t\t\t GREEN UNIVERSITY OF BANGLADESH\n\t\t\t\t\t");
printf("\n\n\t\t\t\t\t MD SAIFUL ISLAM RIMON | 213002039\n\t\t\t\t\t");
printfChar('=',38);
printf("\n\n\n\t\t\t\t\t press any key to Enter");
getch();

while(1)
{
    title();
    printf("\n\t\t\t\t\t");
    printfChar('*',64);

    printf("\n\n\t\t\t\t\t MD SAIFUL ISLAM RIMON | 213002039\n\t\t\t\t\t");
    printf("\n\n\t\t\t\t\t\t1. Add Student");
    printf("\n\n\t\t\t\t\t\t2. Modify Student");
    printf("\n\n\t\t\t\t\t\t3. Show All Student");
    printf("\n\n\t\t\t\t\t\t4. Individual View");
    printf("\n\n\t\t\t\t\t\t5. Remove Student");
    printf("\n\n\t\t\t\t\t\t6. Change Password");
    printf("\n\n\t\t\t\t\t\t7. Logout\n\t\t\t\t\t");
    printfChar('*',64);
    printf("\n\n\n\t\t\t\t\t\tEnter Your Option :--> ");
    scanf("%d",&option);

    switch(option)
    {
        case 1:
            add(fp);

```

```

        break;
    case 2:
        modify(fp);
        break;
    case 3:
        display(fp);
        break;
    case 4:
        Individual(fp);
        break;
    case 5:
        fp=del(fp);
        break;
    case 6:
        system("cls");
        system("color 5f");
        password();

        break;
    case 7:
        return 1;
        break;
    default:
        printf("\n\t\tNo Action Detected");
        printf("\n\t\tPress Any Key\n\n");
        getch();
        system("pause");
    }
}
}
else
{
    printf("Wrong Password . Get Out");
    getch();
}

```



```

    }
    return 1;

}

void password()
{
    char c;
    printf("\nEnter new password :");
    fflush(stdin);
    gets(pa.pass);
    printf("\nSave password (y/n) :");
    fflush(stdin);
    scanf("%c",&c);
    if(c=='y' || c=='Y')
    {
        tp=fopen("F:/Password.txt","w+");
        fwrite(&pa,sizeof(pa),1,tp);
        fclose(tp);
        printf("\n\tPassword Saved\n");
    }
    else
    {
        printf("Password not saved :\n");
        printf("Press any key to continue >>>");
        getch();
    }
}

void printChar(char ch,int n)
{

```

```

while(n--)
{
    putchar(ch);
}

void title()
{
    system("cls");
    system("COLOR 03");
    printf("\n\n\t");
    printChar('=',19);
    printf(" Student Management System ");
    printChar('=',19);
    printf("\n");
}

//Insert at end

void add(FILE * fp)
{
    title();

    char another='y';
    Student s;
    int i;
    float cgpa;

    fseek(fp,0,SEEK_END);
    while(another=='y' || another=='Y')
    {

```

```

printf("\n\n\tEnter Full Name of Student: ");
fflush(stdin);
fgets(s.name,100,stdin);
s.name[strlen(s.name)-1]='\0';

printf("\n\n\tEnter Depertmt Name: ");
fflush(stdin);
fgets(s.dept,50,stdin);
s.dept[strlen(s.dept)-1]='\0';

printf("\n\n\tEnter Roll number: ");
scanf("%d",&s.roll);

printf("\n\n\tEnter SGPA for 12 semesters\n");
for(i=0,cgpa=0; i<12; i++)
{
    scanf("%f",&s.sgpa[i]);
    cgpa+=s.sgpa[i];
}

cgpa/=12.0;
s.cgpa=cgpa;

fwrite(&s,sizeof(s),1,fp);

printf("\n\n\tAdd another student?(Y/N)?");
fflush(stdin);
another=getchar();
}
}

```

```

FILE * del(FILE * fp)
{
    title();

