School of Computing
Faculty of Engineering
UNIVERSITI TEKNOLOGI MALAYSIA

DATA STRUCTURE & ALGORITHM
(SECJ 2013)

SEMESTER 1 2021/2022

Mini Project Documentation

FINAL YEAR PROJECT MANAGEMENT SYSTEM

By
Felicia Chin Hui Fen (A20EC0037) - Leader
Gui Yu Xuan (A20EC0039)
Saidah Binti Saiful Bahari (A20EC0141)

SECTION 01

Lecturer:
Ts Dr Johanna binti Ahmad

25 January 2022

1. INTRODUCTION

1.1. SYNOPSIS PROJECT

The Final Year Project System is a system in which users check the final year projects of the students. The Final Year Project System stores the information of the students and supervisor in a text file to facilitate the final year project process. The design is simple because it provides a clear interface for users to understand. The Final Year Project System consists of two users, which are supervisor and students. Students use the Final Year Project System to check the marks of the project. The supervisor uses the Final Year Project System to enter marks and comments for the students' final year projects. The system will list down the students' information such as name, id, course, section and project name to the supervisor for them to key in the marks and comment. For students, the system will list down the overall information of all students such as students' name, id, course, section, project name, marks, grade and comment for the student to view.

The data structure applied in this system is the linked list and queue. We perform the linked list concept in the queue. For the linked list, we have implemented two linked lists to link two types of student information. Firstly, we used student class pointers, "frontPtr" and "backPtr" in the class queue to link all the student information. Other than that, we used "head" and "tail" pointers to link the students' login information (students' name, id and password) in the class queue. In order to link to the next pointer, we assigned a pointer named "next" in the student class. Almost all the functions in the system have applied the linked list concept. For example, the insertQueue function had used the linked list concept to store the students' information in the queue with the idea of First-In-First-Out.

For the queue, we have used the insertQueue function to store all student information and display function to display all the information of the student. The First-In-First-Out concept is applied, where the first student information will be stored in the queue and it will be displayed first on the screen.

Our system is able to perform several functions. This included insert, delete, search and display. For the insert function, our system is able to insert the students' information in the queue. The system also provides a feature for supervisors to insert marks and comment to the students based on their information that has been stored in the queue.

For the delete function, our system has provided two functions for it. The first one is the supervisor able to delete the students' information in this system. After the student has been deleted from this system, the student is unable to login again in this system. The second one is delete marks, the supervisor able to delete the marks of the students.

For the search function, we applied the sequential search concept in our system. There are two types of search functions in the system. One is the checkValidity function, where this function is used for checking whether the users have accessibility to the system or not. If the user does not have a recorded id and password in the system, they are unable to access the system. The second search function is the searchId function where the users are able to search the students; information based on the students' id.

The last function is the display function, where this function is used to display all the students' information in the queue based on the First-In-First-Out concept.

1.2. OBJECTIVE OF THE PROJECT

The objective of the Final Year Project System is to help the users to access the final year projects. For supervisors, they are able to check the final project of the students. Supervisors are also able to insert marks and comments to the final year project of the students. The system will help to calculate the grade of the students based on their final year project. Beside that, for students, they are able to check their marks and comments for their final year project when they are using the system.

2. SYSTEM ANALYSIS AND DESIGN

2.1. CASE DIAGRAM

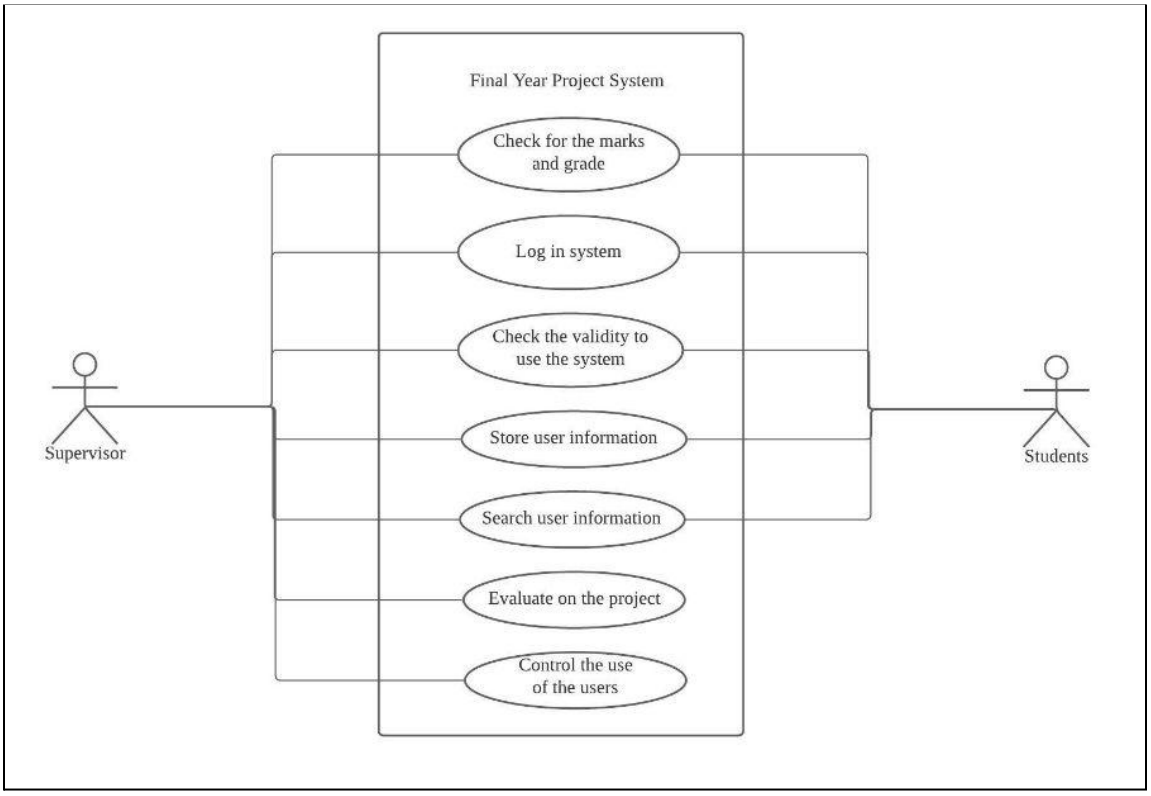


Figure 1: Use Case Diagram for Final Year Project Management System

2.1.1. DETAILS DESCRIPTION

The system users are supervisor and student.

User	Task
Supervisor	● Log in the system by entering the id and password. ● Evaluating the project that had been done by students. ● Searching based on student id to check the grade and marks of students ● Deleting student information to deny students access to the system
Student	● Log in the system by entering the id and password ● Searching based on student id to check the grade and marks of students

The system has 7 main use cases.

Use Case	Purpose
Check for the marks and grade	Users can check for the marks and grade of the students by displaying the overall student information or by searching based on student's id.
Log in system	Users need to log in the system by entering the id and the password.
Check the validity to use the system	The entering id and password will compare with the id and the password that had been stored in the data file. If the id and password entered by users is

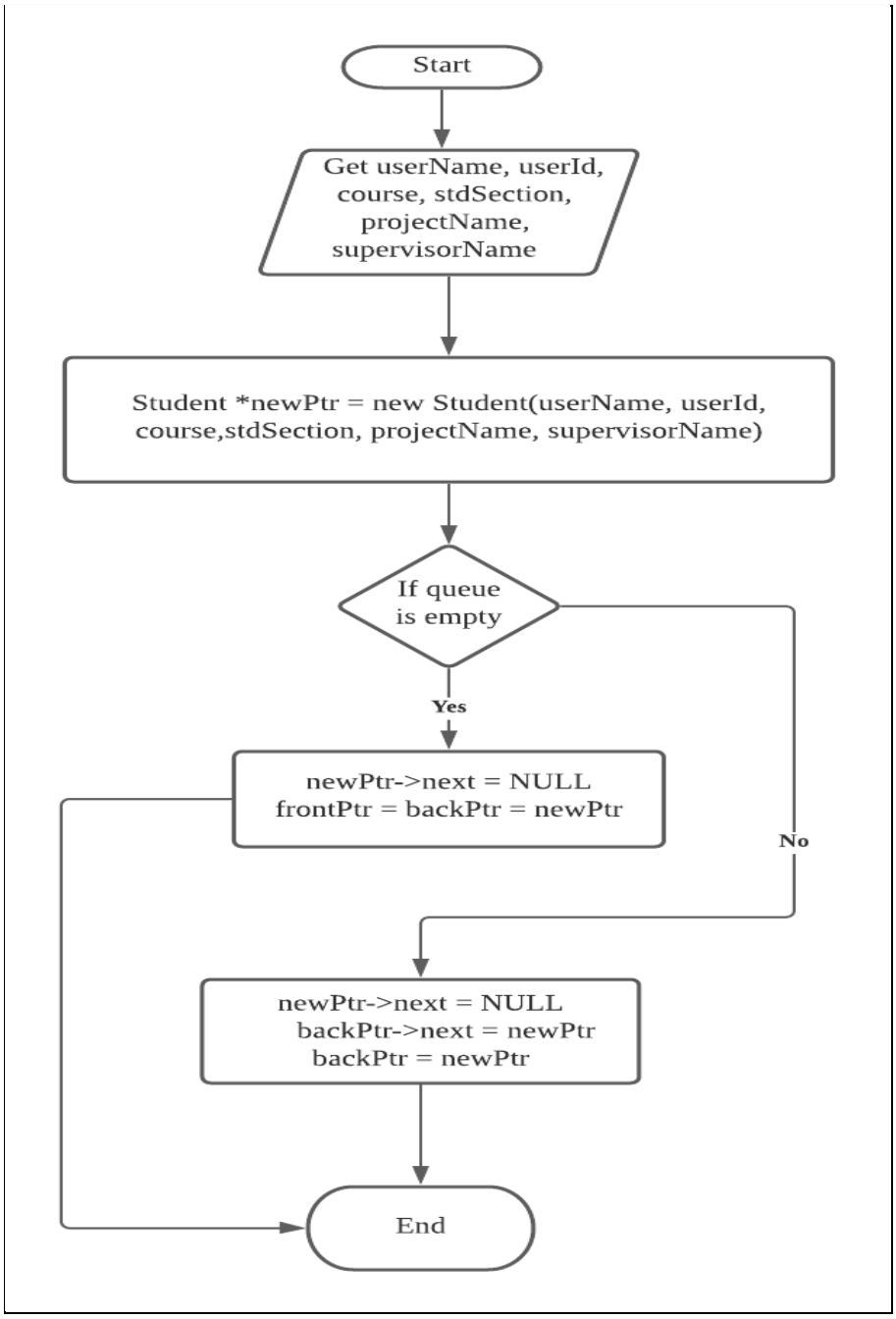
	the same as in the data file, then users are valid to enter the system, otherwise users cannot use the system.
Store user information	The id and the password of users will be stored so that it can be used to check the validity of users to use the system. Next, the student information which contains the student's name, student's id, student's project and more will be stored in the system so that the supervisor is able to evaluate the student's project.
Search user information	Supervisor and student are able to perform searching techniques by entering the student's id to view the result of the student.
Evaluate on the project	Supervisor able to insert the project mark and type the comment on the project that had been done by students.
Control the use of the users	Supervisor able to delete the student's id and password that had been stored in the system so that the system will deny the use of the student.

