National Institute of Textile Engineering and Research

<u>Project Report</u>

<u>Project Title:</u> Student Management System

<u>Course Name:</u> Database Management Systems Lab - I

<u>Course Code:</u> CSE-2211

Submitted By:

Md. Firoj Miah (CS-2102037)

Md. Sohag Hossain (CS-2102016)

Ashik Iqbal (CS-2102032)

Israt Jahan (CS-2102021)

Labony Aktar (CS-2102026)

Afia Tabassum (CS-2102023)

Submitted To:

Anichur Rahman
Assistant Professor
Department of Computer Science & Engineering (CSE)
National Institute of Textile Engineering and Research

Contents

Introduction

The Student Management System (SMS) is a comprehensive database management system designed to streamline and organize student-related processes within an educational institution. It serves as a centralized platform for storing, managing, and accessing student information efficiently. By leveraging modern technology, SMS enhances administrative tasks, facilitates communication, and improves overall student experience. This project aims to develop a robust and user-friendly SMS to meet the diverse needs of educational institutions, enabling them to effectively manage student data and optimize administrative workflows.

Motivations

Motivations for Implementing a Student Management System (SMS):

- *Streamlining Administrative Tasks:*

  - SMS simplifies administrative duties by automating processes such as enrollment, attendance tracking, and grading.

  - It reduces the time and effort required to manage student data, allowing administrators to focus on other important tasks.

- *Error Reduction:*

  - By digitizing student data, SMS minimizes manual entry errors and data inconsistencies.

  - It ensures accuracy in student records, leading to more reliable reporting and decision-making.

- *Enhanced Communication:*

  - SMS facilitates improved communication among stakeholders, including administrators, teachers, and parents.

  - It provides a centralized platform for sharing important information, announcements, and updates regarding student progress and activities.

- *Efficiency and Accessibility:*

  - The system enhances efficiency by providing quick access to student information from anywhere, at any time.

  - Administrators, teachers, and parents can easily retrieve relevant data, such as grades, attendance records, and contact information, through the SMS platform.

- *Valuable Analytics:*

- SMS offers valuable insights through analytics tools, enabling administrators to track student performance, identify trends, and make data-driven decisions.

- These analytics help in identifying areas for improvement and implementing targeted interventions to support student success.

- *Scalability:*

  - SMS is scalable, allowing educational institutions to adapt and grow without compromising system performance.

  - It can accommodate changes in student enrollment, curriculum, and administrative needs as institutions expand or evolve.

- *Focus on Education:*

  - By reducing administrative burdens, SMS enables educational institutions to allocate more resources and attention to educational initiatives and student support services.

  - It promotes a conducive learning environment where educators can dedicate more time to teaching and student engagement, ultimately enhancing the overall student experience.

Background Study

1. *Importance of Student Management:*

   Managing student data is crucial for educational institutions to ensure efficient operations and support student success. It involves handling information related to admissions, enrollment, attendance, grades, and more. A robust Student Management System (SMS) facilitates the organization, retrieval, and utilization of this data, enabling institutions to streamline administrative processes and provide better support to students.

2. *Analyzing Student Management Lifecycle:*

   The student management lifecycle encompasses various stages, from admission to graduation. Each stage involves specific processes such as enrollment, course registration, attendance tracking, academic performance evaluation, and alumni management. Understanding this lifecycle helps in designing an SMS that addresses the needs of students and administrators at each stage, improving overall efficiency and effectiveness.

3. *Research and Case Study:*

   Researching existing SMS implementations and case studies provides valuable insights into best practices, challenges, and success factors. By studying how other institutions have implemented and benefited from SMS, developers can identify key requirements, potential pitfalls, and innovative solutions. Case studies also help in benchmarking and validating the effectiveness of the proposed SMS solution.

4. *Framework and Standards:*

   Utilizing established frameworks and standards in database management system development ensures the reliability, scalability, and interoperability of the SMS. Frameworks such as MVC (Model-View-Controller) and standards like SQL (Structured Query Language) for database management provide a structured approach to system design and implementation. Adhering to industry standards also facilitates integration with existing systems and interoperability with external applications or databases.

