STUDENT INFORMATION SYSTEM IN JAVA CONSOLE

A Project Documentation Submitted to the
College of Information Technology
Cagayan de Oro College
PHINMA Education

In Partial Fulfillment of the Requirements
for Computer Programming I

Submitted by:

Amion, Angela B.
Barillo, Julam A.
Ibona, Kiff Wenn Grey N.
Lavador, Lyca R.
Malong, Art Nathaniel B.
Noble, Ismael C.
Villarampa, Charls Jan Louise B.
Virrey, Jefferson Ray

March 2025

Introduction

In educational institutions, efficient management of student information is essential for maintaining academic records, tracking performance, and ensuring smooth administrative processes. Traditional manual systems or overly complex software can lead to inefficiencies and data inaccuracies. To address these challenges, this project presents a simple, console-based Student Information System developed using Java.

Many schools and small institutions still rely on manual methods or complex software to manage student records. Manual systems are prone to errors, time-consuming, and difficult to maintain. On the other hand, advanced software can be overwhelming and costly for small-scale use. This creates a need for a simple, efficient, and easy-to-use system that can handle basic student record management without unnecessary complexity. According to Brown and Davis (2020), lightweight digital tools for managing student records can significantly reduce administrative burdens while improving data reliability and access speed. This project is designed to provide an accessible and functional solution by implementing a menu-driven Java console application that handles basic student record operations.

This project is a console-based application that allows users to add new student records, view student details, update existing records, and exit the system when operations are complete. The system stores data in memory or simple text files, avoiding the complexity of database integration, which makes it ideal for small-scale use or educational purposes. While the application does not feature a Graphical User Interface (GUI) or advanced error handling, it serves as an effective prototype for understanding the core functionality of a Student Information System. It demonstrates how core programming concepts can be used to solve real-world problems in education through a streamlined, console-based approach.

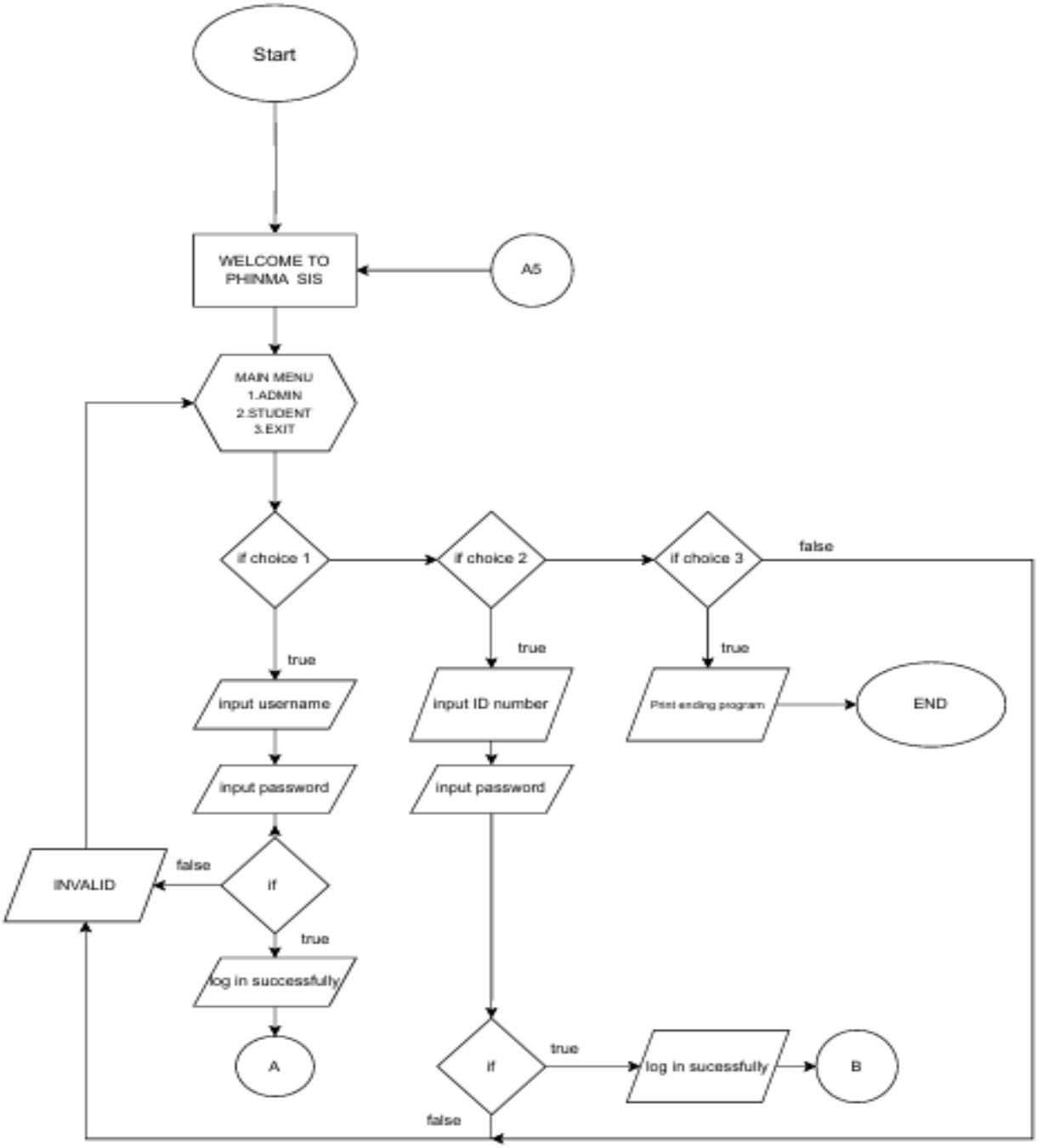
The primary objectives of the program are to build a simple and effective Student Information System using Java. It will support basic functions like adding, viewing, and updating,

which are known as CRU operations. The program will also use Object-Oriented Programming (OOP) principles such as encapsulation (hiding internal details) and abstraction (focusing on essential features), which help organize the code better and make it easier to understand.

