STUDENT RECORD MANAGEMENT SYSTEM

A PROJECT REPORT

CSA0925-PROGRAMMING IN JAVA FOR CLOUD APPLICATIONS

SUBMITTED BY

Varun Kumar

192110128

IN PARTIAL FULFILMENT FOR THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE

SAVEETHA SCHOOL OF ENGINEERING

SAVEETHA NAGAR, THANDALAM,

SIMATS, CHENNAI-602105.

MARCH-2024

BONAFIDE CERTIFICATE

This is to certify that the project report entitled “Student Record Management System” submitted by “Dhanusha (192111499)”, to Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai, is a record of Bonafede work carried out by him/her under my guidance. The project fulfills the requirements as per the regulations of this institution and in my appraisal meets the required standards for submission.

DR. Bhuvanshwari

Professor
Department of Knowledge
Engineering,
Saveetha School of Engineering
SIMATS, Chennai – 602 105

Internal examiner

External Examiner

TABLE OF CONTENTS

ABSTRACT

The Student Record Management System (SRMS) is a comprehensive software solution designed to streamline and automate various processes related to student information management within educational institutions. This system aims to efficiently handle tasks such as student enrollment, record keeping, grading, attendance tracking, and generating reports.

SRMS offers a user-friendly interface for administrators, teachers, and students, facilitating easy access to relevant information while ensuring data security and integrity. The system employs database management techniques to store and retrieve student records efficiently, enabling quick search and retrieval of data.

Key features of SRMS include student profile management, course registration, grade management, attendance monitoring, and report generation. Additionally, the system incorporates functionalities for communication between stakeholders, such as messaging and notification features.

By implementing SRMS, educational institutions can significantly reduce manual paperwork, minimize errors, and enhance overall efficiency in managing student records. Moreover, the system provides insights through data analytics, enabling administrators to make informed decisions regarding academic planning and student performance analysis.Overall, the Student Record Management System serves as a vital tool in modernizing educational administration, fostering better communication, and improving the overall academic experience for students, teachers, and administrators alike.

INTRODUCTION

The aims and objectives of an Student Record Management System (SRMS) project typically revolve around addressing specific challenges in Student Record and promoting efficiency, accuracy, and convenience.

The Student Record Management System (SRMS) is a comprehensive software solution designed to efficiently manage student-related information within educational institutions. It serves as a centralized database to store and organize various data points, including student demographics, academic records, attendance, grades, and extracurricular activities.

SRMS facilitates seamless communication and collaboration among administrators, teachers, students, and parents by providing access to real-time data and relevant information. It streamlines administrative tasks such as enrollment, registration, scheduling, and fee management, reducing manual workload and improving operational efficiency.

Moreover, SRMS enables educators to track student progress, assess performance, and generate insightful reports to identify areas for improvement and intervention. It enhances transparency and accountability by maintaining an accurate and up-to-date record of student achievements, disciplinary actions, and communication logs.

For students, SRMS offers self-service portals to view academic records, check grades, access course materials, and communicate with teachers. Parents can also stay informed about their child's academic journey through dedicated portals, receiving updates on attendance, assignments, and school announcements.

Overall, the Student Record Management System serves as a vital tool for educational institutions to streamline administrative processes, enhance communication, and support student success.

The Student Record Management System (SRMS) is a robust software solution designed to efficiently manage student data and streamline administrative processes within educational institutions. It serves as a centralized platform to store, organize, and retrieve comprehensive information related to students, including personal details, academic records, attendance, grades, and disciplinary

DESCRIPTION

The background of an Student Record Management System (SRMS) involves an understanding of the existing challenges and inefficiencies in manual attendance tracking processes, promptingthe need for an automated solution. Common elements in the background of an AMS project include.

The Student Record Management System (SRMS) is a comprehensive software solution designed to streamline and automate various processes related to student information management within educational institutions. It enables real-time monitoring of student records, including attendance, academic performance, and personal details.