    Student s;
    int flag=0,tempRoll,siz=sizeof(s);
    FILE *ft;

    if((ft=fopen("temp.txt","wb+"))==NULL)
    {
        printf("\n\n\t\t\t\t\t!!! ERROR !!!\n\t\t");
        system("pause");
        return fp;
    }

    printf("\n\n\tEnter Roll number of Student to Delete the Record");
    printf("\n\n\t\t\tRoll No. : ");
    scanf("%d",&tempRoll);

    rewind(fp);

    while((fread(&s,siz,1,fp))==1)
    {
        if(s.roll==tempRoll)
        {
            flag=1;
            printf("\n\tRecord Deleted for");
            printf("\n\n\t\t\t%s\n\t\t\t%s\n\t\t\t%d\n\t",s.name,s.dept,s.roll);
            continue;
        }

        fwrite(&s,siz,1,ft);
    }
}

```

```

    }

    fclose(fp);
    fclose(ft);

    remove("db.txt");
    rename("temp.txt","db.txt");

    if((fp=fopen("db.txt","rb+"))==NULL)
    {
        printf("ERROR");
        return NULL;
    }

    if(flag==0) printf("\n\n\t\tNO STUDENT FOUND WITH THE INFORMATION\n\t");

    printChar('-',65);
    printf("\n\t");
    system("pause");
    return fp;
}

void modify(FILE * fp)
{
    title();

    Student s;
    int i,flag=0,tempRoll,siz=sizeof(s);
    float cgpa;

    printf("\n\n\tEnter Roll Number of Student to MODIFY the Record : ");

```

```

scanf("%d",&tempRoll);

rewind(fp);

while((fread(&s,siz,1,fp))==1)
{
    if(s.roll==tempRoll)
    {
        flag=1;
        break;
    }
}

if(flag==1)
{
    fseek(fp,-siz,SEEK_CUR);
    printf("\n\n\t\t\tRecord Found\n\t\t\t");
    printChar('=',38);
    printf("\n\n\t\t\tStudent Name: %s",s.name);
    printf("\n\n\t\t\tStudent Roll: %d\n\t\t\t",s.roll);
    printChar('=',38);
    printf("\n\n\t\t\tEnter New Data for the student");

    printf("\n\n\t\t\tEnter Full Name of Student: ");
    fflush(stdin);
    fgets(s.name,100,stdin);
    s.name[strlen(s.name)-1]='\0';

    printf("\n\n\t\t\tEnter Department: ");
    fflush(stdin);
    fgets(s.dept,50,stdin);
    s.dept[strlen(s.dept)-1]='\0';
}

```

```

printf("\n\n\t\tEnter Roll number: ");
scanf("%d",&s.roll);


printf("\n\n\t\tEnter SGPA for 12 semesters\n");
for(i=0,cgpa=0; i<12; i++)
{
    scanf("%f",&s.sgpa[i]);
    cgpa+=s.sgpa[i];

}
cgpa=cgpa/8.0;


fwrite(&s,sizeof(s),1,fp);
}

else printf("\n\n\t\t!!! ERROR !!!! RECORD NOT FOUND");


printf("\n\n\t");
system("pause");
}

void display(FILE * fp)
{
    title();
    Student s;
    int i,siz=sizeof(s);

    rewind(fp);

    while((fread(&s,siz,1,fp))==1)

```

```

{
    printf("\n\t\tNAME : %s",s.name);
    printf("\n\t\tDepartment : %s",s.dept);
    printf("\n\t\tROLL : %d",s.roll);
    printf("\n\t\tSGPA: ");

    for(i=0; i<12; i++)
        printf("| %.2f |",s.sgpa[i]);
    printf("\n\t\tCGPA : %.2f\n\t",s.cgpa);
    printChar('-',65);
}
printf("\n\n\n\t");
printChar('*',65);
printf("\n\n\n\t");
system("pause");
}

```

```

void Individual(FILE *fp)

```

```

{
    title();

    int tempRoll,flag,siz,i;
    Student s;
    char another='y';

    siz=sizeof(s);

    while(another=='y' || another=='Y')
    {
        printf("\n\n\tEnter Roll Number: ");
        scanf("%d",&tempRoll);

        rewind(fp);
    }
}

```



```

while((fread(&s,siz,1,fp))==1)
{
    if(s.roll==tempRoll)
    {
        flag=1;
        break;
    }
}

if(flag==1)
{
    printf("\n\t\tNAME : %s",s.name);
    printf("\n\t\tDepartment : %s",s.dept);
    printf("\n\t\tROLL : %d",s.roll);
    printf("\n\t\tSGPA: ");

    for(i=0; i<12; i++)
        printf("| %.2f |",s.sgpa[i]);
    printf("\n\t\tCGPA : %.2f\n\t",s.cgpa);
    printChar('-',65);