The Student Information System includes several key features designed to enhance the system.. It allows for structured input of student details, such as ID, name, and course, ensuring that all information is stored in an organized and consistent manner. The system presents student records in a clear and organized display, making it easy for users to view and interpret data. A built-in search functionality enables users to quickly locate specific student information based on identifiers like ID or name. Additionally, the application uses a simple, menu-driven navigation system that guides users through various operations, making the program easy to use even for those with minimal technical experience.

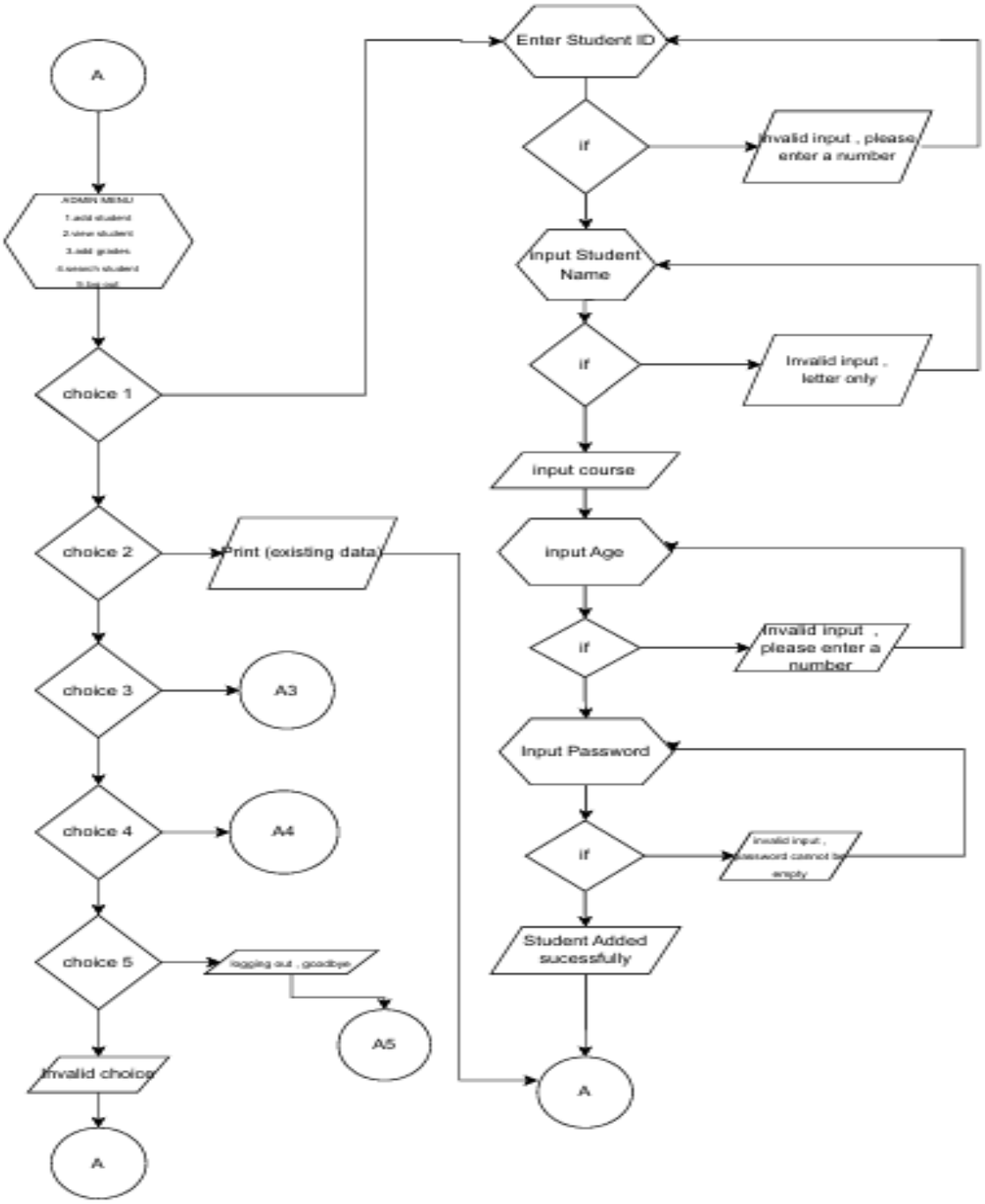
The technologies used in this project include Java for console-based programming, Java I/O for handling user input and file operations, Object-Oriented Programming (OOP) for structuring the system, and the Collections Framework for managing student records efficiently. The system operates through a menu-driven interface where the user can choose from different options to manage student records. The user interacts with the program through text-based commands, and the system processes and displays the corresponding output based on the selected option. This structured approach ensures ease of use and reliability, making it a practical solution for student record management.