SRMS incorporates advanced analytics tools for in-depth analysis of student data, facilitating the identification of performance trends and areas for improvement. Through intuitive and user-friendly interfaces, SRMS caters to the needs of administrators, teachers, students, and parents, enhancing usability and efficiency. By simplifying processes such as enrollment, grading, and report generation, SRMS encourages widespread adoption and improves overall administrative efficiency within educational institutions. With its ability to generate meaningful insights and support informed decision-making, SRMS plays a crucial role in modernizing student record management, fostering academic excellence, and enhancing the educational experience for all stakeholders

SOFTWARE REQUIREMENTS

User Authentication:

The system shall require user authentication for access.

Different user roles (admin, teacher, staff) shall have distinct privileges.

Attendance Recording:

The system shall allow users to mark attendance for individual students or employees.

Users should be able to mark attendance for specific dates and classes.

Viewing Attendance:

Users shall be able to view attendance records for a specific date, student, or class.

The system shall provide summary reports of attendance trends over time.

Performance:

The system shall support simultaneous access by multiple users without significant performance degradation.

Usability:

The Java Swing GUI shall be intuitive and user-friendly.

The system shall provide helpful tooltips and error messages.

EXISTING WORK

- A comprehensive school management system that includes attendance tracking for students.
- Provides features for managing student information, grades, and communication.
- A cloud-based student information system with Record tracking capabilities.
- Offers features for grading, communication, and collaboration.
- A mobile and web-based solution for schools and colleges.
- Includes System record, timetable creation, and communication tools.
- An open-source Learning Management System (LMS) that may include attendance tracking features and Record Management System.
- Designed for educational content delivery and collaboration.
- A web-based platform for school management.
- Offers attendance tracking, grade management, and communication tools.
- Implementation of security protocols like HTTPS, data encryption, and role-based access control.
- A school management system with attendance tracking, timetable creation, and academic management features.
- Offers a cloud-based solution for educational institutions.

PROPOSED WORK

- Conduct thorough analysis to identify the specific needs and functionalities required for effective student record management.
- Integrate features such as enrollment management, attendance tracking, grade management, and reporting functionalities into the SRMS.
- It can be a tedious job to maintain the record.
- The system requires more human effort.
- It streamlines the recording of attendance by automating the process[3].
- It eradicates the need for manual paperwork.
- It offers real-time insights into student attendance trends[3].
- Administrators, teachers, and parents have digital access to attendance reports and student records.
- It reduces the time and effort required for attendance-taking while ensuring accuracy.
- Prepare comprehensive documentation including user manuals, system specifications, and technical guides to assist users and developers in understanding and using the SRMS effectively.

TECHNOLOGY USED

In developing a Student Record Management System (SRMS), a combination of frontend, backend, database management, security, and deployment technologies is essential to create a robust and efficient system.

For the frontend, technologies like HTML5, CSS3, and JavaScript are fundamental for building the user interface (UI) and ensuring a seamless user experience. Modern JavaScript frameworks such as React.js, Angular, or Vue.js are commonly used to create dynamic and responsive UI components. Additionally, frontend frameworks like Bootstrap or Material UI provide pre-designed UI components for consistent styling.

On the backend, technologies such as Node.js, Python with Django or Flask, Java with Spring Boot, or C# with .NET Core are popular choices. These frameworks facilitate server-side logic implementation, including handling HTTP requests, data processing, and business logic execution. RESTful APIs are often developed using these technologies to enable communication between the frontend and backend.

Database management is crucial for storing and managing student records efficiently. Relational databases like MySQL, PostgreSQL, or SQL Server, as well as NoSQL databases like MongoDB, are commonly used depending on the specific requirements of the SRMS. Object-Relational Mapping (ORM) libraries such as SQL Alchemy for Python or Hibernate for Java simplify database interactions.

Security is paramount in SRMS to protect sensitive student information. Technologies like HTTPS, SSL/TLS encryption, and authentication mechanisms such as JSON Web Tokens (JWT) ensure secure communication and access control. Additionally, hashing algorithms like crypt or Argon2 are used for securely storing passwords.