By conducting a comprehensive background study covering the importance of student management, analyzing the student management lifecycle, researching existing solutions and case studies, and adhering to established frameworks and standards, developers can ensure the development of an effective and reliable Student Management System.

Literature Study

Literature Study for Implementing a Student Management System (SMS):

1. *Scope of Literature Review:*

   - The literature review involved analyzing academic papers, articles, and relevant publications related to database management systems and educational technology.

2. *Contemporary Trends Analysis:*

   - We explored current trends in database management systems and educational technology to understand the evolving landscape.

   - This analysis helped us identify advancements and innovations shaping the field.

3. *Established Best Practices:*

   - Our review focused on established best practices in database management systems applicable to educational contexts.

   - We examined methodologies and frameworks that have proven effective in optimizing student data management and system performance.

4. *Exploration of Emerging Technologies:*

   - We investigated emerging technologies within the domain of database management systems and educational technology.

   - This exploration provided insights into potential enhancements and features that could be integrated into our system.

5. *Guidance for System Design and Development:*

- The literature review served as a guide for the design and development of our student management system database.

- It informs decisions regarding architecture, functionality, and usability to ensure the system aligns with current standards and practices in the field.

Problems Statements

1. *Lack of Centralized Coordination:*

   Educational institutions often struggle with decentralized systems for managing student data, leading to fragmentation and inefficiency. Without a centralized platform, coordination among departments becomes challenging, resulting in data silos, duplication of efforts, and inconsistencies in student records.

2. *Manual and Time-Consuming Processes:*

   Many institutions still rely on manual processes for student management, including enrollment, attendance tracking, and grade recording. These manual processes are prone to errors, time-consuming, and resource-intensive. Moreover, they hinder the ability to adapt quickly to changing requirements or scale operations efficiently.

3. *Ineffective Prioritization of Vulnerabilities:*

   With the increasing reliance on digital systems, educational institutions face cybersecurity threats that can compromise the integrity and confidentiality of student data. However, without a systematic approach to vulnerability management, institutions may struggle to prioritize and address security vulnerabilities effectively, leaving them vulnerable to data breaches and cyberattacks.

4. *Overwhelming Vulnerability Scan Reports:*

   Vulnerability scans are essential for identifying security weaknesses in the system. However, the sheer volume of vulnerability scan reports can overwhelm administrators, making it difficult to identify and prioritize critical issues. Without proper tools and processes in place to manage and analyze these reports efficiently, institutions may overlook significant vulnerabilities, putting student data at risk.

Addressing these challenges requires the development of a comprehensive Student Management System (SMS) database management system project that centralizes coordination, automates manual processes, implements effective vulnerability management practices, and provides tools for analyzing vulnerability scan reports systematically.

Objectives

1. Designing a User-Friendly Interface:

- Develop an intuitive and easy-to-navigate interface tailored for administrators, teachers, and students.

- Prioritize user experience by incorporating user feedback and usability testing to ensure efficiency and satisfaction.

2. Providing Real-Time Access to Student Information:

- Enable stakeholders to access up-to-date student information and performance metrics instantaneously.

- Ensure that administrators, teachers, and students have timely access to relevant data for informed decision-making and academic planning.

3. Enhancing Communication and Collaboration:

- Foster improved communication and collaboration among stakeholders within the educational institution.

- Implement features such as messaging systems, discussion forums, and announcement boards to facilitate interaction and information sharing.

4. Implementing Robust Data Management Functionalities:

- Develop robust data management functionalities to maintain the accuracy and integrity of student records.

- Employ data validation techniques and security measures to prevent unauthorized access and data corruption.

5. Streamlining Administrative Tasks:

- Simplify administrative processes such as enrollment, grading, and attendance tracking through automation and optimization.

- Reduce manual intervention and streamline workflows to save time and resources for administrators, teachers, and staff members.