2.2. FLOWCHART

2.2.1. INSERT QUEUE FUNCTION

Flowchart 1: Insert student information in the queue

Prepared By: Felicia Chin Hui Fen

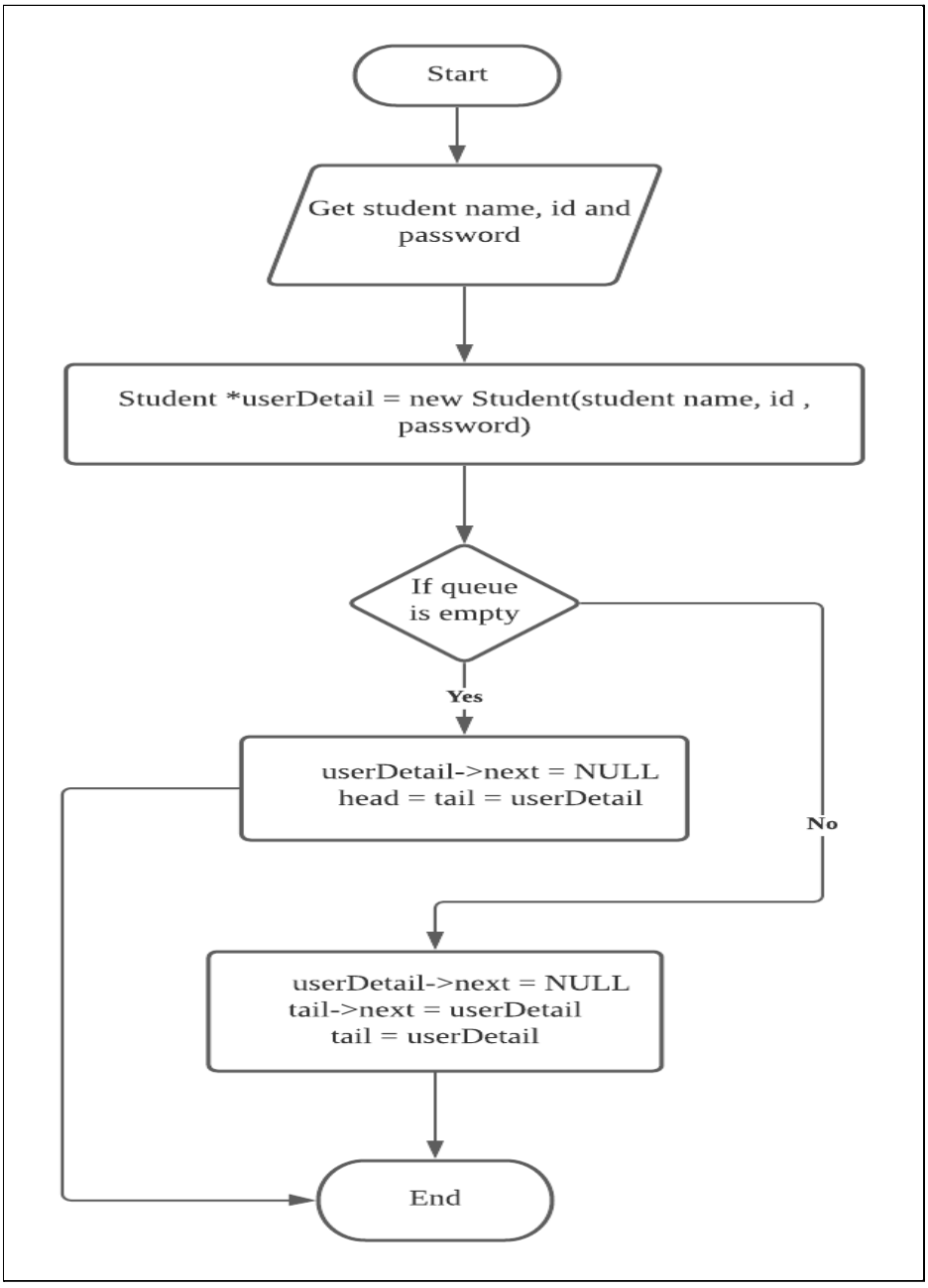


Flowchart 1: Insert student information in the queue

2.2.2. INSERT STUDENT INFO FUNCTION

Flowchart 2: Insert student login information in the queue

Prepared by: Felicia Chin Hui Fen

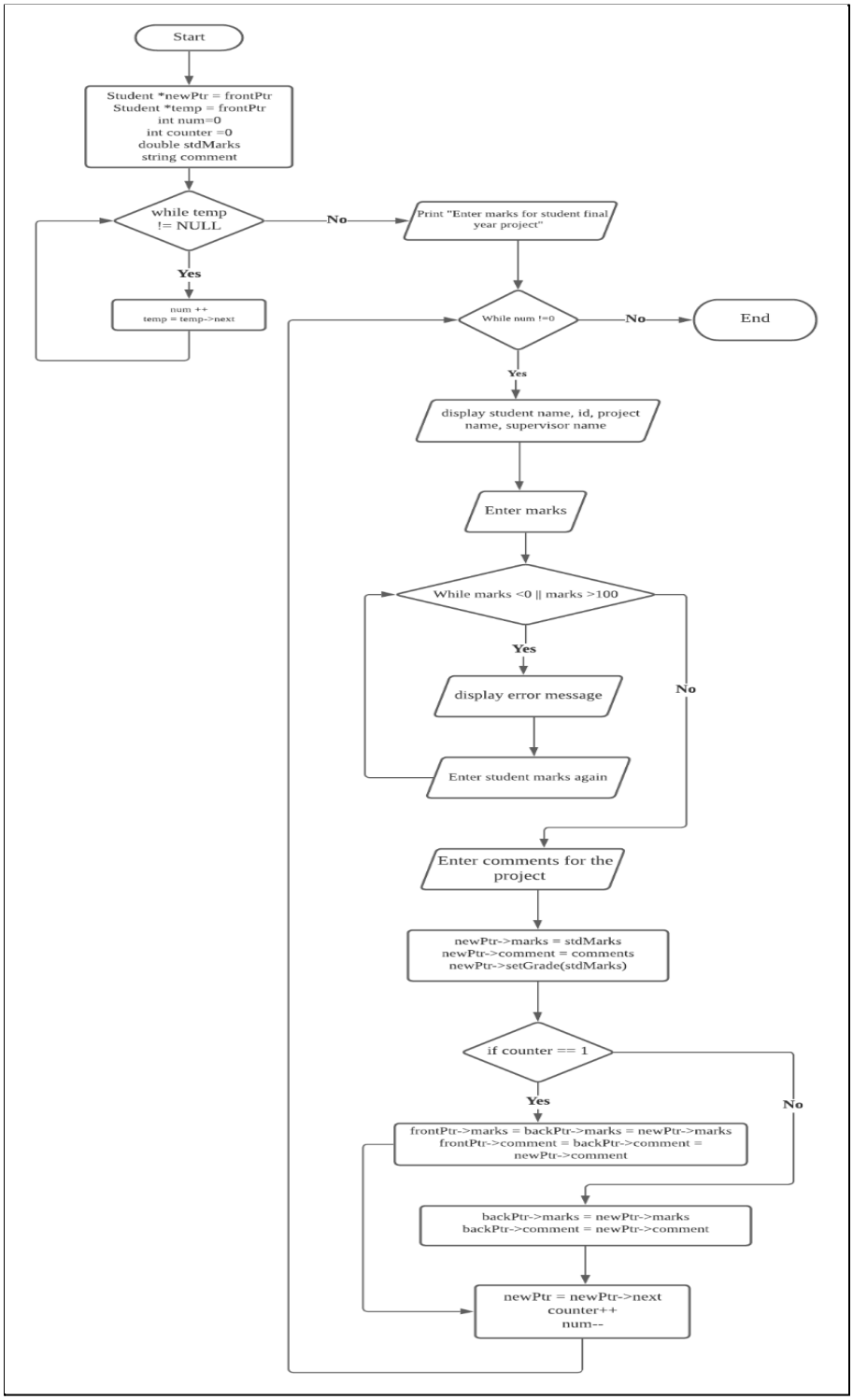


Flowchart 2: Insert student login information in the queue

2.2.3. INSERT MARKS FUNCTION

Flowchart 3: Insert student final year project's marks

Prepared by: Felicia Chin Hui Fen

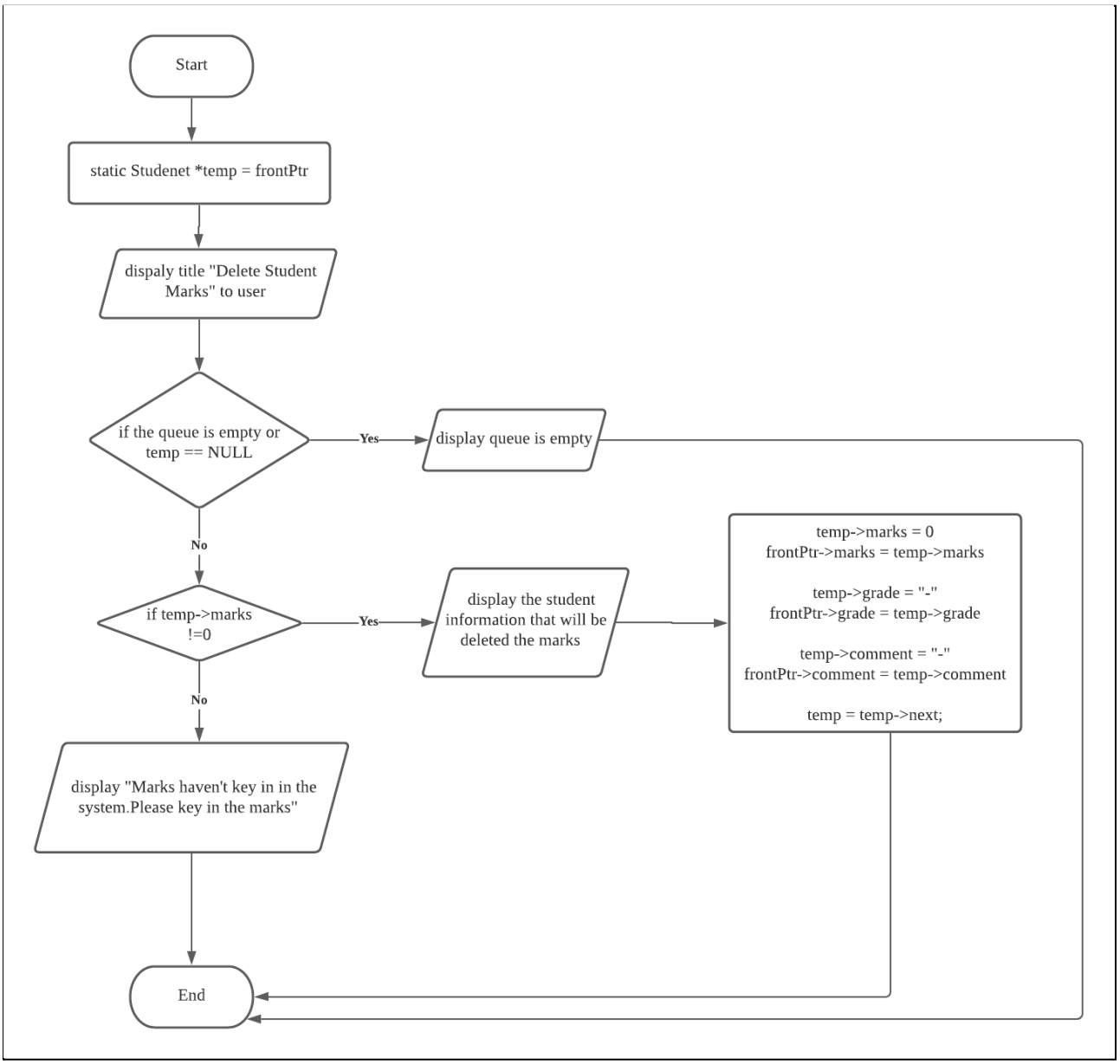


Flowchart 3: Insert student final year project's marks

2.2.4. DELETE MARKS FUNCTION

Flowchart 4: Delete Marks Function

