FINAL SUBISSION REPORT ON PROJECT

STUDENT MANAGEMENT SYSTEM

SUBMITTED BY: -

NAME	REGISTRION NO:	ROLL NO:
ASHISH BHARTI	12109333	28
IMRAN ANSARI	12109519	31
KAYYUM KHAN	12112789	49

ACKNOWLEDGEMENT

We would like to express our heartfelt appreciation and gratitude to all those who have contributed to the successful completion of our Java project. Without their support, dedication, and expertise, this project would not have been possible. We would like to acknowledge the following individuals and organizations for their invaluable contributions:

Our project supervisor, Mrs. Amarinder Kaur Ma'am for their guidance, mentorship, and continuous support throughout the project. Their insights, feedback, and encouragement were instrumental in shaping the project and ensuring its success.

The online Java community, for their vast resources, tutorials, and forums that provided us with valuable insights and solutions during the development process. Their support was invaluable in resolving technical issues and improving our project.

We would also like to extend our gratitude to anyone else who has supportedus in any way, directly or indirectly, during this Java project. Thank you for your contribution and commitment to our project's success.

INTRODUCTION

A student management system is an application designed to help educational institutions manage student data. From tracking student attendance to storing grades, such systems can improve efficiency and reduce errors. A well-designed student management system can streamline administrative tasks and provide teachers and staff with real-time data on student performance. In this paper, we will explore the development of a student management system using the Java programming language.

In today's education system, technology is becoming increasingly important. One area where this is particularly evident is in student management systems. With the use of Java programming, these systems are becoming more efficient and effective than ever before. By automating tasks such as attendance tracking and grade calculation, teachers are free to focus on what they do best: educating their students.

In the world of academia, managing students' records and their academic performance is crucial. With the advancement of technology, student management systems have been developed to facilitate this process. One of the most popular programming languages used for developing student management systems is Java. Java's simplicity, readability and portability make it an excellent choice for such systems.

In this paper, we will explore the development of a student management system using Java. With the increasing demand for effective school management, the development of a reliable student management system has become a necessity. This system will help to manage student records, attendance, grades, and other important information in one centralized location. Through the use of Java's powerful tools, we will present a program that is capable of

meeting these needs efficiently and effectively.

In today's world, technology is advancing at an unprecedented rate. One of the areas where this is most prominent is in the field of education. Many universities and schools are adopting student management systems to help streamline their operations and improve student outcomes. The use of Java programming language has proven especially useful for creating powerful and efficient student management systems that can help institutions manage their resources effectively.

MODULES: -

1. Add New Student Module: -
   This Method helps the user to Add New Student Details in The database.

2. Edit Student Information Module: -
   This Method helps the user to edit The Information Of The Students.

3. Delete Module: -
   This Method helps the user to Delete The Details Of Students From Database.

4. Display All Student module: -
   This Method helps the user to display All Students Information .

5. Quit Module: -
   This Method helps the user to Quit From Program.

Features Of Student Management System

1. Student Data Maintenance

An SIS provides you with complete data insights. You can look at your dashboard and configure your reports on student grades, outcomes, performance, schedules, fees, and events. This can help you make critical decisions more effectively.

2. Student Details

A student information management system also has a comprehensive dashboard with a single view of student details. In this feature, you get a complete student portfolio and details about their enrollment, assessments, course registration, library, research, publications, etc.

3. Admission Management

The student data management system automates the entire admission process. It helps you get rid of the tedious process of manually handling the admission process. Right from enquires to applications and enrollment process, an SIS automates the entire process!

4. Student Record Maintenance

Maintaining discipline within the higher education campuses is hard. Adapting a student database management system is a great way to take care of the disciplinary issues and generate actions to compare, analyse, and report discipline across the departments.

It has the following attributes: –

S-Id:

Student's Identity number attribute is a distinct numeric field that will be given to every student registered in the institute. It helps in making every student unique throughout the system and helps the administrator.

Name:

Student name is the personal information of the student. It helps in making the system friendlier for the user and aids the admin in search or update the record.

Section:

Every student is divided into different segments that belong to the same course. This helps in making the study more efficient for every student.

Email-Id:

This will store the email id of the student required for sending the urgent update to a student from the institute.

C-Id:

As many courses can be opted by a student. So, the multivalued attribute required to store all the reference id of the courses for which the student has enrolled. Courses Id is, therefore, belonging to course entity.

Mobile:

The mobile number of the student is an attribute is used as a point of contact to the student.

Address:

This field is a composite attribute of the city and the pin code. As the address required the full location of the student.