Method

1. *Requirements Gathering:*

   Conduct interviews and surveys with stakeholders including administrators, teachers, and students to understand their needs and expectations from the Student Management System (SMS). Document functional and non-functional requirements comprehensively.

2. *System Design:*

   Design the architecture of the SMS, including database schema, user interface, and system modules. Utilize principles of database management system design to ensure scalability, performance, and data integrity. Employ tools like ER diagrams and UML diagrams for visual representation.

3. *Database Implementation:*

   Develop the database structure based on the design specifications. Utilize appropriate database management system software (e.g., MySQL, PostgreSQL) to create tables, define relationships, and implement data validation constraints. Optimize database performance through indexing and normalization techniques.

4. *Frontend and Backend Development:*

   Implement the frontend user interface for the SMS using web development technologies such as HTML, CSS, and JavaScript. Develop backend functionalities using server-side programming languages (e.g., Python, PHP, Java) to handle user requests, data processing, and business logic implementation.

5. *Integration and Testing:*

   Integrate frontend and backend components to create a cohesive system. Conduct unit testing to ensure individual modules function correctly. Perform integration testing to verify the interaction between different system components. Employ testing frameworks and tools for automated testing where applicable.

6. *Security Implementation:*

   Implement security measures to protect student data from unauthorized access and cyber threats. Utilize encryption techniques to secure data transmission and storage. Implement access control mechanisms to enforce role-based access restrictions. Conduct security testing (e.g., penetration testing, vulnerability scanning) to identify and address potential security vulnerabilities.

7. *Deployment and Maintenance:*

   Deploy the SMS on a suitable hosting environment, considering factors such as scalability, reliability, and cost-effectiveness. Monitor system performance and conduct regular maintenance activities to ensure smooth operation. Gather feedback from users and stakeholders to identify areas for improvement and implement updates as necessary.

8. *Documentation and Training:*

   Prepare comprehensive documentation including user manuals, technical specifications, and system architecture documentation. Provide training sessions for administrators, teachers, and other users to familiarize them with the SMS functionalities and usage guidelines.

9. *Continuous Improvement:*

   Establish mechanisms for collecting user feedback and monitoring system performance post-deployment. Use feedback to prioritize feature enhancements and address usability issues. Adopt agile methodologies for iterative development and continuous improvement of the SMS based on evolving requirements and user needs.