}
else printf("\n\t\t!!!! ERROR RECORD NOT FOUND !!!!");

printf("\n\t\tShow another student information? (Y/N)?");
fflush(stdin);
another=getchar();
}
}

```

2.3 OUTPUT:

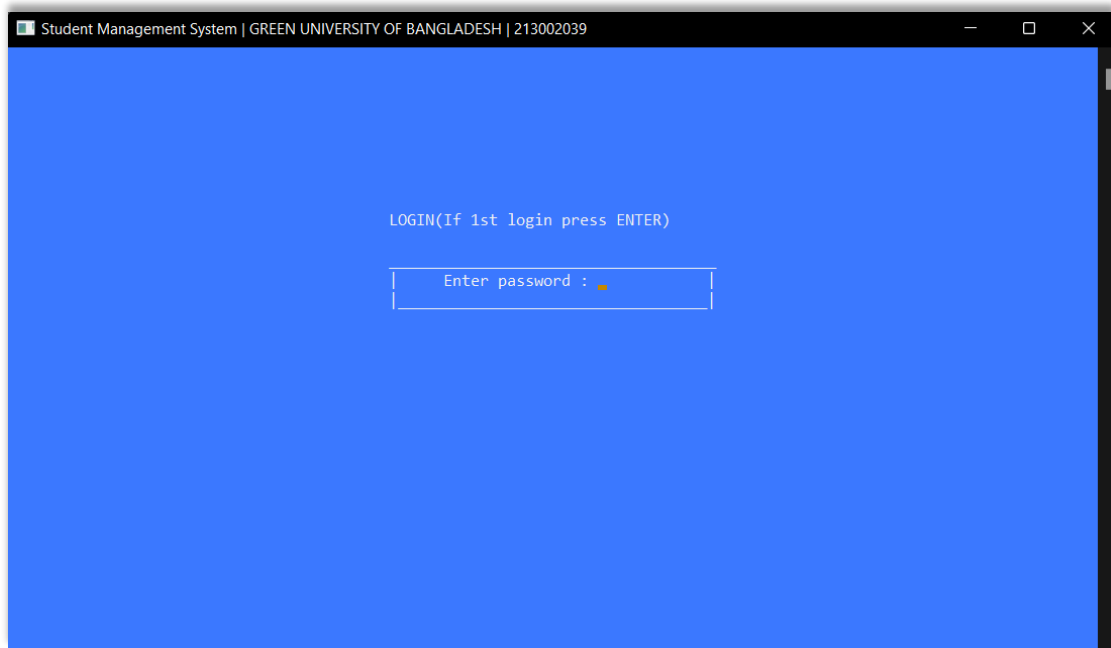


Figure 2.3.1: Starting interface asking password to give access in it.

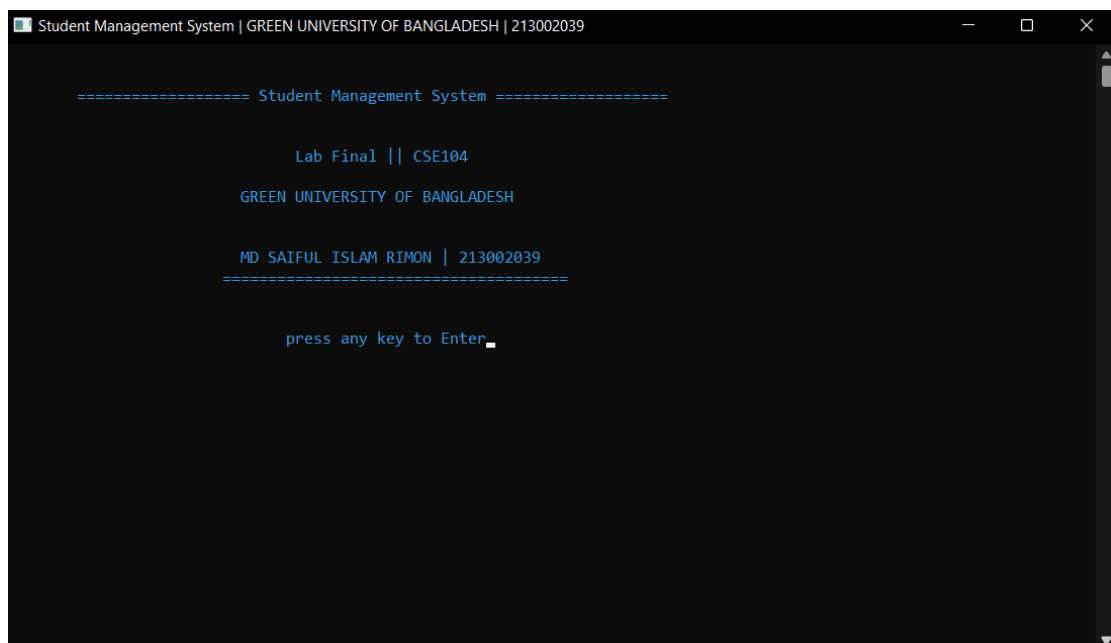


Figure 2.3.2: Landing page of the app after logging

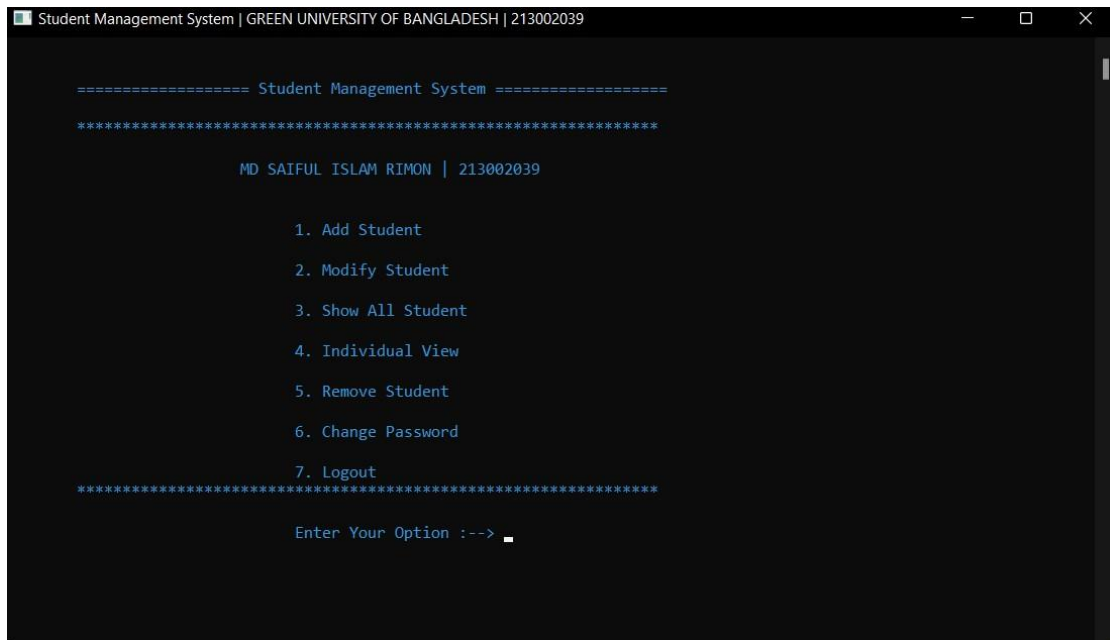


Figure 2.3.3: Option choosing page

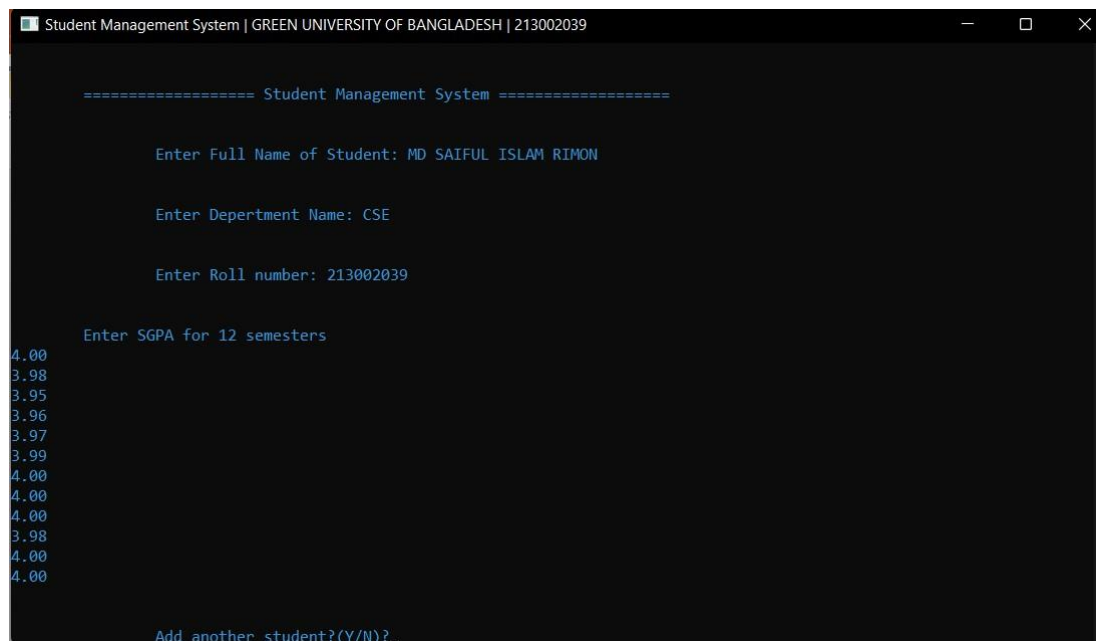


Figure 2.3.4: Student adding page

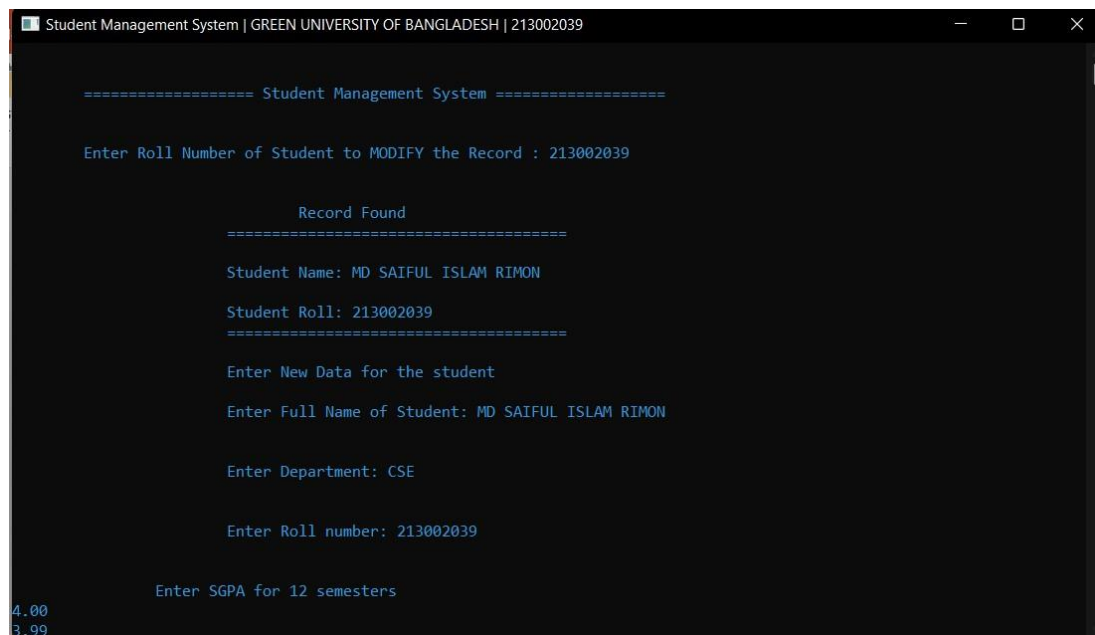


Figure 2.3.5: Student information modifying page

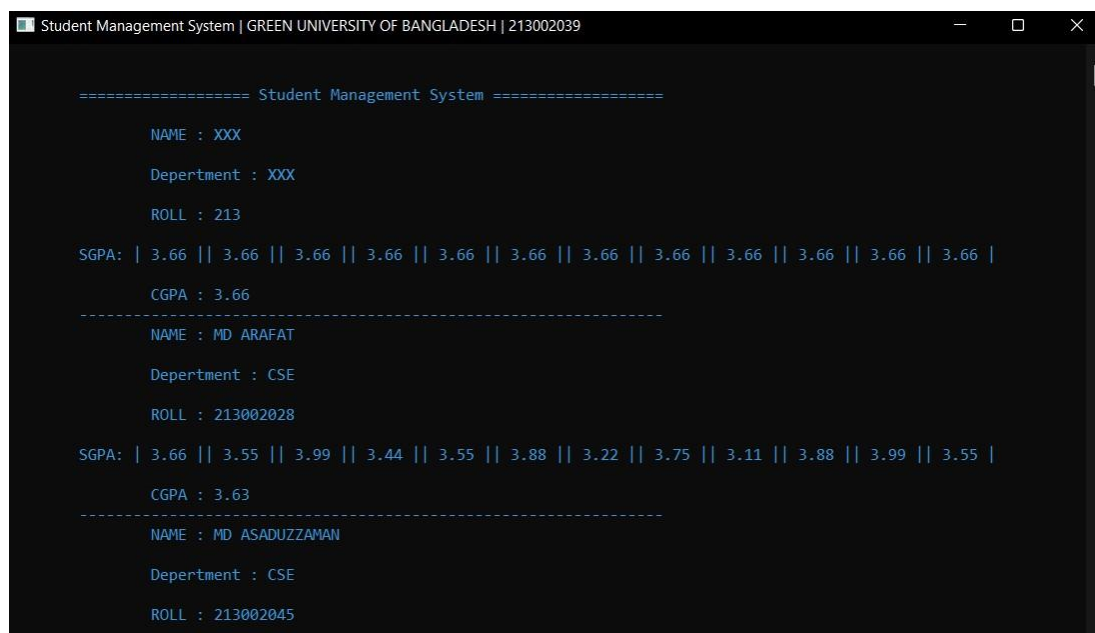


Figure 2.3.6: Showing all student information in one page

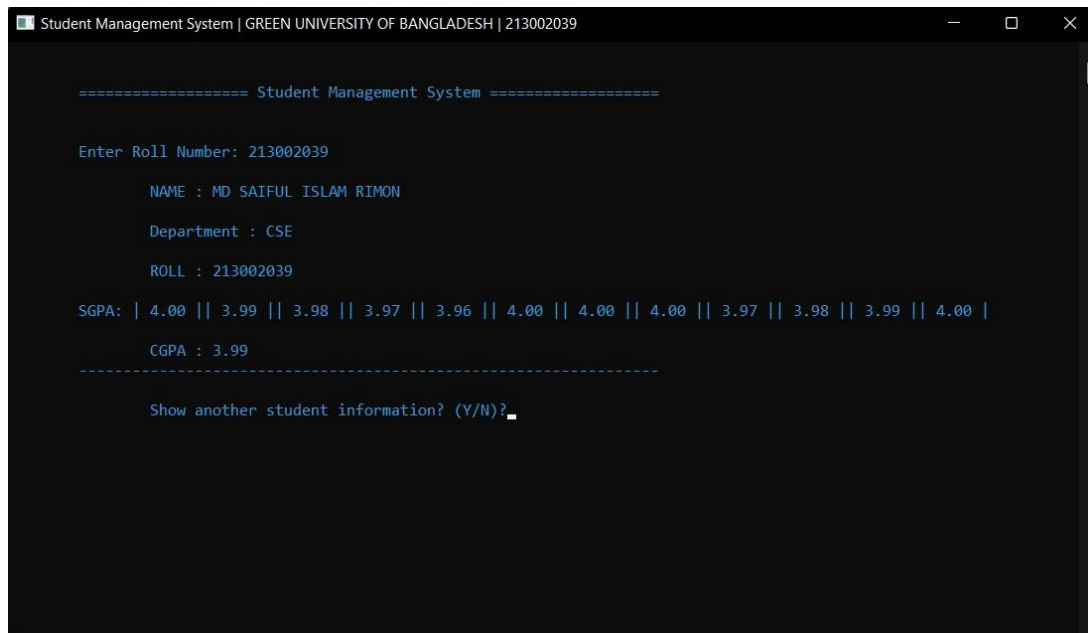


Figure 2.3.7: Showing individual view of student's information

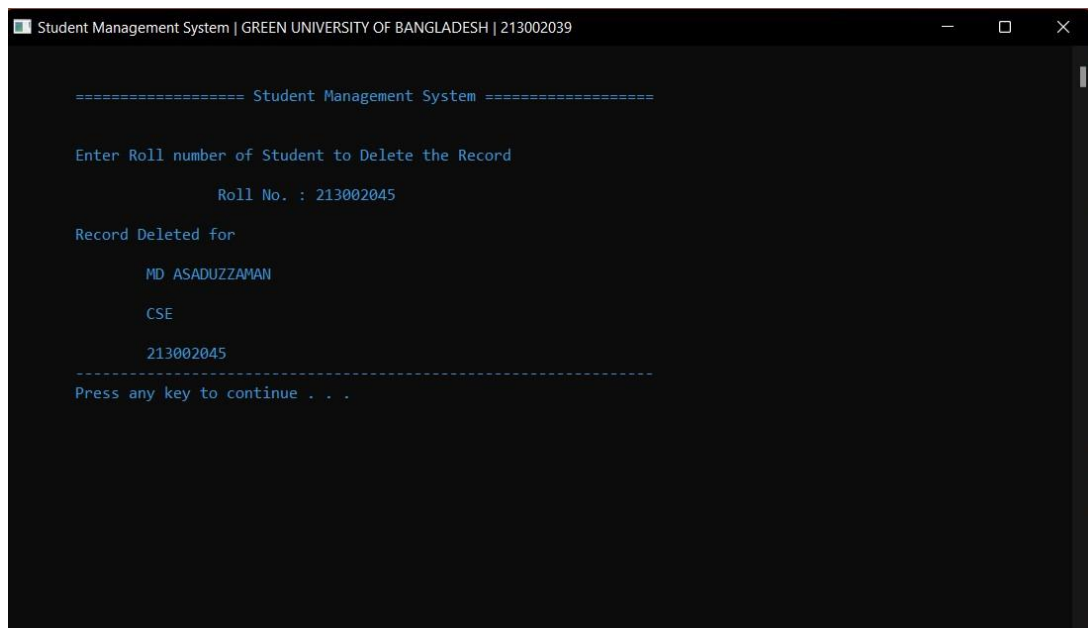


Figure 2.3.8: Student information deleting page



Figure 2.3.9: Password changing page

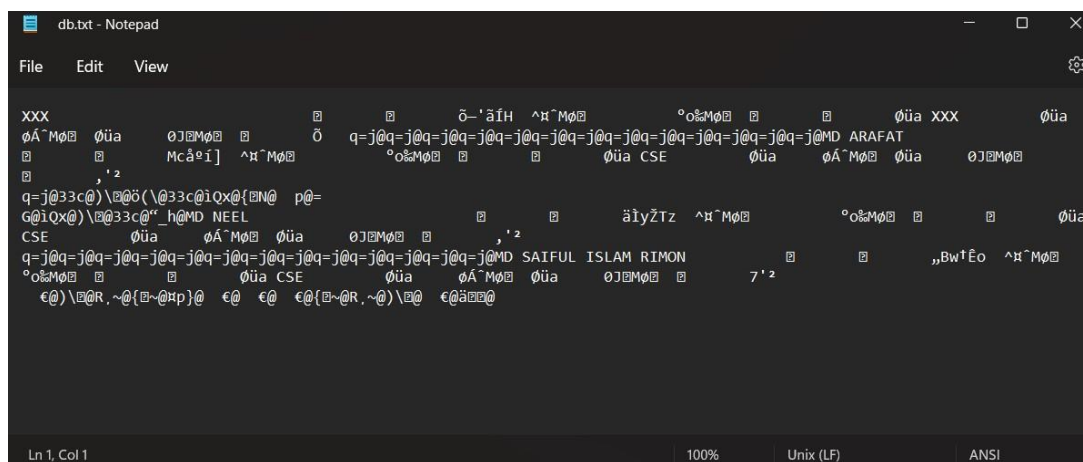


Figure 2.3.10: Encrypted database

CHAPTER 3: MATERIALS AND PROCEDURES

3.1 Materials

The following components were used for the implementation of the project:

- Computer
- IDE (Integrated Development Environment)

Like: Code Blocks, Visual Studio etc.

- Code

3.2 Procedures:

The following procedures need to be followed step by step-

- Install any IDE (like: Code Blocks, Visual Studio etc..)
- Create a project in **code blocks**
- Write down the code which is given in ***chapter 2.2.***
- Compile and run the code
- Enter the password: **213002039**. If it is first time, then press ENTER button to get into the app and change the password as you wish.