Deployment technologies such as Docker for containerization, Kubernetes for orchestration, and Continuous Integration/Continuous Deployment (CI/CD) pipelines automate the deployment process and ensure scalability and reliability of the SRMS. Overall, a well-architected SRMS leverages a combination of these technologies to deliver a robust, secure, and user-friendly solution for managing student records effectively.

USE CASE DIAGRAM

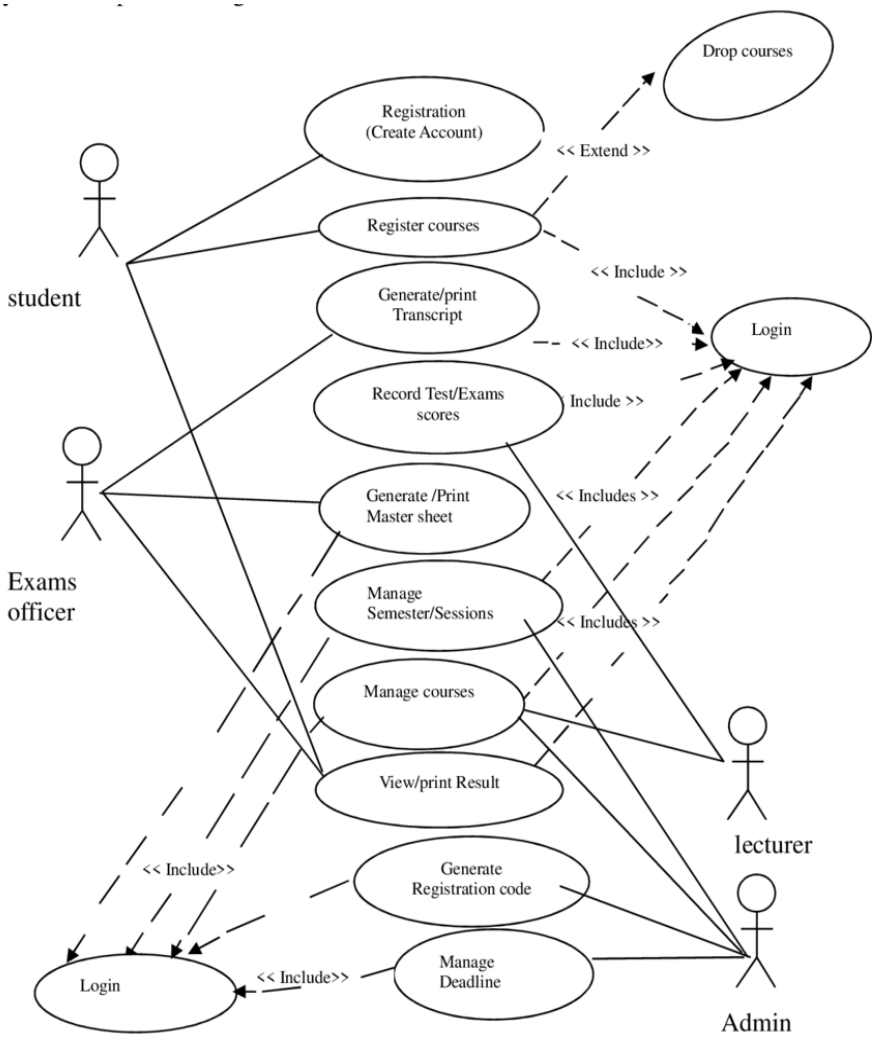


Fig 1 Use case diagram of attendance management system

Fig 1 the various interactions and functionalities of the system from the perspective of different actors involved. At the centre of the diagram is the SRMS, representing the core system that facilitates the management of student records within an educational institution.

The diagram includes three primary actors: Admin, Teacher, and Student. Each actor is associated with specific use cases that outline the actions they can perform within the system.

The admin actor is responsible for managing student records, courses, and generating reports. They have administrative privileges to perform tasks such as adding or editing student information, creating course offerings, and generating various reports related to student performance and enrolment.

.