Prepared By: Saidah Binti Saiful Bahari

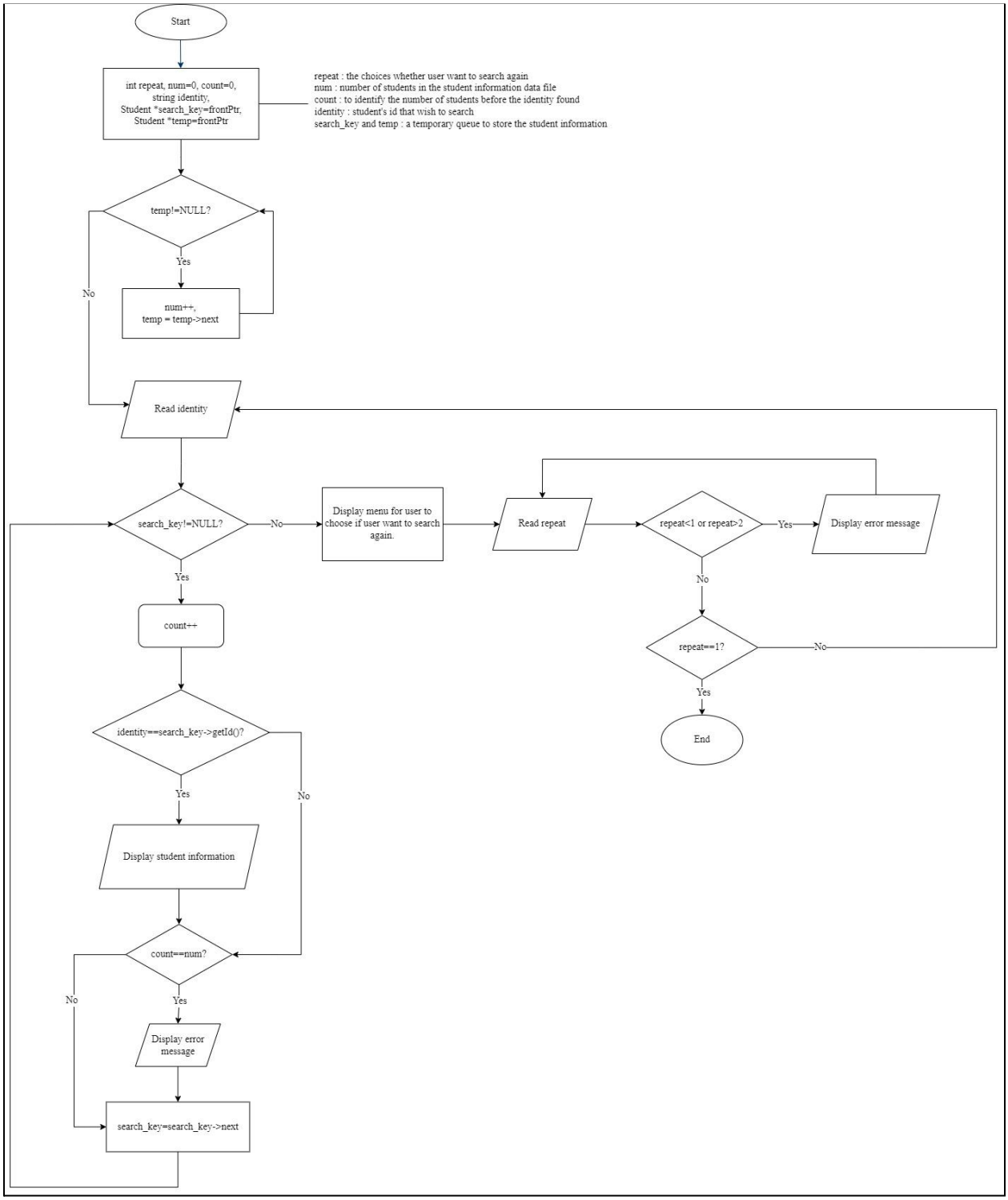


Flowchart 4: Delete Marks Function

2.2.5. SEARCH MARKS FUNCTION

Flowchart 5: Search Based On Id

Prepared By: Gui Yu Xuan

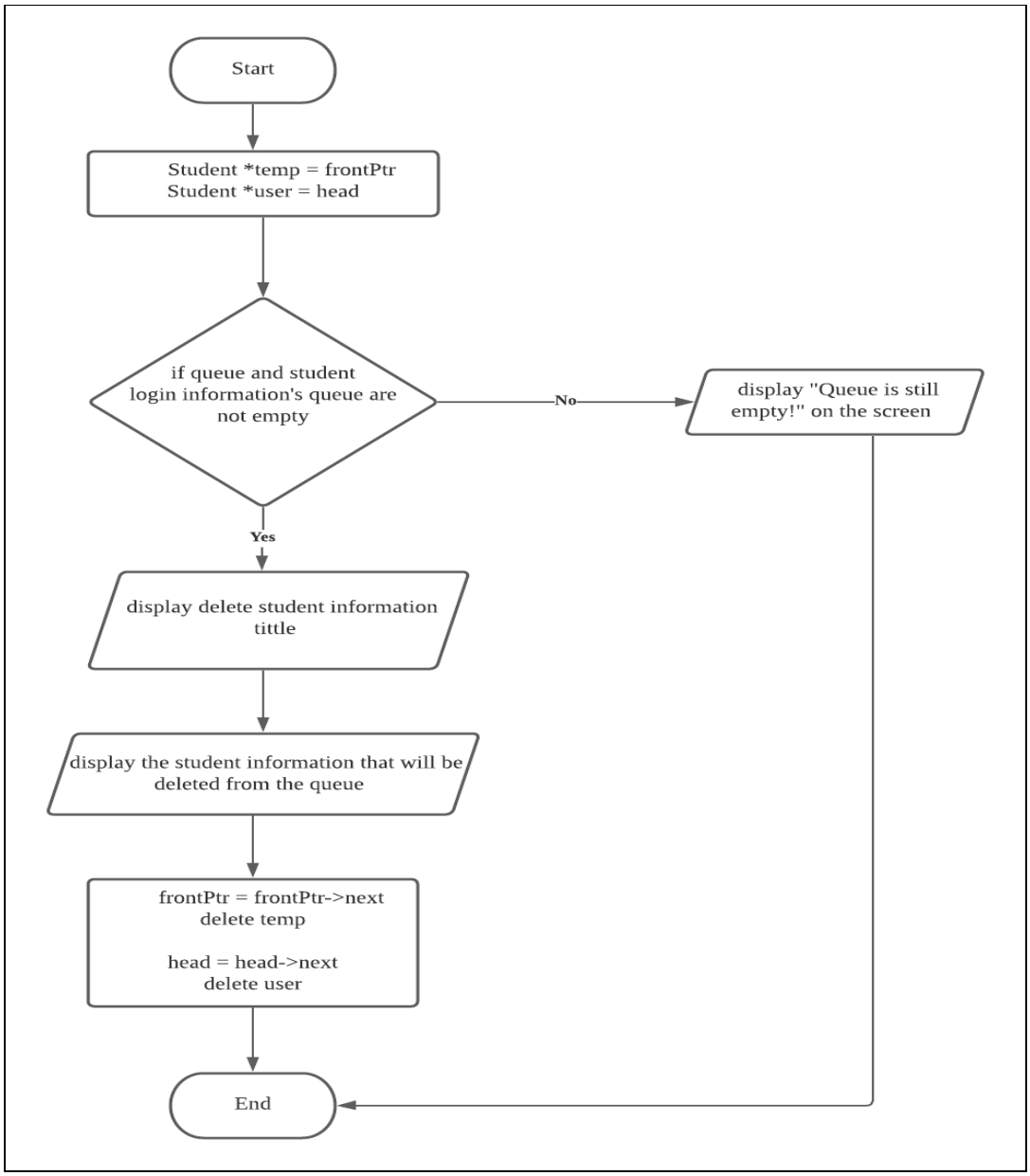


Flowchart 5: Search Based On Id

2.2.6. DELETE STUDENT INFORMATION FUNCTION

Flowchart 6: Delete student information

Prepared by: Saidah binti Saiful Bahari

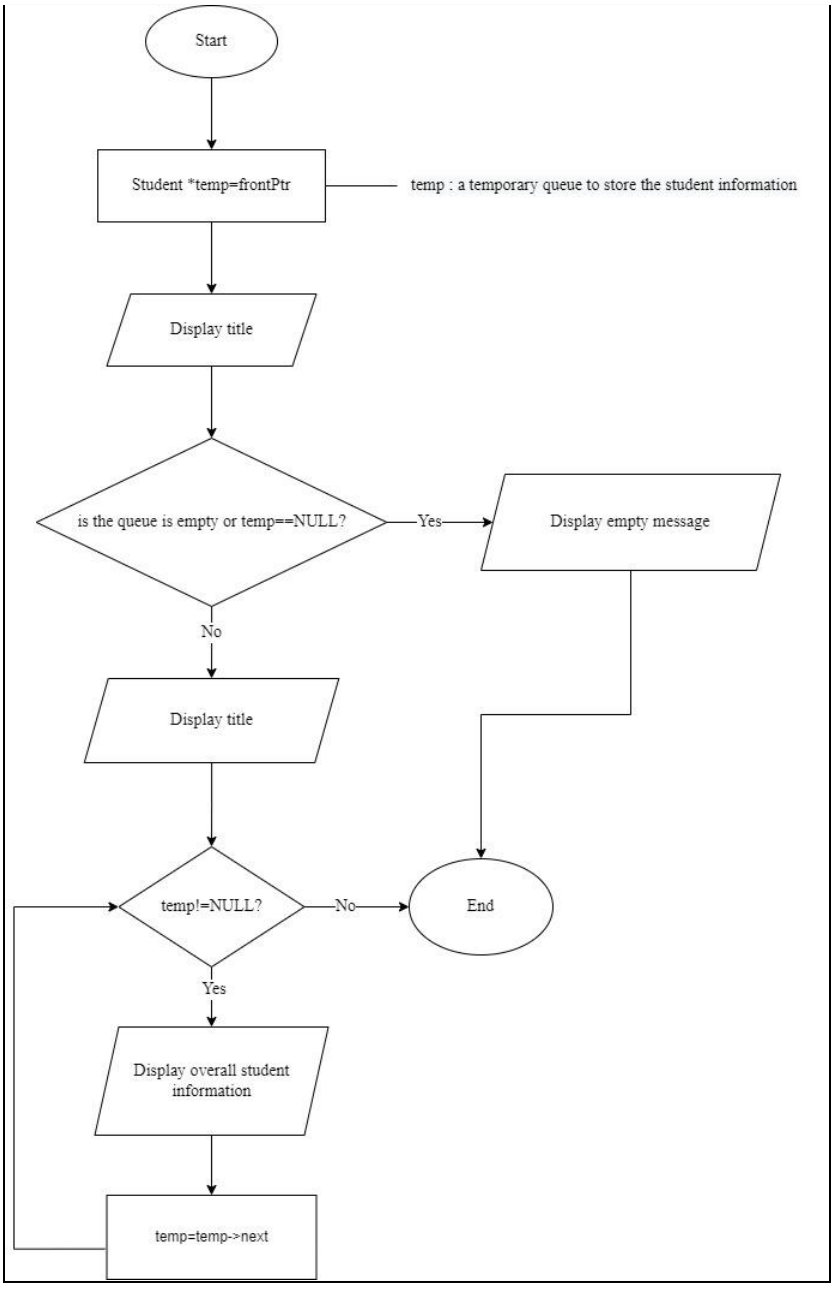


Flowchart 6: Delete student information

2.2.7. DISPLAY FUNCTION

Flowchart 7: Display Student Information

Prepared By: Gui Yu Xuan



Flowchart 7: Display student information

2.3. CLASS DIAGRAM

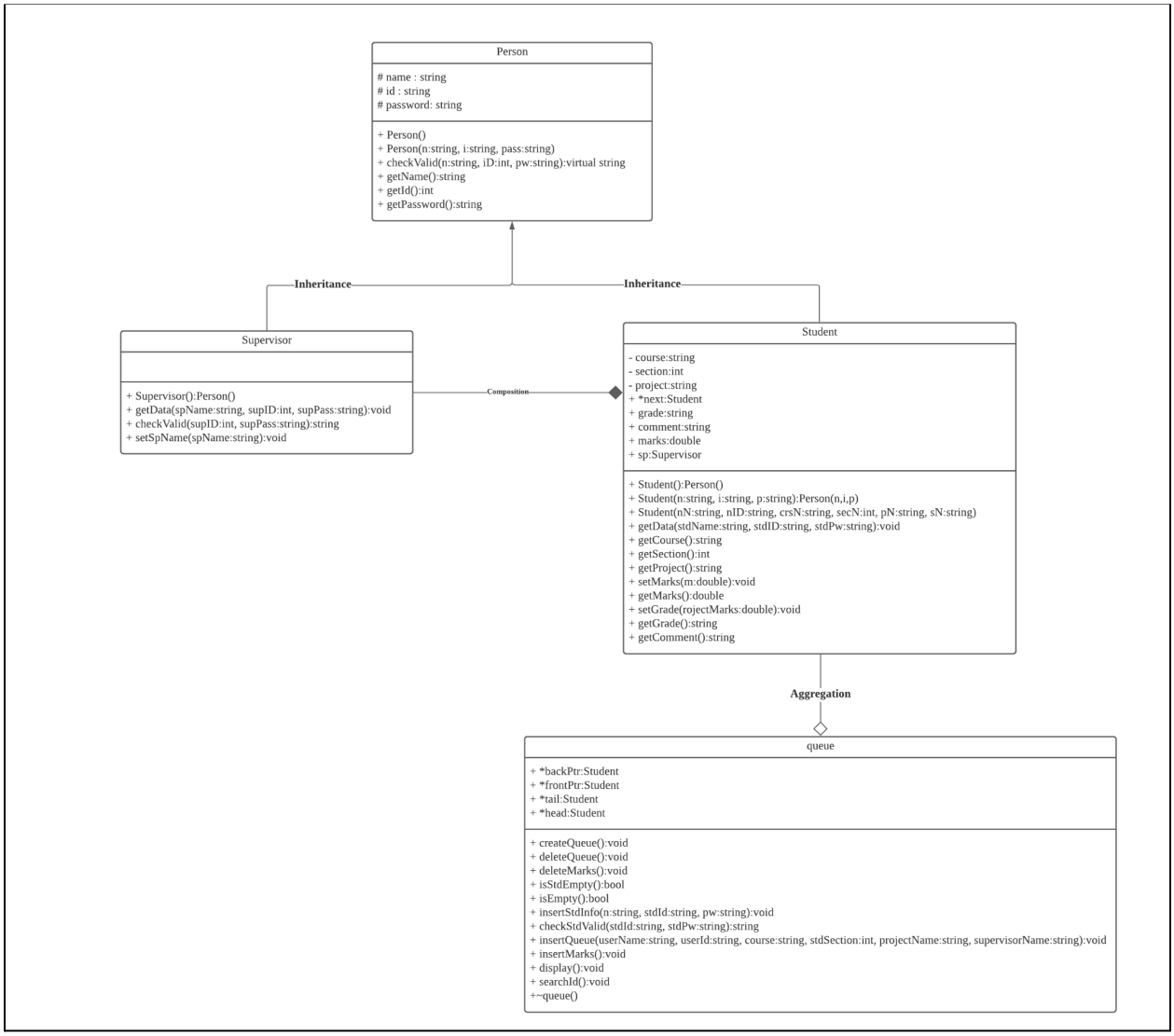
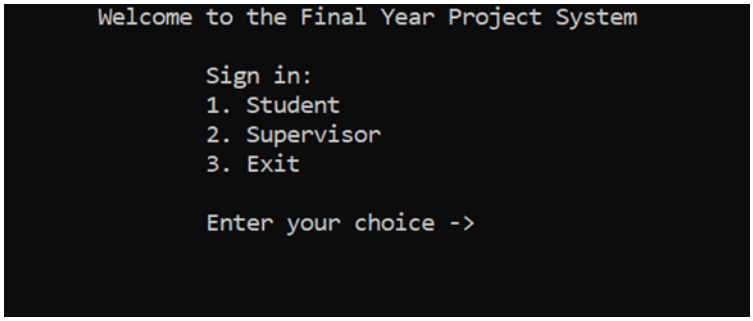