- After logging into app, user will see a landing page with some basic information of the organization. Press ENTER button again.

- Now, a user will see some options for performing tasks (like: ADD student information, MODIFY student information etc..). User have to choose one to perform.
- Once an operation has complete, app will say “*Press any key*”. If user press any key, then app will give all the options again.
- This is all how a user can perform with this app.
- At last, a user can logout from the app.

[DON'T WORRY! YOUR DATA HAS BEEN SAVED ENCRYPTED IN PROJECT FOLDER IN A TEXT FILE NAMED “*db.txt*”.]

CHAPTER 4: TERMINOLOGIES

4.1 IDE (Integrated Development Environment)

Integrated development environments (IDE) are applications that facilitates the development of other applications. Designed to encompass all programming tasks in one application, one of the main benefits of an IDE is that they offer a central interface with all the tools a developer needs, including:

- **Code editor:** Designed for writing and editing source code, these editors are distinguished from text editors because work to either simplify or enhance the process of writing and editing of code for developers
- **Compiler:** Compilers transform source code that is written in a human readable/writable language in a form that computers can execute.
- **Debugger:** Debuggers are used during testing and can help developers debug their application programs.
- **Build automation tools:** These can help automate developer tasks that are more common to save time.

In addition, some IDEs may also include:

- **Class browser:** Used to study and reference properties of an object-oriented class hierarchy.