System Design

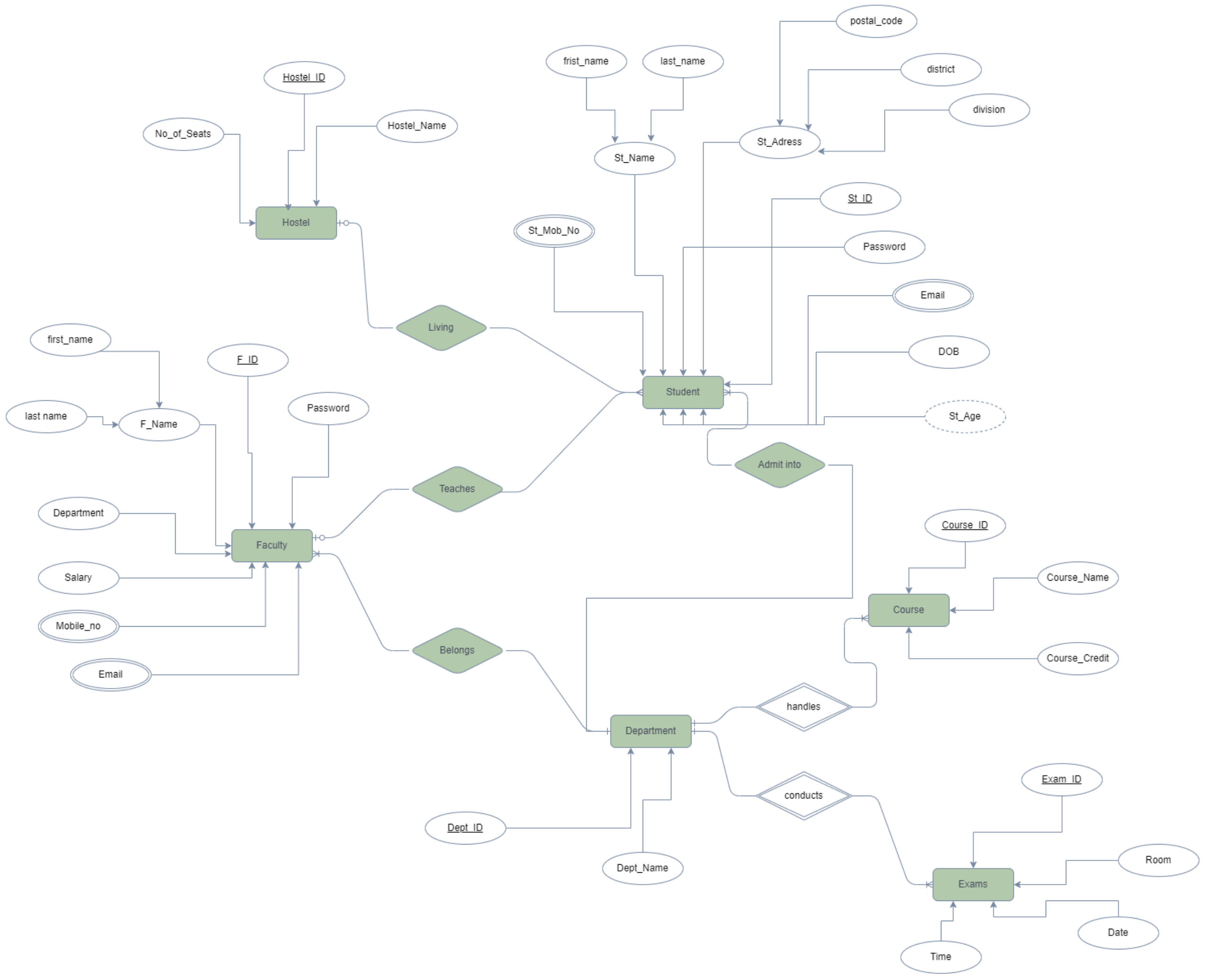

Figure: ER diagram of Student management system

ER diagram to relational model transformation:

1. student(st_ID, first_name, last_name, postal_code, district, division, DOB, password, dept_ID, hostel_ID) where st_ID is the primary key and dept_ID and hostel_ID are foreign keys
2. student_mobile(st_ID, mobile_no) where st_ID is a foreign key
3. student_email(st_ID, email) where st_ID is a foreign key
4. faculty(f_ID, first_name, last_name, department, salary, password) where f_ID is the primary key
5. faculty_mobile(f_ID, mobile_no) where f_ID is a foreign key
6. faculty_email(f_ID, email) where f_ID is a foreign key
7. department(dept_ID, dept_name, f_ID) where dept_id is the primary key and f_ID is a foreign key
8. course(dept_ID, course_ID, course_name, course_credit) where dept_ID and course_ID is together a composite primary key
9. exams(dept_ID, exam_ID, room, date, time) where dept_ID and exam_ID is together a composite primary key
10. hostel(hostel_ID, hostel_name, no_of_seats) where hostel_ID is the primary key