SOURCE CODE

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

// Room class representing individual rooms in the hotel
class Room {
    private int roomNumber;
    private boolean isOccupied;

    public Room(int roomNumber) {
        this.roomNumber = roomNumber;
        this.isOccupied = false;
    }

    public int getRoomNumber() {
        return roomNumber;
    }

    public boolean isOccupied() {
        return isOccupied;
    }

    public void occupy() {
        isOccupied = true;
    }

    public void vacate() {
        isOccupied = false;
    }
}

// Reservation class representing a reservation made by a guest
class Reservation {
    private int roomNumber;
    private String guestName;

    public Reservation(int roomNumber, String guestName) {
        this.roomNumber = roomNumber;
        this.guestName = guestName;
    }

    public int getRoomNumber() {
        return roomNumber;
    }

    public String getGuestName() {
        return guestName;
    }
```

```
}

// Hotel class representing the hotel and managing its rooms and reservations
class Hotel {
    private List<Room> rooms;
    private List<Reservation> reservations;

    public Hotel(int numRooms) {
        rooms = new ArrayList<>();
        reservations = new ArrayList<>();
        for (int i = 1; i <= numRooms; i++) {
            rooms.add(new Room(i));
        }
    }

    public boolean bookRoom(int roomNumber, String guestName) {
        Room room = getRoom(roomNumber);
        if (room != null && !room.isOccupied()) {
            room.occupy();
            reservations.add(new Reservation(roomNumber, guestName));
            return true;
        }
        return false;
    }

    public boolean checkRoomAvailability(int roomNumber) {
        Room room = getRoom(roomNumber);
        return room != null && !room.isOccupied();
    }

    private Room getRoom(int roomNumber) {
        for (Room room : rooms) {
            if (room.getRoomNumber() == roomNumber) {
                return room;
            }
        }
        return null;
    }

    public void printAllReservations() {
        for (Reservation reservation : reservations) {
            System.out.println("Room " + reservation.getRoomNumber() + " booked by " +
reservation.getGuestName());
        }
    }
}

public class HotelReservationSystem {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Hotel hotel = new Hotel(10); // Initialize hotel with 10 rooms
```