- **Object browser:** Used to inspect objects instantiated in a running application program.
- **Class hierarchy diagram:** Allows developers to visualize the structure of object-oriented programming code.

The IDE may be a stand-alone application, though it might also be included as part of one or more compatible applications.

4.2 Code

In computer programming, *computer code* refers to the set of instructions, or a system of rules, written in a particular programming language (i.e., the source code).

It is also the term used for the source code after it has been processed by a compiler and made ready to run on the computer (i.e., the object code).

4.3 Build and Run

- *Running* is getting some binary executable (or a script, for interpreted languages) to be, well... *executed* as a new *process* on the computer;
- *Compiling* is the process of parsing a program written in some high level language (higher if compared to machine code), checking it's syntax, semantics, linking libraries, maybe doing some optimization, then creating a binary *executable* program as an output. This executable may be in the form of machine code, or some kind of byte code — that is, instructions targeting some kind of *virtual machine*;
- *Building* usually involves checking and providing dependencies, inspecting code, compiling the code into binary, running automated tests and packaging the resulting binary[ies] and other assets (images, configuration files, libraries, etc.) into some specific format of deployable file. Note that most processes are optional and some depend on the targeted platform you are building for. As an example, packaging a Java application for Tomcat will output a .war file. Building a Win32 executable out of C code could just output the .exe program, or could also package it inside a .msi installer.