Figure 8: UML diagram of Final Year Project system

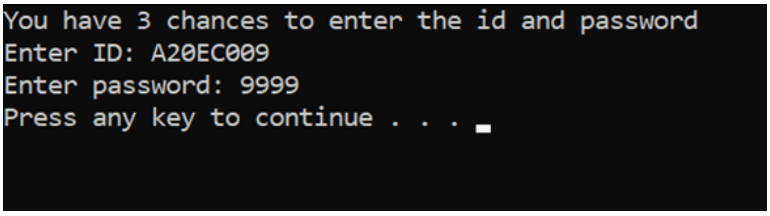
3. SYSTEM PROTOTYPE



Screen 1: Main Menu for role of users

Screen 1: The user must insert integer values in the range 1-3. If the user enters another number, the system will prompt an error message and it will ask to enter another value again.

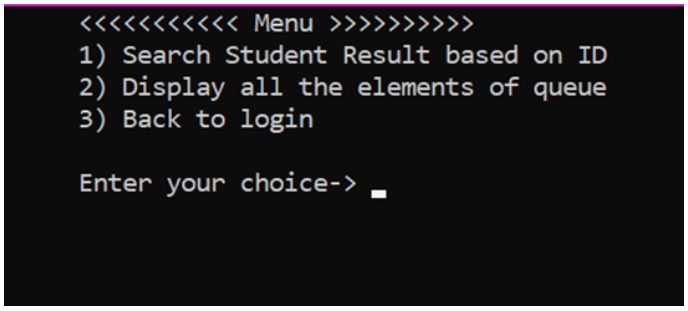
Prepared by: Saidah binti Saiful Bahari



Screen 2*: Students log in*

Screen 2: If user enters value 1 in screen 1, hence user will log into the system as students. Users need to key in the right ID and password. If the user enters the wrong ID and password, the system will prompt an error message and the screen is displayed again.