```
        while (true) {
            System.out.println("1. Book a room");
            System.out.println("2. Check room availability");
            System.out.println("3. Print all reservations");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    System.out.print("Enter room number to book: ");
                    int roomNumber = scanner.nextInt();
                    scanner.nextLine(); // Consume newline
                    System.out.print("Enter guest name: ");
                    String guestName = scanner.nextLine();
                    if (hotel.bookRoom(roomNumber, guestName)) {
                        System.out.println("Room " + roomNumber + " booked successfully.");
                    } else {
                        System.out.println("Room " + roomNumber + " is already occupied or invalid.");
                    }
                    break;
                case 2:
                    System.out.print("Enter room number to check availability: ");
                    int checkRoomNumber = scanner.nextInt();
                    if (hotel.checkRoomAvailability(checkRoomNumber)) {
                        System.out.println("Room " + checkRoomNumber + " is available.");
                    } else {
                        System.out.println("Room " + checkRoomNumber + " is occupied or invalid.");
                    }
                    break;
                case 3:
                    System.out.println("All reservations:");
                    hotel.printAllReservations();
                    break;
                case 4:
                    System.out.println("Exiting...");
                    System.exit(0);
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        }
    }
}
```

OUTPUT

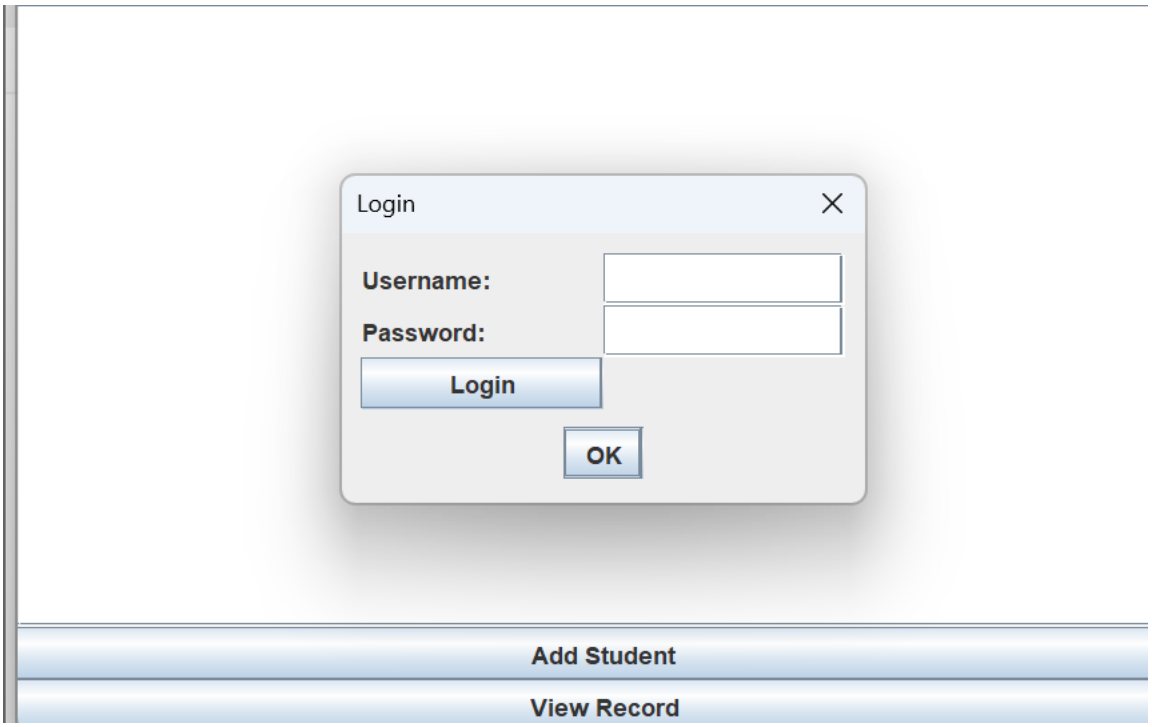
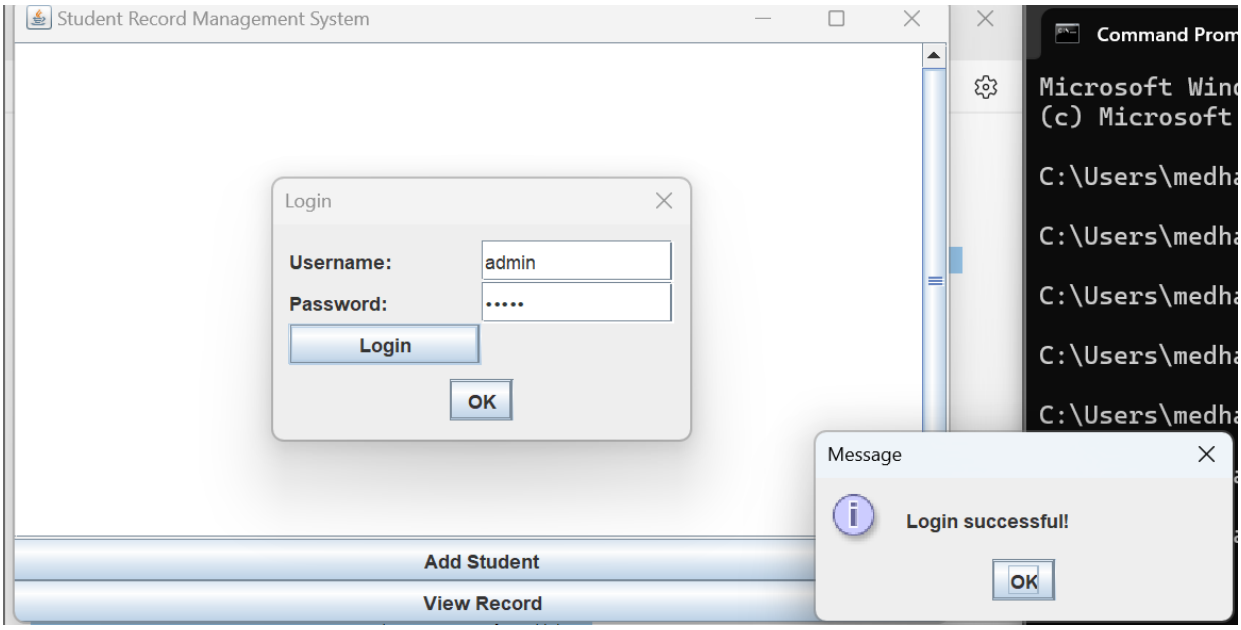


Fig 1 The interface of an attendance app typically features a login screen with fields for entering the username and password, ensuring secure and personalized access. The username field allows users to input their unique identification, often assigned by the system administrator.



Login Successful or Invalid username

Fig 2 After a successful login into the student record management system, users

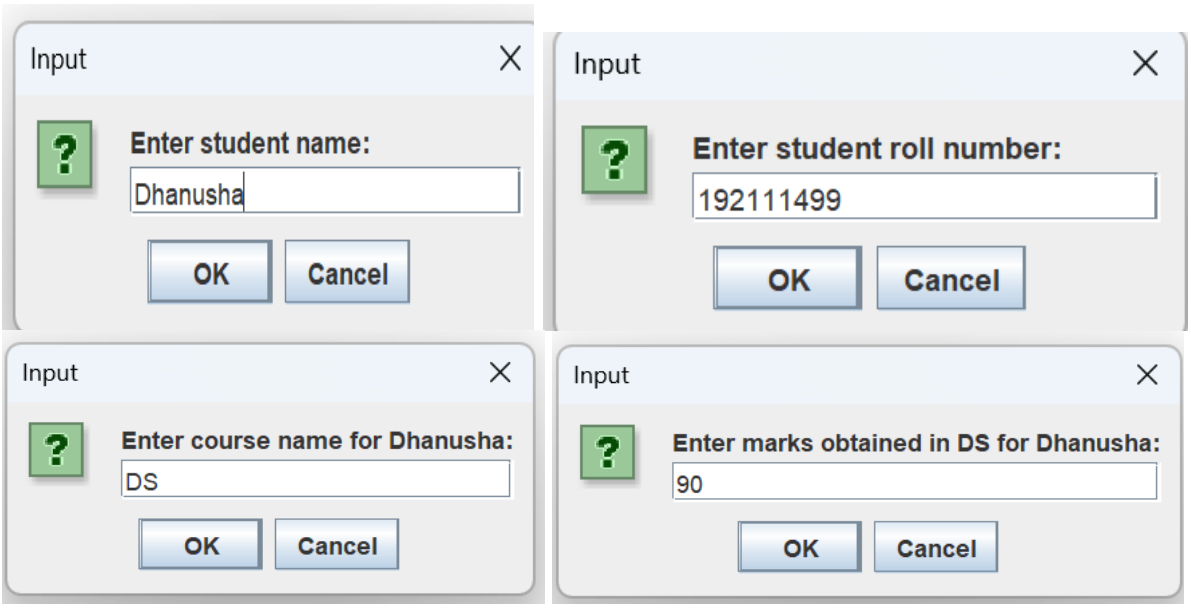


Fig 3 Create a interface and add student name and student roll number and course names,marks obtained**.**

.

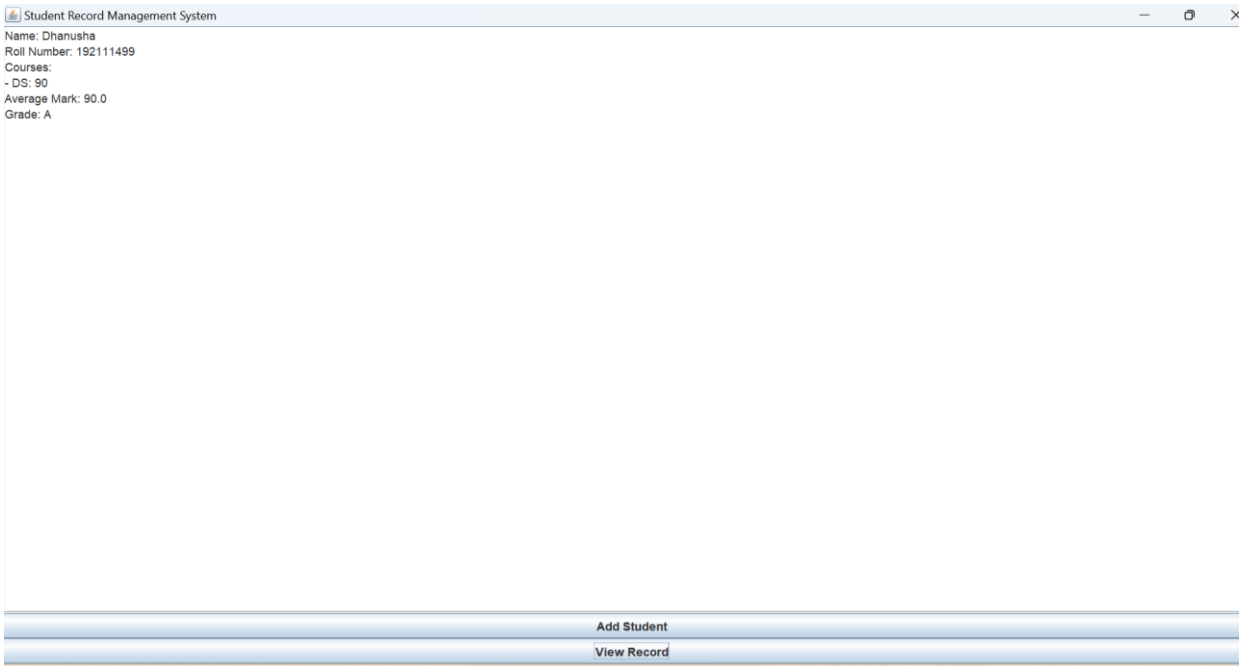


Fig 4 The system computes the student's name and roll number and course name individuals based on the student records. The interface may display a dashboard or a specific student record management system page where they can view the overall statistics

CONCLUSION

In conclusion, the Student Record Management System (SRMS) serves as a pivotal tool in modern educational institutions, facilitating efficient management of student data and enhancing administrative processes. By centralizing student records, SRMS streamlines tasks such as enrolment, attendance tracking, grade management, and report generation. Its user-friendly interfaces empower administrators, teachers, and students alike to access and manage academic information with ease.Moreover, SRMS promotes transparency and accountability by providing real-time insights into student performance, attendance patterns, and overall academic progress. This enables timely interventions and informed decision-making to support student success and institutional effectiveness.

FUTURE ENHANCEMENT