SQL code for database implementation:

```
CREATE DATABASE IF NOT EXISTS student_management_system;

USE student_management_system;


-- Create the 'student' table


CREATE TABLE IF NOT EXISTS student (

   st_ID INT PRIMARY KEY,

   first_name VARCHAR(255),

   last_name VARCHAR(255),

   postal_code VARCHAR(10),

   district VARCHAR(255),

   division VARCHAR(255),

   DOB varchar(20),

   password VARCHAR(255) default "12345",

   dept_ID INT,

   hostel_ID INT,

   FOREIGN KEY (dept_ID) REFERENCES department(dept_ID),

   FOREIGN KEY (hostel_ID) REFERENCES hostel(hostel_ID)

);
```

```
-- Create the 'student_mobile' table

CREATE TABLE IF NOT EXISTS student_mobile (
    st_ID INT,
    mobile_no VARCHAR(15),
    FOREIGN KEY (st_ID) REFERENCES student(st_ID)
);

-- Create the 'student_email' table

CREATE TABLE IF NOT EXISTS student_email (
    st_ID INT,
    email VARCHAR(255),
    FOREIGN KEY (st_ID) REFERENCES student(st_ID)
);

-- Create the 'faculty' table

CREATE TABLE IF NOT EXISTS faculty (
    f_ID INT PRIMARY KEY,
    first_name VARCHAR(255),
    last_name VARCHAR(255),
    dept_ID INT,
    salary DECIMAL(10, 2),
    password VARCHAR(255) default "56789",
    FOREIGN KEY (dept_ID) REFERENCES department(dept_ID)
);

-- Create the 'faculty_mobile' table

CREATE TABLE IF NOT EXISTS faculty_mobile (
    f_ID INT,
    mobile_no VARCHAR(15),
    FOREIGN KEY (f_ID) REFERENCES faculty(f_ID)
);

-- Create the 'faculty_email' table

CREATE TABLE IF NOT EXISTS faculty_email (
    f_ID INT,
    email VARCHAR(255),
    FOREIGN KEY (f_ID) REFERENCES faculty(f_ID)
);

-- Create the 'department' table

CREATE TABLE IF NOT EXISTS department (
    dept_ID INT PRIMARY KEY,
    dept_name VARCHAR(255)
);

-- Create the 'course' table

CREATE TABLE IF NOT EXISTS course (
    dept_ID INT,
    course_ID INT,
    course_name VARCHAR(255),
    course_credit INT,
    PRIMARY KEY (dept_ID, course_ID),
    FOREIGN KEY (dept_ID) REFERENCES department(dept_ID)
);
```

```
-- Create the 'exams' table

CREATE TABLE IF NOT EXISTS exams (
    dept_ID INT,
    exam_ID INT,
    room VARCHAR(255),
    date varchar(20),
    time varchar(20),
    PRIMARY KEY (dept_ID, exam_ID),
    FOREIGN KEY (dept_ID) REFERENCES department(dept_ID)
);

-- Create the 'hostel' table

CREATE TABLE IF NOT EXISTS hostel (
    hostel_ID INT PRIMARY KEY,
    hostel_name VARCHAR(255),
    no_of_seats INT
);
```

Data Insertion:

Insert into student table:

```
INSERT INTO student(st_ID, first_name, last_name, postal_code, district, division,
DOB, password, dept_ID, hostel_ID)
VALUES
(1, "mahmudul",  "hasan",1110 ,  "manikgong",  "Dhaka","1/1/2000", "1234", 4,
1002),
(2, "Shourav",  "Das",1210 ,  "Rajbari",  "Rajbari","1/1/2000", "1234", 4, 1002),
(3, "Rosdania ",  "Ahmed",1510 ,  "Narsingdi",  "Narsingdi","1/1/2000", "1234", 4,
1002),
(4, "Shahrin Siraj ",  "Aurpi",1610 ,  "Gazipur",  "Dhaka","1/1/2000", "1234", 4,
1002),
(5, "Md. Hasan",  "Nawab",1110 ,  "Pabna",  "Dhaka","1/1/2000", "1234", 4, 1002),
(6, "Al",  "momen",1150 ,  "Mymensingh", "Dhaka","1/1/2000", "1234", 4, 1002),
(7, "MST. JANNATUL MAWA",  "ANONNA
",1110 ,  "Savar",  "Dhaka","1/1/2000", "1234", 4, 1002),
(8, "Md. Ahsan ",  "Karim",1220 ,  "Dhaka",  "Dhaka","1/1/2000", "1234", 4, 1002),
(9, "Md. Aslam",  "Ahmed",1940 ,  "Gopalgonj", "Dhaka","1/1/2000", "1234", 4,
1002),
(10, "Samiul",  "Islam",1840 ,  "Mymensingh", "Dhaka","1/1/2000", "1234", 4,
1002),
(11, "Mohammad Rakib ",  "Hossain",1510 ,  "Brahmanbaria", "Dhaka","1/1/2000",
"1234", 4, 1002),
(12, "Foysal ",  "Ahmed",1910 ,  "Shibganj",  "Dhaka","1/1/2000", "1234", 4, 1002),
(13, "Md. Shohag",  "hossain",1410 ,  "Pabna",  "Dhaka","1/1/2000", "1234", 4,
1002),
(14, "Shovona ",  "Sutradhar",1220 ,  "Cumilla",  "Dhaka","1/1/2000", "1234", 4,
1002),
(15, "Md. Sanzid",  " ",1310 ,  "Chandpur", "Dhaka","1/1/2000", "1234", 4, 1002),
(16, "Tanjim ",  "Hasan",1330 ,  "Jamalpur", "Dhaka","1/1/2000", "1234", 4, 1002),
(17, "Shibly ",  "Sadik",1510 ,  "Joypurhat", "Dhaka","1/1/2000", "1234", 4, 1002),
(18, "Israt ",  "Jahan",1890 ,  "Tangail",  "Dhaka","1/1/2000", "1234", 4, 1002),
(19, "Abu Jamil ",  "Sarker",1120 ,  "Dhaka", "Dhaka","1/1/2000", "1234", 4, 1002),
(20, "Afia ",  "Tabassum",1150 ,  "Dhaka",  "Dhaka","1/1/2000", "1234", 4, 1002),
(21, "Md. Mozammel Hossain",  "Tanvir",1910 ,  "Munshiganj",
"Dhaka","1/1/2000", "1234", 4, 1002),
```

(22, "MST. MORIOM AKTHER", "BITHEE",1510 , "Dhaka", , "Dhaka","1/1/2000", "1234", 4, 1002),
(23, "Laboni ", "Akter",1310 , "Dhaka", "Dhaka","1/1/2000", "1234", 4, 1002),
(24, "Touhidul Kabir", "Alvee",1180 , "Mymenshing", "Dhaka","1/1/2000", "1234", 4, 1002),
(25, "Takwa ", "Jerin",1170 , "Cumilla", "Dhaka","1/1/2000", "1234", 4, 1002),
(26, "Ashik ", "Iqbal",1170 , "Jamalpur", "Dhaka","1/1/2000", "1234", 4, 1002),
(27, "RAISA ", "AMIN",1610 , "Dhaka", "Dhaka","1/1/2000", "1234", 4, 1002),
(28, "Feroj ", "Miah",1910 , "Shirajganj", "Dhaka","1/1/2000", "1234", 4, 1002),
(29, "Protiva ", "Bose",1710 , "Pirojpur", "Dhaka","1/1/2000", "1234", 4, 1002),
(30, "Shahnaz Fateme", "Esha",1780 , "Khulna", "Dhaka","1/1/2000", "1234", 4, 1002);

Insert into faculty table:

INSERT INTO faculty (f_ID, first_name, last_name, dept_ID, salary, password) VALUES
(1,"Anichur", "Rahman", "1", 70000, "1234"),
(2,"Dipanjali", "Kundu", "1", 70000, "5678"),
(3,"Umme ", "Sara", "1", 70000, "9123"),
(4,"Mohammad", "Sayduzzaman", "1", 70000, "4567"),
(5,"Sadia", "Sazzad", "1", 70000, "8912"),
(6,"Jarin Tasnim", "Tamanna", "1", 70000, "3456"),
(7,"Shakila ", "Shafiq", "1", 70000, "7891"),
(8,"Suraiya ", "Sultana", "1", 70000, "2345"),
(9,"Dr. Md. Abul", "Kalam", "1", 70000, "6789"),
(10,"Md. Mamun Ur", "Rashid", "2", 70000, "1234"),
(11,"Muaz", "Rahman", "2", 70000, "5678"),
(12,"Md. Ashiqur ", "Rahman", "2", 70000, "9123"),
(13,"Kamrun", "Nahar", "4", 70000, "4567"),
(14,"Md. Didarul", "Islam", "2", 70000, "8912"),
(15,"Masuma", "Akter", "4", 70000, "3456"),
(16,"Sharmin", "Sultana", "4", 70000, "7891"),
(17,"Lailatul", "Nehar", "4", 70000, "2345"),
(18,"Md.Redwanul", "Islam", "5", 70000, "6789"),
(19,"Mowshumi ", "Roy", "3", 70000, "1234"),
(20,"Sadia Binte", "Kausar", "4", 70000, "5678");