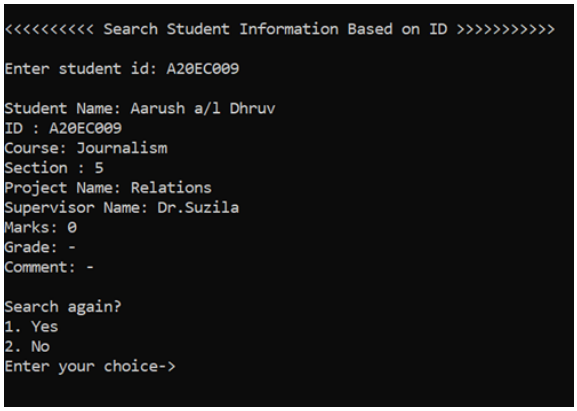
Prepared by: Gui Yu Xuan



Screen 3*: Menu in Student's view*

Screen 3: After user log in to the system as students, user must insert an integer value in the range 1-3. If the user enters another number, the system will prompt an error message and it will ask to enter another value again.

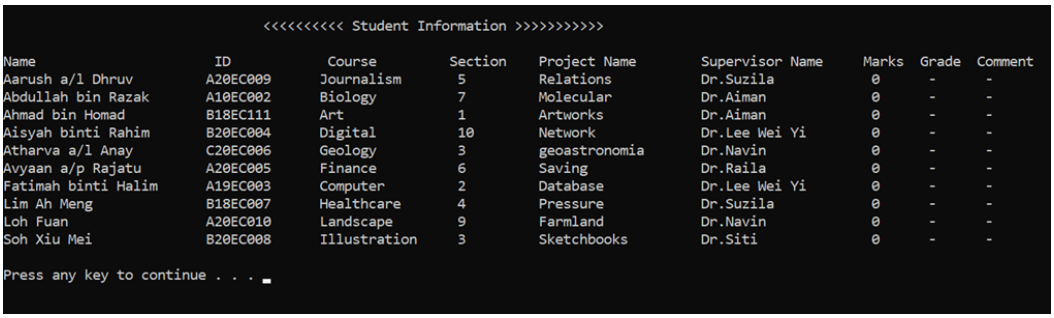
Prepared by: Saidah Binti Saiful Bahari



Screen 4: Search students result based on ID

Screen 4: If user key in value 1 which is search students result based on ID, students need to key in their ID to check their marks.

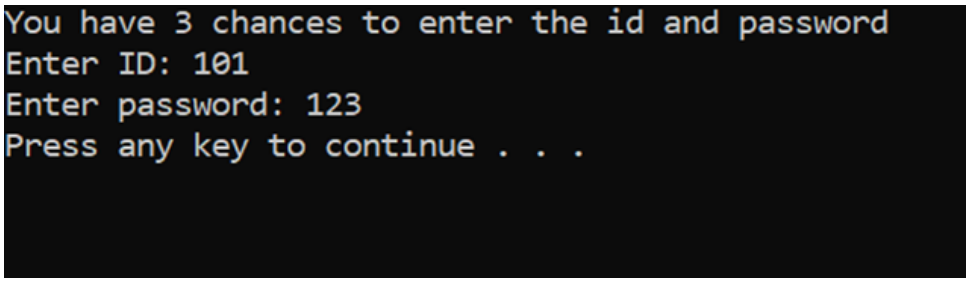
Prepared by: Gui Yu Xuan



Screen 5: Display all the elements of queue

Screen 5: If students key in value 2 as shown on screen 3, screen 5 will display all the elements of the queue.

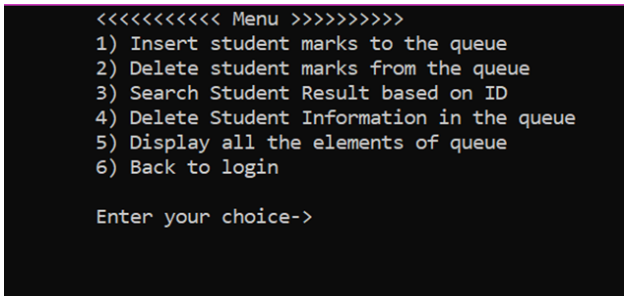
Prepared by: Gui Yu Xuan



Screen 6: Supervisor log in

Screen 6: Screen 6 is where the user will log in to the system as supervisor. Users need to key in the right ID and password. If the user enters the wrong ID and password, the system will prompt an error message and the screen is displayed again.

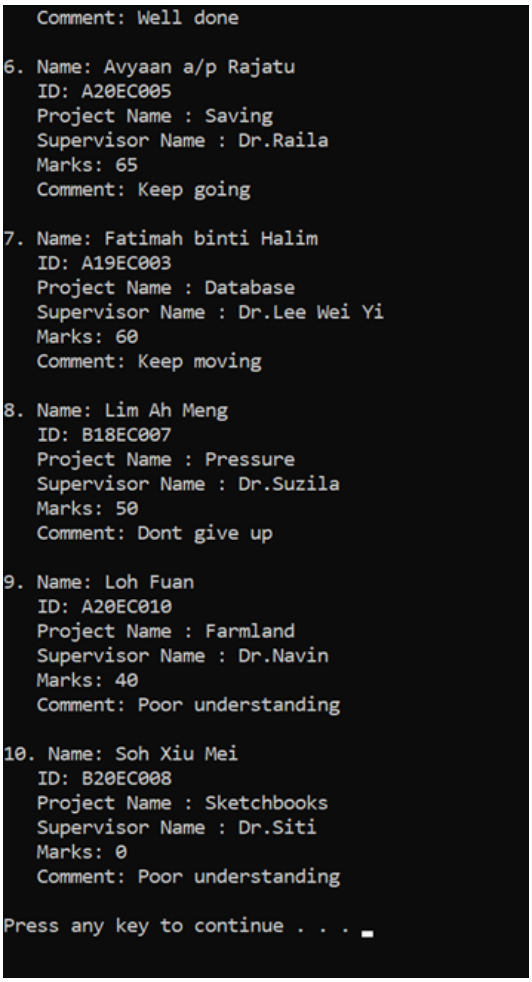
Prepared by: Felicia Chin Hui Fen



Screen 7: Menu in Supervisor's view

Screen 7: Screen 7 shows menu for supervisor. Users need to key in integer value in the range 1-6. If the user enters another number, the system will prompt an error message and ask the user to enter another integer value.

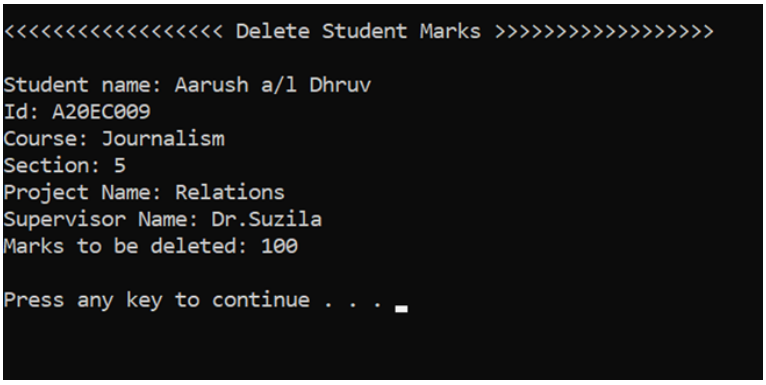
Prepared by: Saidah Binti Saiful Bahari



Screen 8: Insert student marks to the queue

Screen 8: If choice 1 is chosen, then the supervisor needs to key in students' marks and also comment.

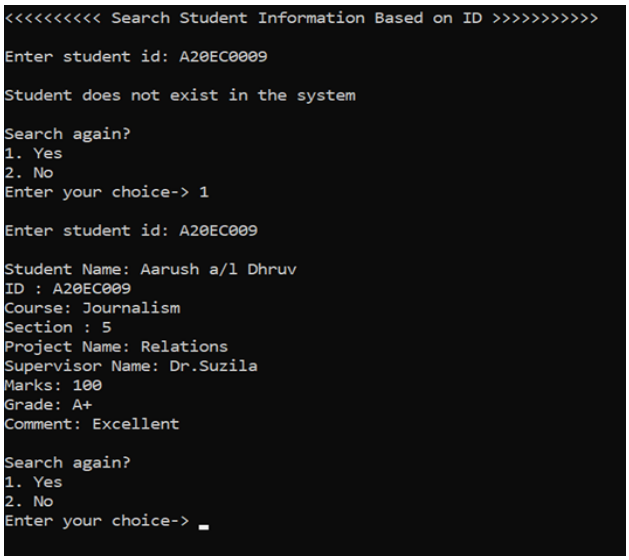
Prepared by: Felicia Chin Hui Fen



Screen 9: Delete student marks from the queue

Screen 9: When the choice 2 is chosen, the screen shows the user has deleted the mark of the first student from the queue.

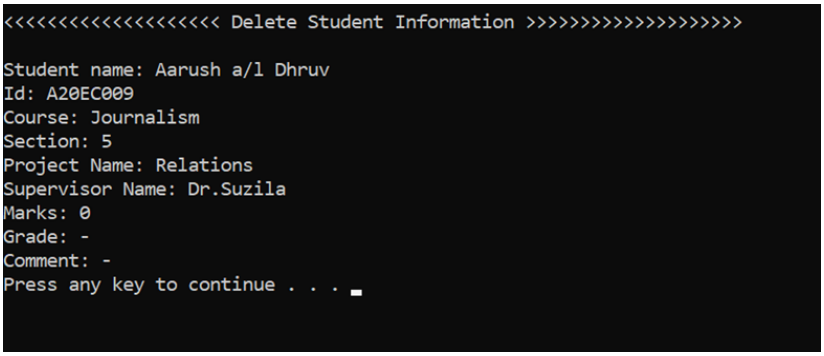
Prepared by: Saidah Binti Saiful Bahari



Screen 10: Search students result based on ID

Screen 10: When choice 3 is chosen on screen 7. Users must insert a student's ID in order to search student information. The system will prompt an error message if the id that had been entered is not found in the queue and the screen is displayed again.