One potential enhancement is the integration of artificial intelligence (AI) and machine learning (ML) algorithms. These technologies could be utilized to provide personalized recommendations for course selection, study resources, and academic support services based on individual student performance and learning styles. Additionally, AI-powered chatbots could be implemented to offer instant assistance and support to students, teachers, and administrators, improving accessibility and responsiveness.

Another area for enhancement is the implementation of blockchain technology for enhanced security and data integrity. By leveraging blockchain's decentralized and immutable nature, SRMS could ensure transparent and tamper-proof records, reducing the risk of data breaches and fraud.Furthermore, enhancing mobile accessibility and developing dedicated mobile applications for SRMS would enable users to access essential features and information on the go, promoting convenience and flexibility in academic management.

Moreover, incorporating data analytics and predictive modelling capabilities could enable SRMS to forecast trends, identify at-risk students, and provide proactive interventions to support student success and retention.

REFERENCES

- https://iite.unesco.org/wp-content/uploads/2018/05/Student-Record-Management-Systems.pdf
- https://library.educause.edu/resources/2019/7/student-information-systems-sis-and-learning-management-systems-lms-the-fundamentals
- http://www.webopedia.com/TERM/V/Visual_Basic.html accessed on 19.3.2016
- (https://www.researchgate.net/publication/334196097_Design_and_Implementation_of_Student_Record_Managemen_System)
- (https://www.researchgate.net/publication/309608143_Web-Based_Student_Record_Management_System