Flow Chart


Start
WELCOME TO PHINMA SIS
A5
MAIN MENU
1.ADMIN
2.STUDENT
3.EXIT
if choice 1
if choice 2
if choice 3
false
true
true
true
input username
input ID number
Print ending program
END
input password
input password
INVALID
false
if
true
log in successfully
A
if
true
log in sucessfully
B
false

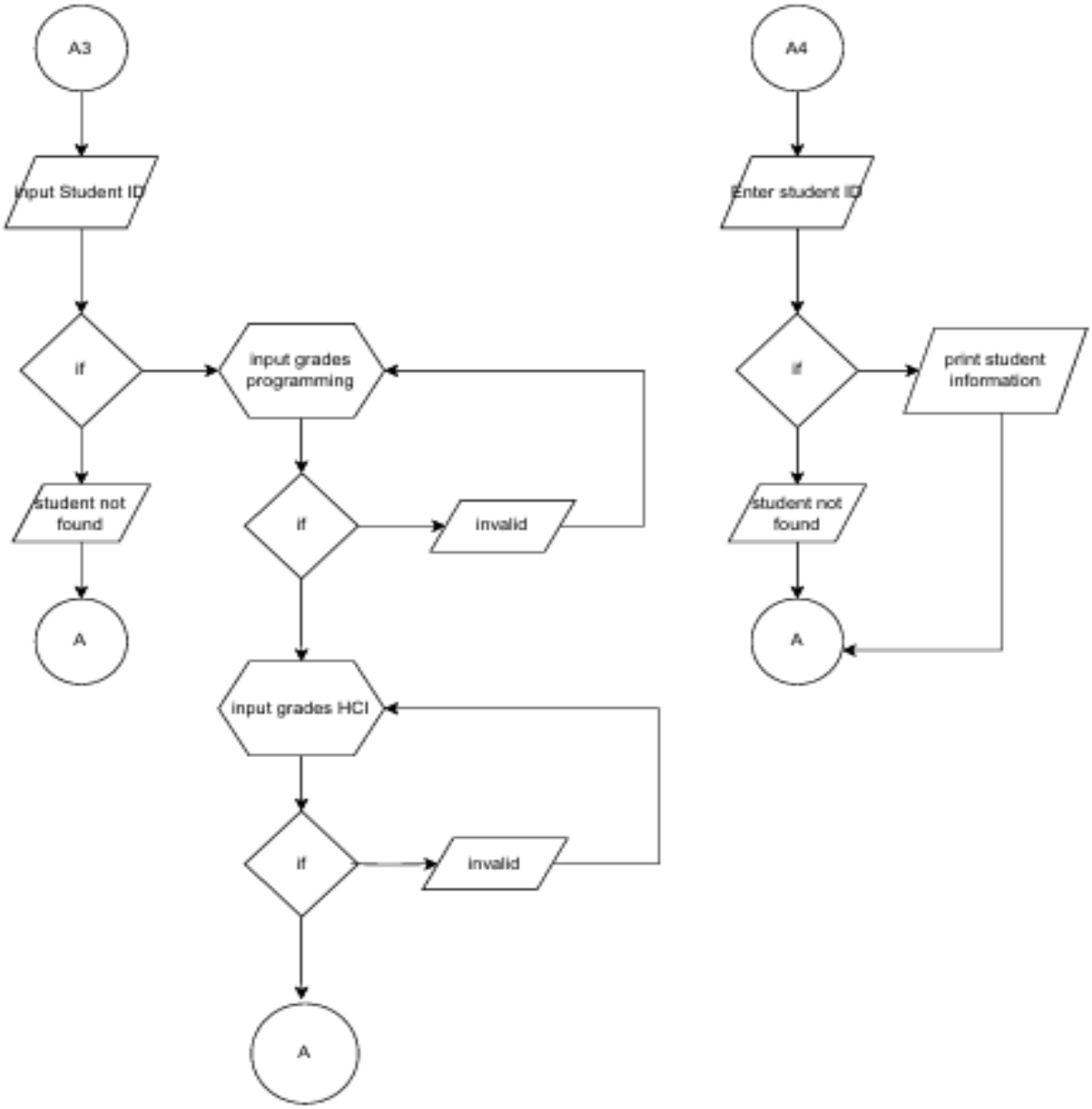
CONNECTOR A

ADMIN SECTION - MAIN



Connector A - 3 - 4

admin section choice 3 to 4



Connector B - STUDENT MAIN

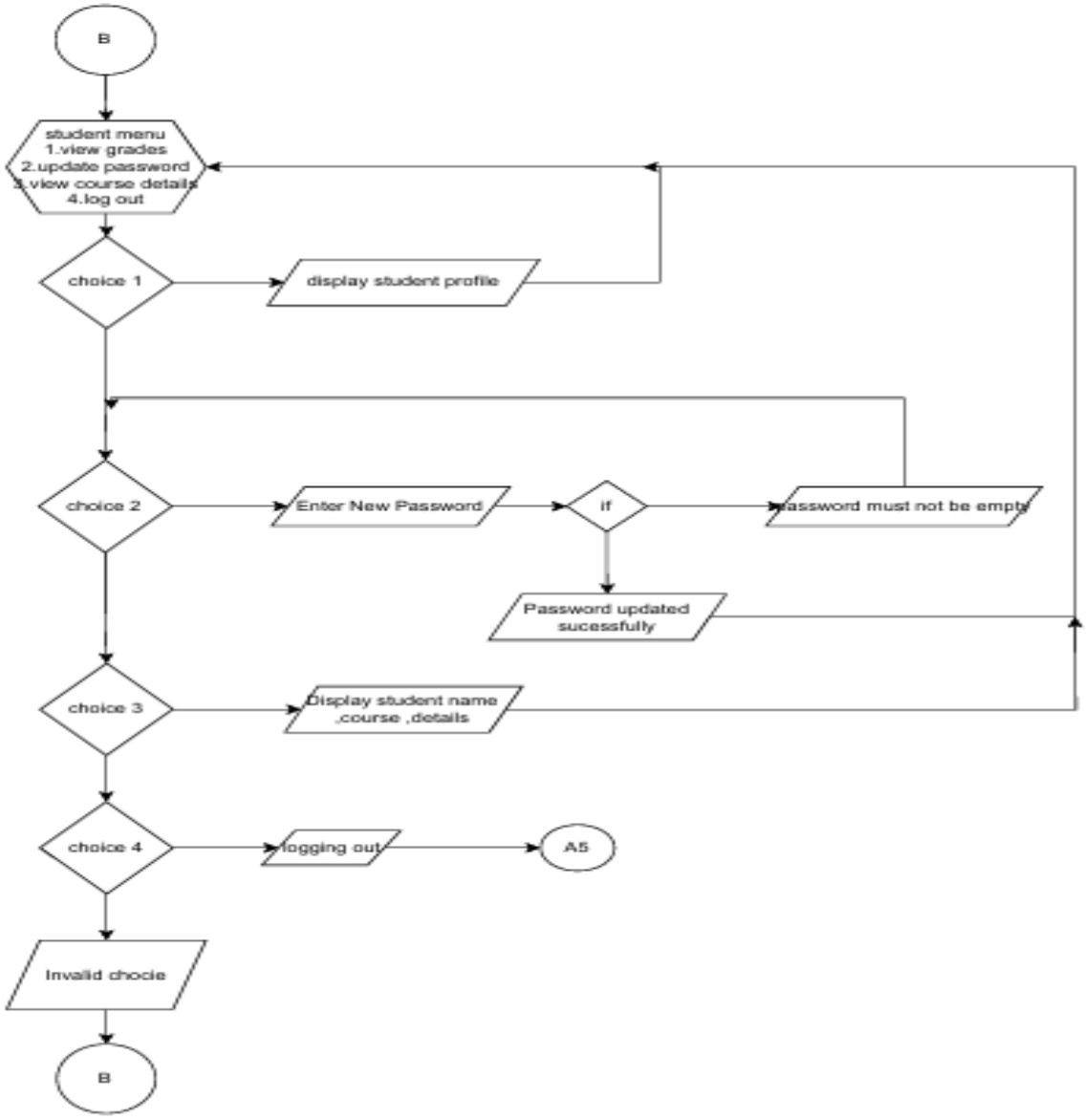
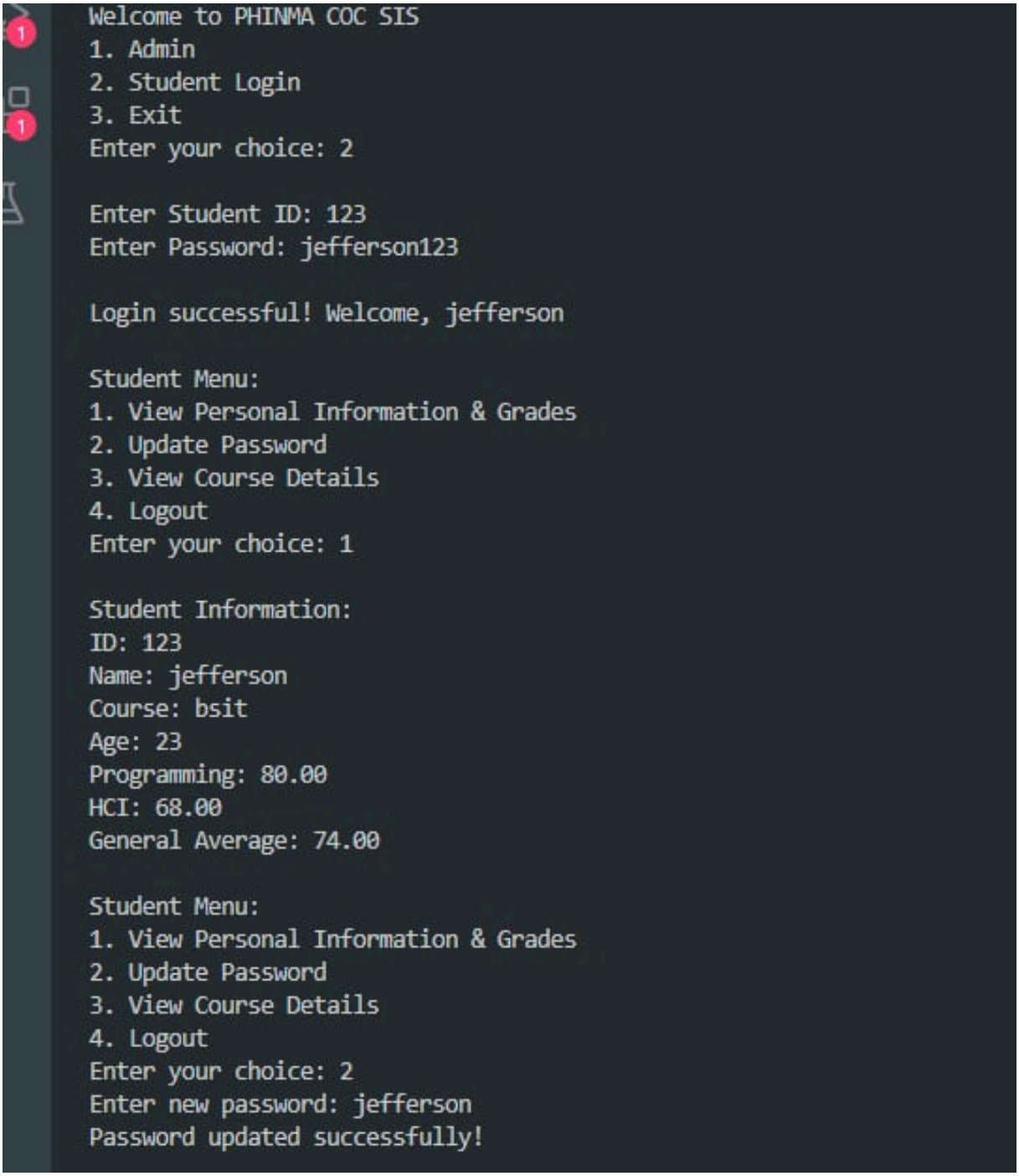