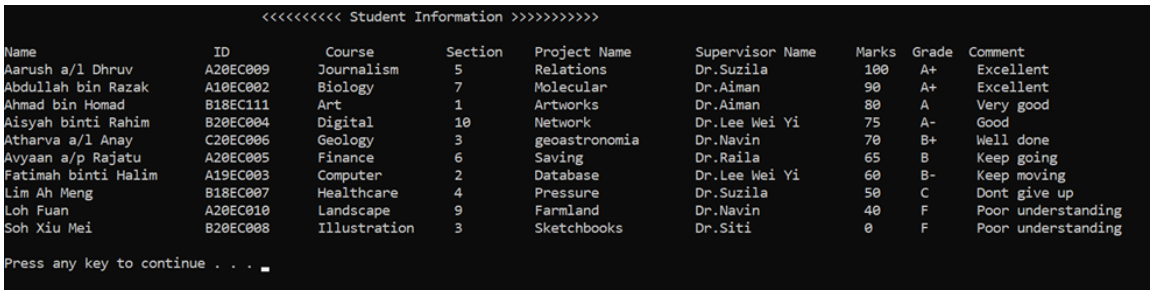
Prepared by: Gui Yu Xuan



Screen 11: Delete Students Information in the queue

Screen 11: When choice 4 is chosen on screen 7. This screen shows the user has deleted the information of the first student from the queue.

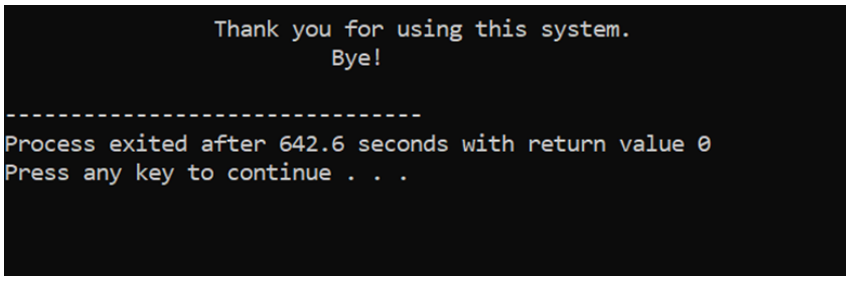
Prepared by: Saidah Binti Saiful Bahari



Screen 12: Display all elements of the queue

Screen 12: When choice 5 is chosen on screen 7. The system will display all students' information along with the marks that have been key in earlier.

Prepared by: Gui Yu Xuan



Screen 13: Exit

Screen 13: The user has exit from the system if the user chooses 3 on screen 1.

Prepared by: Saidah Binti Saiful Bahari

4. DEVELOPMENT ACTIVITIES

Meeting Date	Members Participate in the meeting	Activity	Task for each member	Task Achieved (yes/no)
14/1/2022	Felicia Chin Hui Fen	❖ Discussing the function needed for the system ❖ Discuss the interface for the system ❖ Divide the task for the function used in the system ❖ Divide the task for the report	➔ Decorate the interface ➔ Create menu and insert marks function ➔ Compile the code	yes
	Gui Yu Xuan		➔ Create search marks function ➔ Create display function	yes
	Saidah		➔ Create delete marks function ➔ Create delete student information marks	yes
18/1/2022	Felicia Chin Hui Fen	❖ Interim checking ❖ Solve the problems together	➔ Checking for the errors ➔ Discuss the problems met	yes
	Gui Yu Xuan			
	Saidah			
19/1/2022	Felicia Chin Hui Fen	❖ Meeting with Dr Johanna		yes

	Gui Yu Xuan Saidah			
23/1/2022	Felicia Chin Hui Fen Gui Yu Xuan Saidah	❖ Finalize the system ❖ Starting to do report	➔ Checking for the errors ➔ Discuss the problems met	yes

5. APPENDIX

5.1. Drive Link

Data File and Source Code

5.2. nIDPass File

Name	ID	Password
Aarush a/l Dhruv	A20EC009	9999
Abdullah bin Razak	A10EC002	2222
Ahmad bin Homad	B18EC111	1111
Aisyah binti Rahim	B20EC004	4444
Atharva a/l Anay	C20EC006	6666
Avyaan a/p Rajatu	A20EC005	5555
Fatimah binti Halim	A19EC003	3333
Lim Ah Meng	B18EC007	7777
Loh Fuan	A20EC010	0000
Soh Xiu Mei	B20EC008	8888

5.3. superData File

ID	Name	Password
101	Dr.Aiman	123

5.4. studentInfo File

File Edit Format View Help

Name	ID	Course	Section	Project Name	Supervisor Name
Aarush a/l Dhruv	A20EC009		Journalism	5	Relations Dr.Suzila
Abdullah bin Razak	A10EC002		Biology	7	Molecular Dr.Aiman
Ahmad bin Homad	B18EC111		Art	1	Artworks Dr.Aiman
Aisyah binti Rahim	B20EC004		Digital	10	Network Dr.Lee Wei Yi
Atharva a/l Anay	C20EC006		Geology	3	geoastronomia Dr.Navin
Avyaan a/p Rajatu	A20EC005		Finance	6	Saving Dr.Raila
Fatimah binti Halim	A19EC003		Computer	2	Database Dr.Lee Wei Yi
Lim Ah Meng	B18EC007		Healthcare	4	Pressure Dr.Suzil
Loh Fuan	A20EC010	Landscape		9	Farmland Dr.Navin
Soh Xiu Mei	B20EC008		Illustration	3	Sketchbooks Dr.Siti