ROLES AND RESPONSIBLITIES

ASHISH BHARTI	By using the concept of java, will be implementing the module 1 and module 3. Also,will be collecting information and writing the report.
KAYYUM KHAN	By using the concept of java, will be implementing the module 2 and module 4. Also,testing the functionality of this program and designing and writing the report.
IMRAN ANSARI	By using the concept of java, will be implementing the module 1 and module 5. Also,collecting information and creating content for report making.

GANTT CHART

PROJECT PHASES	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 5
PLANNING					
DESIGN					
CODING					
TESTING					
FINAL TOUCH					

Week1-

Planning and Research on the topic allocated.

Week2-

Data collection and designing the project.

Week3-

Writing the code for our project.

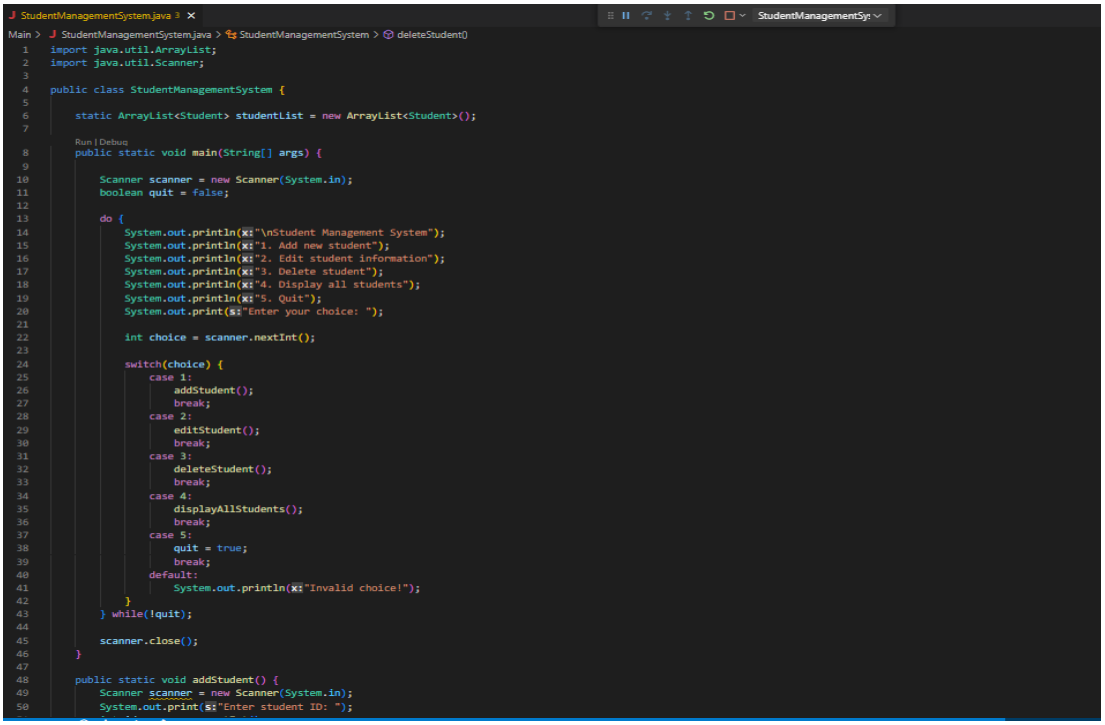
Week4-

Testing is done to check the functionality of code.

Week5-

Report writing and submission is done.