Figure 1.0 System Flow Chart

Output Screenshot and Program Code



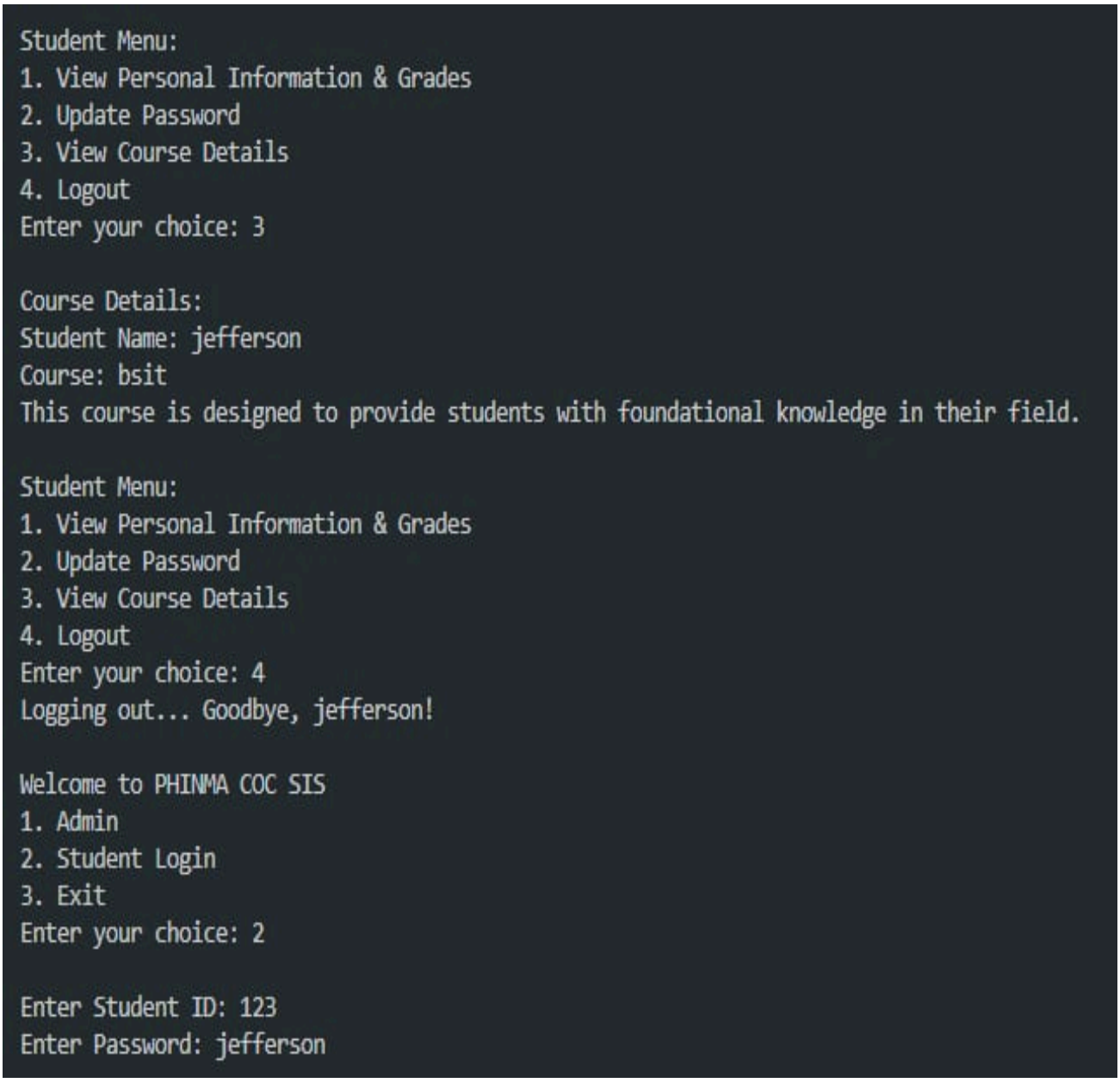


Figure 2.0 System Output Screenshot

Program Code

```
package sis;

import java.util.Scanner;

public class SIS {
    private static final int MAX_STUDENTS = 100;
    private static int[][] studentData = new int[MAX_STUDENTS][2]; // [ID, Age]
    private static String[] studentNames = new String[MAX_STUDENTS];
    private static String[] studentPasswords = new String[MAX_STUDENTS];
    private static String[] studentCourses = new String[MAX_STUDENTS];
    private static double[][] studentGrades = new double[MAX_STUDENTS][3]; //
[Programming, HCI, Average]
    private static int studentCount = 0;
    private static boolean isAdminLoggedIn = false;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\nWelcome to PHINMA COC SIS");
            System.out.println("1. Admin");
            System.out.println("2. Student Login");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            int choice = getIntInput(scanner);

            switch (choice) {
                case 1:
                    if (adminLogin(scanner)) {
                        adminMenu(scanner);
                    }
                    break;
                case 2:
                    studentLogin(scanner);
                    break;
                case 3:
                    System.out.println("Exiting the system. Goodbye!");
                    System.exit(0);
                default:
```

```
                System.out.println("Invalid choice! Please try again.");
            }
        }
    }

    private static boolean adminLogin(Scanner scanner) {
        System.out.print("\nEnter admin username: ");
        String username = scanner.nextLine();
        System.out.print("Enter admin password: ");
        String password = scanner.nextLine();

        if (username.equals("admin") && password.equals("admin123")) {
            System.out.println("Admin login successful! Welcome, Admin.");
            isAdminLoggedIn = true;
            return true;
        }
        System.out.println("Invalid admin credentials. Please try again.");
        return false;
    }

    private static void adminMenu(Scanner scanner) {
        while (isAdminLoggedIn) {
            System.out.println("\nAdmin Menu");
            System.out.println("1. Add Student");
            System.out.println("2. View Students");
            System.out.println("3. Add Grades");
            System.out.println("4. Search Student by ID");
            System.out.println("5. Logout");
            System.out.print("Enter your choice: ");
            int choice = getIntInput(scanner);

            switch (choice) {
                case 1:
                    addStudent(scanner);
                    break;
                case 2:
                    viewStudents();