5.5. Source Code

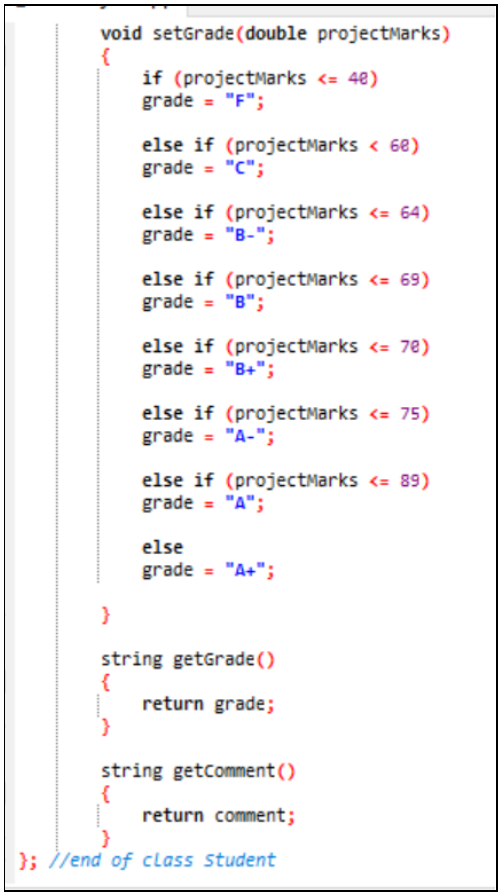
```
//*****************************************************************************
//           Mini Project
//
//  Title: Final Year Project System
//  Lecturer: Ts Dr Johanna binti Ahmad
//  Section: 01
//  Semester : 2021/2022 - 1
//  Group 6 Members:
//          1. Felicia Chin Hui Fen (A20EC0037)
//          2. Gui Yu Xuan (A20EC0039)
//          3. Saidah Binti Saiful Bahari (A20EC0141)
//
//*****************************************************************************

#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>
#include <unistd.h>
#include <ctime>
#include <sstream>
#include <cstdlib>
#include<limits>
#define MAXSTREAM numeric_limits<streamsize>::max()

using namespace std;

//declare global variable
//declare for data file
ifstream infile, inputFile, inp;
//infile - student file
//inputFile - supervisor file
//inp - student information file
```

```
class Person
{
    protected:
        string name;        // name of person
        string id;          // id of person
        string password;    //password of person

    public:
        //constructor
        Person(){   }
        Person(string n, string i, string pass )
        {
            name = n;
            id = i;
            password = pass;
        }

        //check for the validity of users to enter
        virtual string checkValid(string n, string iD, string pw)
        {
            string validity= "Invalid";
            if(iD == id)
            {
                if(pw == password)
                {
                    validity = "Valid";
                }
            }
            return validity;
        }

        //getter
        string getName(){ return name;}
        string getId(){ return id;}
        string getPassword(){ return password;}

};//end of class Person declaration
```

```
//declaration of class Supervisor
class Supervisor : public Person
{
    public:
        //constructor
        Supervisor():Person(){      }


        //to set the value of the variable
        void getData(string spName, string supID, string supPass)
        {
            name = spName;
            id = supID;
            password = supPass;
        }

        //the function is virtual to the checkValid function in class Person
        //check the validity of the supervisor whether can enter to the system or not
        string checkValid(string supID, string supPass)
        {
            string validity = "Invalid";
            //if supID (id enter by the supervisor) is same as data stored in data file
            if(supID == id)
            {
                //if supPass (password enter by the supervisor) is same as data stored in data file
                if(supPass == password )
                {
                    validity = "Valid";
                }
            }
            return validity;
        }

        //to set the name of supervisor
        void setSpName(string spName)
        {
            name =  spName;
        }

}; //end of class Supervisor
```

```
//Declaration of class Student
class Student : public Person
{
    private:
        string course;          //course that student enrolled in
        int section;            //section that students enrolled in
        string project;         //proejct title done by students


    public:
        Student *next;          //variable to store student information in node
        string grade;           //grade of the project
        string comment;         //comment made by supervisor
        double marks;           //marks of the project
        Supervisor sp;          //composite with class Supervisor to get the supervisor name and able to process of marking and grading

        //constructor
        Student():Person(){     }
        Student(string n, string i, string p):Person(n,i,p){    }
        //set the variable of student name, student id, course, section, project, supervisor, marks, grade and comment
        //used to store in the node
        Student(string nN, string nID, string crsN, int secN, string pN, string sN)
        {
            name = nN;
            id = nID;
            course= crsN;
            section = secN;
            project = pN;
            sp.setSpName(sN);   //call the function to set the supervisor name
            grade = "-";
            comment = "-";
        }
```

```
//used to store in the node
void getData(string stdName, string stdID, string stdPw)
{
    name = stdName;
    id = stdID;
    password = stdPw;
}

//getter
string getCourse()
{
    return course;
}

int getSection()
{
    return section;
}

string getProject()
{
    return project;
}

void setMarks(double m)
{
    marks = m;
}

double getMarks()
{
    return marks;
}
```



```
        void setGrade(double projectMarks)
        {
            if (projectMarks <= 40)
            grade = "F";

            else if (projectMarks < 60)
            grade = "C";

            else if (projectMarks <= 64)
            grade = "B-";

            else if (projectMarks <= 69)
            grade = "B";

            else if (projectMarks <= 70)
            grade = "B+";

            else if (projectMarks <= 75)
            grade = "A-";

            else if (projectMarks <= 89)
            grade = "A";

            else
            grade = "A+";

        }

        string getGrade()
        {
            return grade;
        }

        string getComment()
        {
            return comment;
        }
};  //end of class Student
```

```
//Declaration of class queue
class queue{
    public:
        //Associate with class Student to store the student information in node
        Student *backPtr, *frontPtr;
        //Associate with class Student to store the student file in node
        Student *tail, *head;

        //create the queue for the information to be stored
        void createQueue()
        {
            backPtr = NULL;
            frontPtr = NULL;
            tail = NULL;
            head = NULL;
        }

        //delete the elements in queue (student file) from beginning
        void deleteQueue()
        {
            Student *temp = frontPtr;
            Student *user = head;

            if(! isEmpty() && ! isStdEmpty())
            {
                cout << "\n<<<<<<<<<<<<<<<<<<<< Delete Student Information >>>>>>>>>>>>>>>>>>>>\n\n";
                cout << "Student name: " << temp->getName() << endl;
                cout << "Id: " << temp->getId() << endl;
                cout << "Course: " << temp->getCourse() << endl;
                cout << "Section: " << temp->getSection() << endl;
                cout << "Project Name: " << temp->getProject() << endl;
                cout << "Supervisor Name: " << temp->sp.getName() << endl;
                cout << "Marks: " << temp->getMarks() << endl;
                cout << "Grade: " << temp->getGrade() << endl;
                cout << "Comment: " << temp->getComment() << endl;


                frontPtr = frontPtr->next;
                delete temp;

                head = head->next;
                delete user;

            }

            else
            {
                cout << "\nQueue is still empty!" << endl;
            }
        }
```

```
//delete the marks of student that stored in the queue from beginning
void deleteMarks()
{
    static Student *temp = frontPtr;

    cout << "\n<<<<<<<<<<<<<<<<<< Delete Student Marks >>>>>>>>>>>>>>>>>>\n\n";
    if(isEmpty() || temp == NULL)
    {
        cout << "Queue is empty" << endl;
    }

    else if(temp->marks != 0)
    {
        cout << "Student name: " << temp->getName() << endl;
        cout << "Id: " << temp->getId() << endl;
        cout << "Course: " << temp->getCourse() << endl;
        cout << "Section: " << temp->getSection() << endl;
        cout << "Project Name: " << temp->getProject() << endl;
        cout << "Supervisor Name: " << temp->sp.getName() << endl;
        cout << "Marks to be deleted: " << temp->getMarks() << endl << endl;

        temp->marks = 0;
        frontPtr->marks = temp->marks;

        temp->grade = "-";
        frontPtr->grade = temp->grade;

        temp->comment = "-";
        frontPtr->comment = temp->comment;

        temp = temp->next;
    }

    else
    {
        cout << "\nMarks haven't key in in the system.\nPlease key in the marks first\n" << endl;
    }
}
```

```
//check if the student file is empty
//empty, return 1
bool isStdEmpty()
{
    return (tail == NULL && head == NULL);
}

//check if the student information file is empty
//empty, return 1
bool isEmpty()
{
    return (backPtr == NULL && frontPtr==NULL);
}

//insert the student from the data file into the queue
void insertStdInfo(string n, string stdId, string pw)
{
    Student *userDetail = new Student(n, stdId, pw);

    if(isStdEmpty())
    {
        userDetail->next = NULL;
        head = tail = userDetail;
    }

    else
    {
        userDetail->next = NULL;
        tail->next = userDetail;
        tail = userDetail;
    }
}
```

```
//the function is virtual to the checkValid function in class Person
//check the validity of the supervisor whether can enter to the system or not
string checkStdValid(string stdId, string stdPw)
{
    Student *check = head;
    string validity = "Invalid";

    while(check != NULL)
    {
        //if student enter the ID is found in the data file
        if(stdId == check->getId())
        {
            //if student enter the password is found in the data file
            if(stdPw == check->getPassword() )
            {
                validity = "Valid";
                return validity;
            }
        }

        check = check->next;
    }
}
```

```
//insert the student information in the data file to the queue
void insertQueue(string userName, string userId, string course, int stdSection, string projectName,
string supervisorName)
{
    Student *newPtr = new Student(userName, userId, course,stdSection, projectName,
    supervisorName);

    //if the queue is still empty
    //store the first information in the queue
    if(isEmpty())
    {
        newPtr->next = NULL;
        frontPtr = backPtr = newPtr;
    }

    else
    {
        newPtr->next = NULL;
        backPtr->next = newPtr;
        backPtr = newPtr;
    }

}
```

```
//supervisor insert marks for the project of the students and make comments
void insertMarks()
{
    Student *newPtr = frontPtr;
    Student *temp = frontPtr;
    int num = 0, counter = 1;
    double stdMarks;
    string comments;

    while(temp != NULL)
    {
        num++;
        temp = temp->next;
    }


    cout << "\t\t\tEnter marks for student Final Year Project\n\n";
    while(num != 0)
    {

        cout << counter << ". Name: " << newPtr->getName() << endl;
        cout << "   ID: " << newPtr->getId() << endl;
        cout << "   Project Name : " << newPtr->getProject() << endl;
        cout << "   Supervisor Name : " << newPtr->sg.getName() << endl;
        cout << "   Marks: ";
        cin >> stdMarks;

        //supervisor only can enter the marks between 0 and 100
        while(stdMarks<0 || stdMarks> 100)
        {
            cout << "\n   Error! Please enter the marks again (0-100)\n";
            cout << "   Marks: ";
            cin >> stdMarks;
        }

        cin.ignore();
        cout << "   Comment: ";
        getline(cin, comments);

        newPtr->marks = stdMarks;
        newPtr->comment = comments;
        newPtr->setGrade(stdMarks);

        if(counter == 1)
        {
            frontPtr->marks = backPtr->marks = newPtr->marks;
            frontPtr->comment = backPtr->comment = newPtr->comment;
        }
        else
        {
            backPtr->marks = newPtr->marks;
            backPtr->comment = newPtr->comment;
        }

        newPtr = newPtr->next;

        cout << endl;
        counter++;
        num--;
    }
}
```

```
//display the overall student information
void display()
{
    Student *temp = frontPtr;

    cout << "\n\t\t\t\t<<<<<<<<<<< Student Information >>>>>>>>>>>\n\n";

    if(isEmpty() || temp == NULL)
    {
        cout << "\nQueue is Empty" << endl;
    }

    else
    {
        cout << left << setw(26) << "Name"
            << left << setw(14) << "ID"
            << left << setw(15) << "Course"
            << left << setw(11) << "Section"
            << left << setw(28) << "Project Name"
            << left << setw(28) << "Supervisor Name"
            << left << setw(7) << "Marks"
            << left << setw(7) << "Grade"
            << left << setw(18) << "Comment";
            cout << endl;

        while(temp != NULL)
        {
            cout << left << setw(25) << temp->getName()
                << left << setw(14) << temp->getId()
                << left << setw(17) << temp->getCourse()
                << left << setw(10) << temp->getSection()
                << left << setw(28) << temp->getProject()
                << left << setw(21) << temp->sp.getName()
                << left << setw(7) << temp->getMarks()
                << left << setw(7) << temp->getGrade()
                << left << setw(28) << temp->getComment() << endl;

                temp = temp->next;
        }
    }

    cout << endl;

}

//student and supervisor able to search the student information
```

```
//based on the id enter by users
void searchId()
{
    cin.ignore();
    int repeat, num =0;
    string identity;
    int count = 0;
    Student *search_key = frontPtr;
    Student *temp = frontPtr;

    while(temp != NULL)
    {
        num++;
        temp = temp->next;
    }


    system("cls");
    cout << "\n<<<<<<<<<<< Search Student Information Based on ID >>>>>>>>>>>\n";
    do{

        search_key = frontPtr;
        count = 0;

        cout << "\nEnter student id: ";
        getline(cin, identity);

        while(search_key != NULL)
        {
            count++;
            if(identity == search_key->getId())
            {
                cout << "\nStudent Name: " << search_key->getName() << endl;
                cout << "ID : " << search_key->getId() << endl;
                cout << "Course: " << search_key->getCourse() << endl;
                cout << "Section : " << search_key->getSection() << endl;
                cout << "Project Name: " << search_key->getProject() << endl;
                cout << "Supervisor Name: " << search_key->sp.getName() << endl;
                cout << "Marks: " << search_key->getMarks() << endl;
                cout << "Grade: " << search_key->getGrade() << endl;
                cout << "Comment: " << search_key->getComment() << endl;

                break;
            }
```

```
            }

            //when the searching come to end of the queue, display the prompt to user
            if(count == num )
            {
                cout << "\nStudent does not exist in the system" << endl;
                break;
            }

            search_key = search_key->next;

        }

        cout << "\nSearch again?";
        cout << "\n1. Yes ";
        cout << "\n2. No ";
        cout << "\nEnter your choice-> ";
        cin >> repeat;

        while(repeat <1 || repeat >2)
        {
            cout << "\nWorng choice ! Enter again." << endl;
            cout << "Enter your choice-> ";
            cin >> repeat;
        }

        cin.ignore();

    }while(repeat == 1);

    system("cls");
    cout << "\n\t\t\tLoading";
    for(int i=0; i<3; i++)
    {
        sleep(1);
        cout << ". ";
    }

    cout << endl;
}
```

```
        ~queue()
        {
            Student *currNode1 = frontPtr = backPtr, *nextNode = NULL;
            while(currNode1 != NULL)
            {
                nextNode = currNode1->next;
                //destroy the current node
                delete currNode1;
                currNode1 = nextNode;
            }

            frontPtr = backPtr = NULL;

            Student *currNode2 = head = tail, *nextNode2 = NULL;
            while(currNode2 != NULL)
            {
                nextNode2 = currNode2->next;
                //destroy the current node
                delete currNode2;
                currNode2 = nextNode2;
            }

            head = tail = NULL;
        }

};
```

```
//menu display to users for choosing
void menu(int user, queue &m)
{
    int choice;

    //if the user is student
    if(user == 1)
    {
        do{
        system("cls");
        cout << "\t\t\t<<<<<<<<<<< Menu >>>>>>>>>>>\n";
        cout << "\t\t\t1) Search Student Result based on ID " << endl;
        cout << "\t\t\t2) Display all the elements of queue" << endl;
        cout << "\t\t\t3) Back to login"<< endl << endl;
        cout << "\t\t\tEnter your choice-> ";
        cin >> choice;


        switch(choice)
        {

            case 1: system("cls");
                    m.searchId();   //serach results based on id
                    break;

            case 2: system("cls");
                    m.display();    //display overall student information
                    system("pause");
                    break;

            //back to Log in interface
            case 3: system("cls");
                    cin.ignore();
                    cout << "\n\t\t\tLoading";
                    for(int i=0; i<3; i++)
                    {
                        sleep(1);
                        cout << ". ";
                    }
                    break;

            default: cout << "\n\t\t\tWrong Choice!Enter again!\n\t\t\t";
                    system("pause");
        }

        }while(choice != 3);

    }
```

```
    //if supervisor use the system
    if(user == 2)
    {
        do{
        system("cls");
        cout << "\t\t\t<<<<<<<<<<< Menu >>>>>>>>>>>\n";
        cout << "\t\t\t1) Insert student marks to the queue" <<endl;
        cout << "\t\t\t2) Delete student marks from the queue" <<endl;
        cout << "\t\t\t3) Search Student Result based on ID " << endl;
        cout << "\t\t\t4) Delete Student information in the queue " << endl;
        cout << "\t\t\t5) Display all the elements of queue" << endl;
        cout << "\t\t\t6) Back to login" << endl <<endl;
        cout << "\t\t\tEnter your choice-> ";
        cin >> choice;

        switch(choice)
        {
            case 1: system("cls");
                    m.insertMarks();    //insert marks
                    system("pause");
                    break;

            case 2: system("cls");
                    m.deleteMarks();    //delete marks from the beginning
                    system("pause");
                    break;

            case 3: system("cls");
                    m.searchId();       //search student's result based on id
                    break;

            case 4: system("cls");
                    m.deleteQueue();    //delete the validity of student to use the system
                    system("pause");
                    break;

            case 5: system("cls");
                    m.display();        //display overall student information
                    system("pause");
                    break;

            //back to log in interface
            case 6: system("cls");
                    cin.ignore();
                    cout << "\n\t\t\tLoading";
                    for(int i=0; i<3; i++)
                    {
                        sleep(1);
                        cout << ". ";
                    }
                    break;

            default: cout << "\n\t\t\tWrong Choice!Enter again!\n\t\t\t";
                    system("pause");
        }

        }while(choice != 6);
    }
}
```

```
int main()
{
    //declare local variable
    int choice, stdSection, chance;
    //choice - the choice enter by users to perform different function
    //stdSection - to get the section where student enroll in in the text file
    //chance - the chance for user to enter the id and password

    string userName, userPassword, course, userId, projectName, supervisorName, validity, validId, validPw;
    //userName - to get the name of the user in text file
    //userPassword - to compare the password of the user
    //course - to get the course of the student in text file
    //userID - the id of the user
    //projectName - to get the project title done by student in text file
    //supervisorName - to get the supervisor name of the student in text file
    //validity - to check the validity of users for using the system based on the id and password enter
    //validId - to check the id enter by user is same as in text file
    //validPw - to check the password enter by user is same as in text file

    Supervisor super;
    //super - to access the member function in the class Supervisor

    queue q;
    //q - to access the member function in the class queue

    //call the function to create queue
    q.createQueue();
```

```
//open student file
infile.open("nIDpass.txt");

//if file not exist
if(! infile)
{
    cout << "File not exits! Terminating";
    exit(1);
}

//if file exist
//read file from student file
infile.ignore(MAXSTREAM, '\n');
while(! infile.eof())
{
    getline(infile, userName, '\t');
    getline(infile, userId, '\t');
    getline(infile, userPassword, '\n');
    q.insertStdInfo(userName, userId, userPassword);
}
//close student file
infile.close();
```