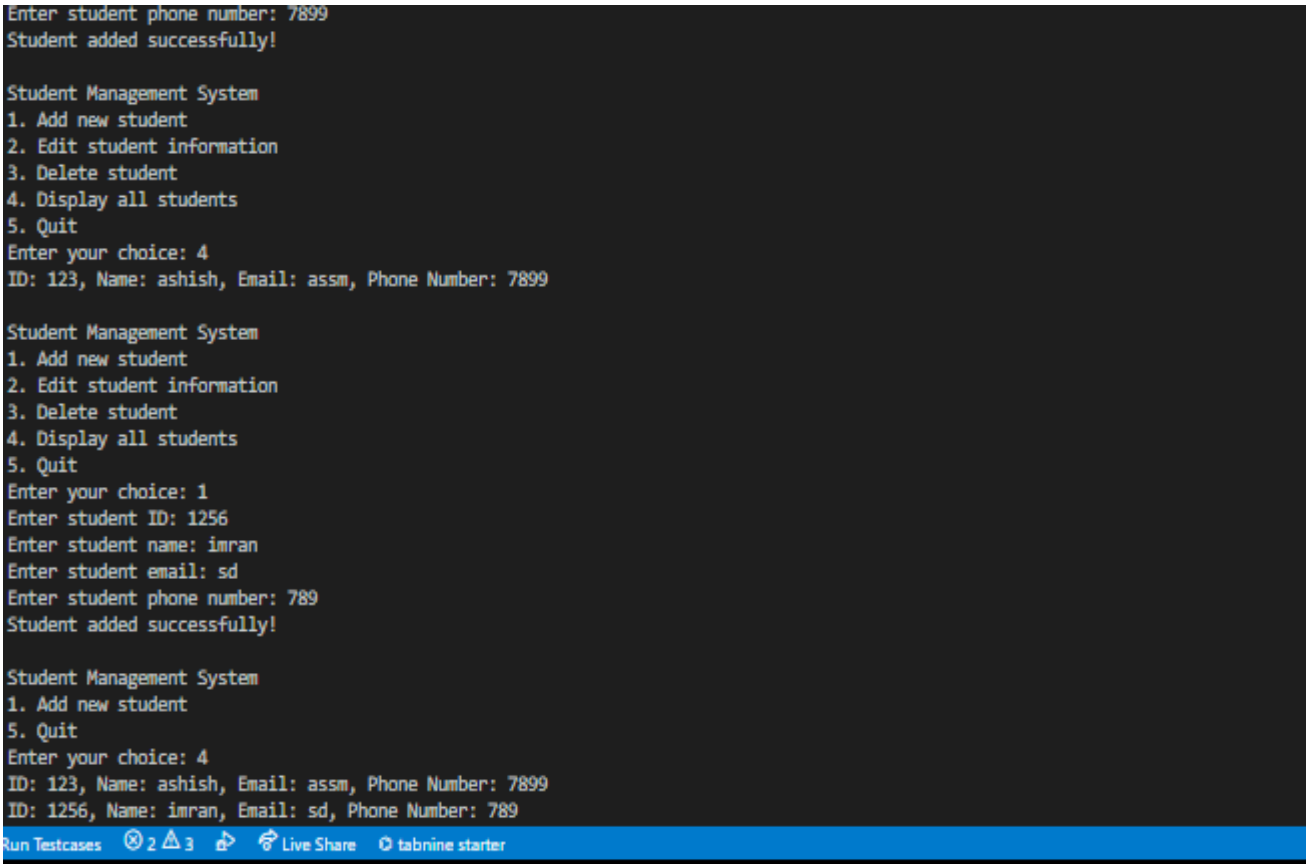
SCREENSHOTS OF THE CODE



```java
import java.util.ArrayList;
import java.util.Scanner;

public class StudentManagementSystem {

    static ArrayList<Student> studentList = new ArrayList<Student>();

    Run | Debug
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        boolean quit = false;

        do {
            System.out.println("\nStudent Management System");
            System.out.println("1. Add new student");
            System.out.println("2. Edit student information");
            System.out.println("3. Delete student");
            System.out.println("4. Display all students");
            System.out.println("5. Quit");
            System.out.print("Enter your choice: ");

            int choice = scanner.nextInt();

            switch(choice) {
                case 1:
                    addStudent();
                    break;
                case 2:
                    editStudent();
                    break;
                case 3:
                    deleteStudent();
                    break;
                case 4:
                    displayAllStudents();
                    break;
                case 5:
                    quit = true;
                    break;
                default:
                    System.out.println("Invalid choice!");
            }
        } while(!quit);

        scanner.close();
    }

    public static void addStudent() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter student ID: ");
```

```java
    public static void addStudent() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter student ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter student name: ");
        String name = scanner.nextLine();
        System.out.print("Enter student email: ");
        String email = scanner.nextLine();
        System.out.print("Enter student phone number: ");
        String phoneNumber = scanner.nextLine();
        Student student = new Student(id, name, email, phoneNumber);
        studentList.add(student);
        System.out.println("Student added successfully!");
    }

    public static void editStudent() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter student ID to edit: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        Student student = getStudentById(id);
        if(student != null) {
            System.out.print("Enter new student name: ");
            String name = scanner.nextLine();
            System.out.print("Enter new student email: ");
            String email = scanner.nextLine();
            System.out.print("Enter new student phone number: ");
            String phoneNumber = scanner.nextLine();
            student.setName(name);
            student.setEmail(email);
            student.setPhoneNumber(phoneNumber);
            System.out.println("Student information updated successfully!");
        } else {
            System.out.println("Student not found!");
        }
    }

    public static void deleteStudent() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter student ID to delete: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        Student student = getStudentById(id);
        if(student != null) {
            studentList.remove(student);
            System.out.println("Student deleted successfully!");
        } else {
            System.out.println("Student not found!");
```

OUTPUT OF THE CODE: -



CONCLUSION

Student Management System can be used by educational institutions to maintain their student records easily. Achieving this objective is difficult using the manual system as the information is scattered, can be redundant, and collecting relevant information may be very time-consuming. All these problems are solved by this project.

This system helps in maintaining the information of pupils of the organization. It can be easily accessed by the manager and kept safe for a long period of time without any changes.