                    break;
                case 3:
                    addGrades(scanner);
                    break;
                case 4:
                    searchStudentByID(scanner);
                    break;
```

```
            case 5:
                System.out.println("Logging out... Goodbye, Admin!");
                isAdminLoggedIn = false;
                break;
            default:
                System.out.println("Invalid choice! Try again.");
        }
    }
}

private static void addStudent(Scanner scanner) {
if (studentCount >= MAX_STUDENTS) {
    System.out.println("Student limit reached! Cannot add more students.");
    return;
}

System.out.print("Enter Student ID: ");
int id = getIntInput(scanner);

if (isDuplicateID(id)) {
    System.out.println("Student ID already exists. Try again.");
    return;
}

String name;
while (true) {
    System.out.print("Enter Student Name: ");
    name = scanner.nextLine();

    // Check if the name contains only letters and spaces
    if (name.matches("[a-zA-Z ]+")) {
        break;
    } else {
        System.out.println("Invalid name! Please enter a valid name (letters only).");
    }
}

System.out.print("Enter Student Course: ");
String course = scanner.nextLine();

System.out.print("Enter Student Age: ");
int age = getIntInput(scanner);

String password;
```

```
    while (true) {
        System.out.print("Enter Student Password: ");
        password = scanner.nextLine().trim(); // Trim to remove spaces

        // Check if password is empty
        if (!password.isEmpty()) {
            break;
        } else {
            System.out.println("Invalid password! Password cannot be empty. Try again.");
        }
    }


    studentData[studentCount][0] = id;
    studentData[studentCount][1] = age;
    studentNames[studentCount] = name;
    studentCourses[studentCount] = course;
    studentPasswords[studentCount] = password;

    studentGrades[studentCount][0] = -1;
    studentGrades[studentCount][1] = -1;
    studentGrades[studentCount][2] = -1;

    studentCount++;
    System.out.println("Student added successfully!");
}

    private static void viewStudents() {
        if (studentCount == 0) {
            System.out.println("No students available.");
            return;
        }

        System.out.println("\nList of Students:");
        for (int i = 0; i < studentCount; i++) {
            System.out.println("ID: " + studentData[i][0] +
                    ", Name: " + studentNames[i] +
                    ", Course: " + studentCourses[i] +
                    ", Age: " + studentData[i][1] +
                    ", Programming: " + formatGrade(studentGrades[i][0]) +
                    ", HCI: " + formatGrade(studentGrades[i][1]) +
                    ", Average: " + formatGrade(studentGrades[i][2]));
        }
    }
```