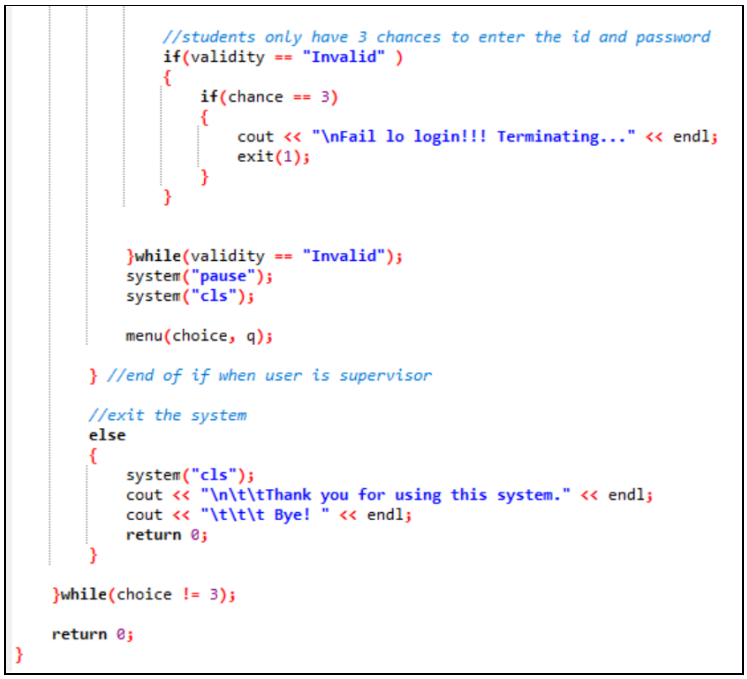
```
//open Supervisor File
inputFile.open("superData.txt");

//if file not exist
if(! inputFile)
{
    cout << "File not exits! Terminating";
    exit(1);
}

//if file exist
//read file from supervisor file
inputFile.ignore(MAXSTREAM, '\n');
inputFile >> userId;
inputFile.ignore(MAXSTREAM, '\t');
getline(inputFile, userName, '\t');
getline(inputFile, userPassword, '\n');
super.getData(userName, userId, userPassword);

//close supervisor file
inputFile.close();
```

```
//open student information file
inp.open("studentInfo.txt");

//if file not exist
if(! inp)
{
    cout << "File not exits! Terminating";
    exit(1);
}

//if file exist
//read file from student information file
inp.ignore(MAXSTREAM, '\n');
while(! inp.eof())
{
    getline(inp, userName, '\t');
    getline(inp, userId, '\t');
    getline(inp, course, '\t');
    inp >> stdSection;
    inp.ignore(MAXSTREAM, '\t');
    getline(inp, projectName, '\t');
    getline(inp, supervisorName, '\n');
    q.insertQueue(userName, userId, course, stdSection, projectName, supervisorName);
}

//close student information file
inp.close();
```

```
do{
    system("cls");

    //menu for choosing role as student or supervisor
    cout << "\t\tWelcome to the Final Year Project System" << endl << endl;
    cout << "\t\t\tSign in:\n";
    cout << "\t\t\t1. Student\n";
    cout << "\t\t\t2. Supervisor\n";
    cout << "\t\t\t3. Exit\n\n";
    cout << "\t\t\tEnter your choice -> ";
    cin >> choice;

    //if choice below 1 and above 3 will ask for choosing again
    while (choice < 1 || choice >3)
    {
        cout << "\n\t\t\tError! Please choose again!\n";
        cout << "\t\t\tEnter your choice -> ";
        cin >> choice;
    }

    system("cls");
    cin.ignore();
    cout << "\n\t\t\tLoading";
    for(int i=0; i<3; i++)
    {
        sleep(1);
        cout << ". ";
    }
```

```
//Student check the marks and grade of their project
if(choice ==  1)
{
    chance = 0;
    do{
        system("cls");
        //student have 3 chances to enter
        cout << "You have 3 chances to enter the id and password" << endl;

        //check student identification
        cout << "Enter ID: ";
        cin >>  validId;
        cin.ignore();
        cout << "Enter password: ";
        getline(cin, validPw);

        validity = q.checkStdValid(validId, validPw);
        chance++;

        while(validity == "Invalid")
        {
            cout << "\nError, System Record Not Found! Please enter again!" << endl;
            system("pause");
            system("cls");
            break;
        }

        //students only have 3 chances to enter the id and password
        if(validity == "Invalid" )
        {
            if(chance == 3)
            {
                cout << "\nFail lo login!!! Terminating..." << endl;
                exit(1);
            }
        }


    }while(validity == "Invalid");
    system("pause");
    system("cls");

    menu(choice, q);

}//end of if when user is a student
```

```
//supervisor evaluate and view the results
else if(choice == 2)
{
    chance = 0;

    do{

        system("cls");
        //supervisor have 3 chances to enter
        cout << "You have 3 chances to enter the id and password" << endl;

        //check supervisor identification
        cout << "Enter ID: ";
        cin >>  validId;
        cin.ignore();
        cout << "Enter password: ";
        getline(cin, validPw);

        validity = super.checkValid(validId, validPw);
        chance++;

        while(validity == "Invalid")
        {
            cout << "\nError, System Record Not Found! Please enter again!" << endl;
            system("pause");
            system("cls");
            break;
        }

        //students only have 3 chances to enter the id and password
        if(validity == "Invalid" )
        {
            if(chance == 3)
            {
                cout << "\nFail lo login!!! Terminating..." << endl;
                exit(1);
            }
        }


    }while(validity == "Invalid");
    system("pause");
    system("cls");

    menu(choice, q);

} //end of if when user is supervisor
```



```
            //students only have 3 chances to enter the id and password
            if(validity == "Invalid" )
            {
                if(chance == 3)
                {
                    cout << "\nFail lo login!!! Terminating..." << endl;
                    exit(1);
                }
            }


        }while(validity == "Invalid");
        system("pause");
        system("cls");

        menu(choice, q);

    } //end of if when user is supervisor

    //exit the system
    else
    {
        system("cls");
        cout << "\n\t\tThank you for using this system." << endl;
        cout << "\t\t\t Bye! " << endl;
        return 0;
    }

}while(choice != 3);

return 0;
}
```