4.4 Database

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.

Data within the most common types of databases in operation today is typically modeled in rows and columns in a series of tables to make processing and data querying efficient. The data can then be easily accessed, managed, modified, updated, controlled, and organized.

4.5 Encryption

Encryption is a way of scrambling data so that only authorized parties can understand the information. In technical terms, it is the process of converting human-readable plaintext to incomprehensible text, also known as ciphertext. In simpler terms, encryption takes readable data and alters it so that it appears random. Encryption requires the use of a cryptographic key: a set of mathematical values that both the sender and the recipient of an encrypted message agree on.



Although encrypted data appears random, encryption proceeds in a logical, predictable way, allowing a party that receives the encrypted data and possesses the right key to decrypt the data, turning it back into plaintext. Truly secure encryption will use keys complex enough that a third party is highly unlikely to decrypt or break the ciphertext by brute force — in other words, by guessing the key.

Data can be encrypted "at rest," when it is stored, or "in transit," while it is being transmitted somewhere else.

3.6 Data Redundancy

Data redundancy is a condition created within a database or data storage technology in which the same piece of data is held in two separate places.

This can mean two different fields within a single database, or two different spots in multiple software environments or platforms. Whenever data is repeated, it basically constitutes data redundancy.

Data redundancy can occur by accident but is also done deliberately for backup and recovery purposes.

CHAPTER 5: FEATURES AND LIMITATIONS

5.1 Features

- Security (Includes: login, logout)
- Landing page
- Easy option choosing
- Adding student information
- Modifying student information
- Display all student information
- Display Individual information
- Removing student information
- Encrypted database
- CGPA calculator
- Password changing

5.2 Limitation

- Can only store few information
- Interface is not smart
- No reduction system existing for data redundancy
- No student's finance system
- No student's health record system
- No transcript system

CHAPTER 6: PERFORMANCE EVALUATION

6.1 Performance Evaluation

- ✓ Security successfully implemented, especially the login and logout system.
- ✓ Option choosing perfectly working
- ✓ Adding student information perfectly working
- ✓ Modifying student information perfectly working
- ✓ Displaying all student information together perfectly working
- ✓ Removing student's information perfectly working
- ✓ Database is encrypted successfully
- ✓ CGPA is giving correctly
- ✓ Password changing is successfully working

- Storing big amount of data is unavailable
- Interface is not so smart
- There is no data redundancy system for database
- There is no student's billing and account system found
- There is no student's health record system found
- There is no transcript system found

6.2 Result and Discussion