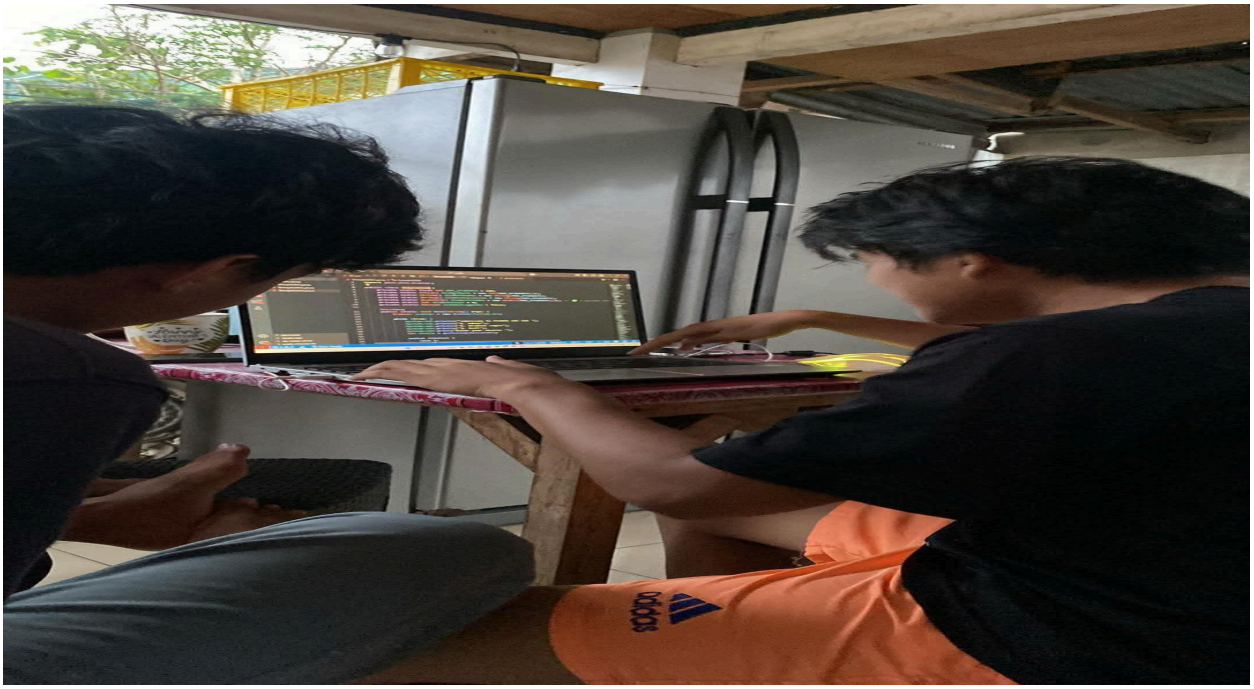
```
    private static void addGrades(Scanner scanner) {
        System.out.print("\nEnter Student ID: ");
        int id = getIntInput(scanner);
        int index = getStudentIndex(id);

        if (index == -1) {
            System.out.println("Student not found.");
            return;
        }

        System.out.print("Enter Programming Grade: ");
        studentGrades[index][0] = getDoubleInput(scanner);

        System.out.print("Enter HCI Grade: ");
        studentGrades[index][1] = getDoubleInput(scanner);

        studentGrades[index][2] = (studentGrades[index][0] + studentGrades[index][1]) / 2; //
Calculate Average

        System.out.println("Grades added successfully!");
    }

    private static void searchStudentByID(Scanner scanner) {
        System.out.print("\nEnter Student ID to search: ");
        int id = getIntInput(scanner);
        int index = getStudentIndex(id);

        if (index == -1) {
            System.out.println("Student not found.");
        } else {
            System.out.println("\nStudent Information:");
            System.out.println("ID: " + studentData[index][0]);
            System.out.println("Name: " + studentNames[index]);
            System.out.println("Course: " + studentCourses[index]);
            System.out.println("Age: " + studentData[index][1]);
            System.out.println("Programming: " + formatGrade(studentGrades[index][0]));
            System.out.println("HCI: " + formatGrade(studentGrades[index][1]));
            System.out.println("General Average: " + formatGrade(studentGrades[index][2]));
        }
    }

    private static void studentLogin(Scanner scanner) {
    System.out.print("\nEnter Student ID: ");
```

```
    int id = getIntInput(scanner);
    int index = getStudentIndex(id);

    if (index == -1) {
        System.out.println("Student not found.");
        return;
    }

    System.out.print("Enter Password: ");
    String password = scanner.nextLine();

    if (studentPasswords[index].equals(password)) {
        System.out.println("\nLogin successful! Welcome, " + studentNames[index]);
        studentMenu(scanner, index); // Open student menu
    } else {
        System.out.println("Incorrect password. Try again.");
    }
}

private static void studentMenu(Scanner scanner, int index) {
    while (true) {
        System.out.println("\nStudent Menu:");
        System.out.println("1. View Personal Information & Grades");
        System.out.println("2. Update Password");
        System.out.println("3. View Course Details");
        System.out.println("4. Logout");
        System.out.print("Enter your choice: ");
        int choice = getIntInput(scanner);

        switch (choice) {
            case 1:
                displayStudentInfo(index);
                break;
            case 2:
                updateStudentPassword(scanner, index);
                break;
            case 3:
                viewCourseDetails(index);
                break;
            case 4:
                System.out.println("Logging out... Goodbye, " + studentNames[index] + "!");
                return; // Exit student menu
            default:
                System.out.println("Invalid choice! Please try again.");
```

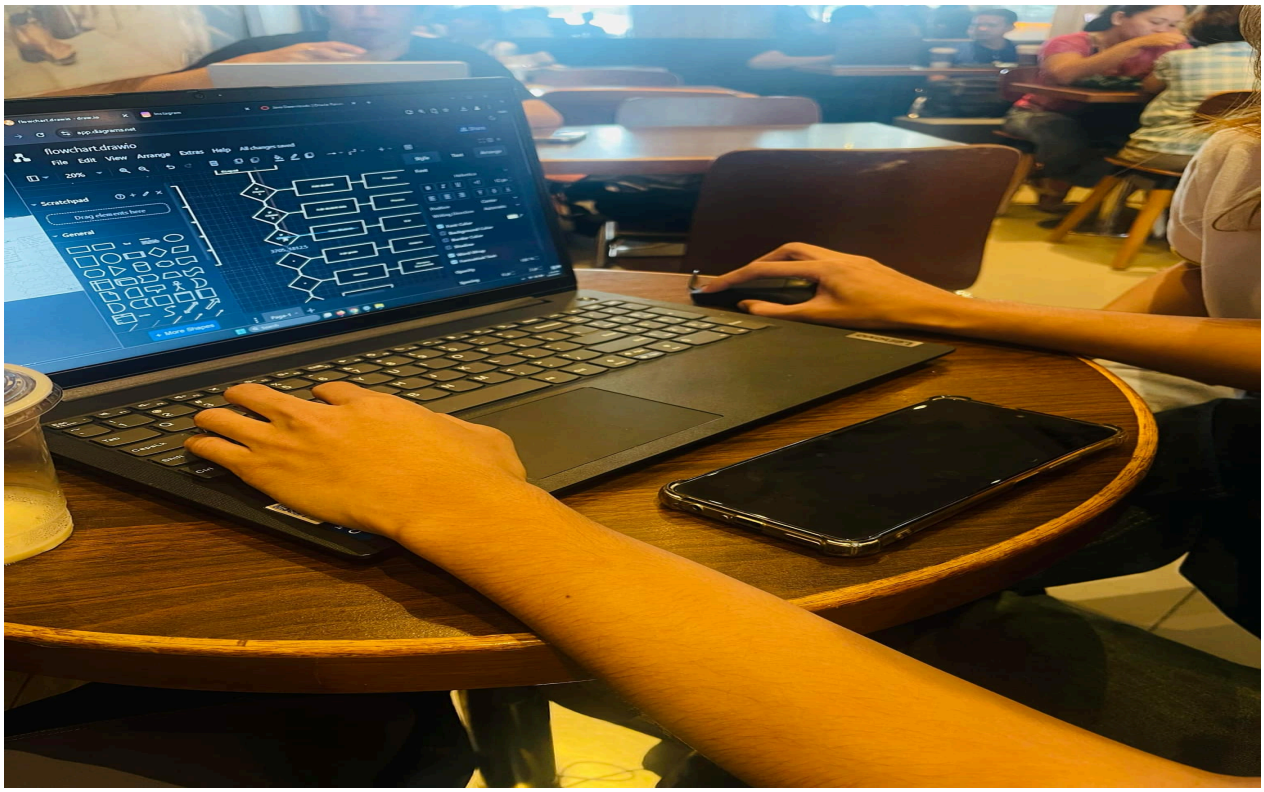
```
        }
    }
}
private static void displayStudentInfo(int index) {
    System.out.println("\nStudent Information:");
    System.out.println("ID: " + studentData[index][0]);
    System.out.println("Name: " + studentNames[index]);
    System.out.println("Course: " + studentCourses[index]);
    System.out.println("Age: " + studentData[index][1]);
    System.out.println("Programming: " + formatGrade(studentGrades[index][0]));
    System.out.println("HCI: " + formatGrade(studentGrades[index][1]));
    System.out.println("General Average: " + formatGrade(studentGrades[index][2]));
}
private static void updateStudentPassword(Scanner scanner, int index) {
    System.out.print("Enter new password: ");
    String newPassword = scanner.nextLine();

    while (newPassword.trim().isEmpty()) {
        System.out.println("Password cannot be empty! Try again.");
        System.out.print("Enter new password: ");
        newPassword = scanner.nextLine();
    }

    studentPasswords[index] = newPassword;
    System.out.println("Password updated successfully!");
}
private static void viewCourseDetails(int index) {
    System.out.println("\nCourse Details:");
    System.out.println("Student Name: " + studentNames[index]);
    System.out.println("Course: " + studentCourses[index]);
    System.out.println("This course is designed to provide students with foundational knowledge
in their field.");
}

    private static boolean isDuplicateID(int id) {
        return getStudentIndex(id) != -1;
    }

    private static int getStudentIndex(int id) {
        for (int i = 0; i < studentCount; i++) {
            if (studentData[i][0] == id) {
                return i;
            }
        }
```

```
        return -1;
    }

    private static int getIntInput(Scanner scanner) {
        while (!scanner.hasNextInt()) {
            System.out.print("Invalid input! Please enter a number: ");
            scanner.next();
        }
        int value = scanner.nextInt();
        scanner.nextLine();
        return value;
    }

    private static double getDoubleInput(Scanner scanner) {
        while (!scanner.hasNextDouble()) {
            System.out.print("Invalid input! Please enter a valid number: ");
            scanner.next();
        }
        return scanner.nextDouble();
    }

    private static String formatGrade(double grade) {
        return (grade == -1) ? "N/A" : String.format("%.2f", grade);
    }
}
```

Documentation




BE PART OF OUR ONLINE COMMUNITY
Oracle


flowchart.drawio
File Edit View Arrange Extras Help
Scratchpad
General