The student management system is working fine. For implementing this project I have to use codeblocks as IDE, write codes, setup all security systems, setup database system, setup all tasks which will be performed while using, setup encryption system etc.. There are so many limitations exist in this program, but still it's fine to work.

CHAPTER 7: CONCLUSION

7.1 Conclusion

Our project is only a humble venture to satisfy the needs in an Institution. Several user-friendly coding has also adopted. This package shall prove to be a powerful package in satisfying all the requirements of the organization. The objective of software planning is to provide a frame work that enables the manger to make reasonable estimates made within a limited time frame at the beginning of the software project and should be updated regularly as the project progresses.

7.2 Recommendation and Future Plan

1. Add more field to store information
2. making the interface smart
3. add data redundancy detecting system
4. adding student's finance system
5. adding student's health record
6. adding transcript system

REFERENCES

- [1] Student management system from <https://itsourcecode.com/fyp/student-management-system-project/>
- [2] Student management system from <https://www.studocu.com/row/document/riphah-international-university/database/student-management-system-project-report/12309562>
- [3] IDE from <https://www.veracode.com>
- [4] Code from <https://www.techtarget.com/whatis/definition/code>
- [6] Database from <https://www.oracle.com/database/what-is-database/>
- [7] Encryption from <https://www.cloudflare.com/learning/ssl/what-is-encryption/>
- [8] Data Redundancy from <https://www.techopedia.com/definition/18707/data-redundancy>

WORD COUNT: 3097

[END]