Insert into department table:

```
insert into department (dept_id, dept_name)
 values
(1, "Computer Science and Engineering"),
(2, "Eelectrical and Electronics Engineering"),
(3, "Textile Engineering"),
(4, "Industrial and Production Engineering"),
(5, "Fashion Design and Apperal Engineering");
```

Insert into hostel table:

```
insert into hostel (hostel_ID, hostel_name, no_of_seats)
 values
(1001, "Boys Hostel - 1", 500),
(1002, "Boys Hostel Extended - 1", 200),
(1003, "Girls Hostel", 350);
```

Java code implementation:

Student class:

```
package com.feroj.sms;



import java.sql.*;
import java.util.*;

public class Student {
    private final Connection connection;
    private final Scanner scanner;

    public Student(Connection connection, Scanner scanner) {
        this.connection = connection;
        this.scanner = scanner;
    }

    public void addStudent(){
        System.out.print("Student's ID: ");
        int ID = scanner.nextInt();
        System.out.print("First Name: ");
        String FirstName = scanner.next();
        System.out.print("Last Name: ");
        String LastName = scanner.next();
        System.out.print("Postal code: ");
        String PostCode = scanner.next();
        System.out.print("District: ");
        String District = scanner.next();
        System.out.print("Division: ");
        String Division = scanner.next();
        System.out.print("Date of Birth: ");
        String DOB = scanner.next();
        System.out.print("\n1 for CSE\n2 for EEE\n3 for TE\n4 for IPE\n5 for
FDAE\n");
        System.out.print("Select department: ");
        int DeptID = scanner.nextInt();
        System.out.println("Allocated hostel: ");
        int HostelID = scanner.nextInt();

        try {
            String query = "INSERT INTO student(st_ID, first_name, last_name,
postal_code, district, division, DOB, dept_ID, hostel_ID) VALUES (?, ?, ?, ?, ?, ?, ?,
?, ?)";
            PreparedStatement preparedStatement =
connection.prepareStatement(query);
            preparedStatement.setInt(1, ID);
            preparedStatement.setString(2, FirstName);
            preparedStatement.setString(3, LastName);
            preparedStatement.setString(4, PostCode);
            preparedStatement.setString(5, District);
            preparedStatement.setString(6, Division);
            preparedStatement.setString(7, DOB);
            preparedStatement.setInt(8, DeptID);
            preparedStatement.setInt(9, HostelID);
            int affectedRows = preparedStatement.executeUpdate();
            if(affectedRows > 0){
                System.out.println("Student data added Successfully!");
            }else

            {
                        System.out.println("Failed to add student data!");
                    }



                }
catch (SQLException e)

{
            e.printStackTrace();
        }
    }

    public void viewStudent()

{
        String query = "select * from student";

        try{
            PreparedStatement preparedStatement =
connection.prepareStatement(query);
            ResultSet resultSet = preparedStatement.executeQuery();
            System.out.println("Students: ");
```

```
          System.out.println("+-----+--------------+-------------+------+------------+------------
+--------------+--------------+------+------+");


          System.out.println("|ID   | First Name    | Last Name    | P.C  | District    |
Division   | DOB          | Password     | D_ID | H_ID |");
          System.out.println("+-----+--------------+-------------+------+------------+------------
+--------------+--------------+------+------+");
          while (resultSet.next())
{
             int ID = resultSet.getInt("st_ID");
             String Fname = resultSet.getString("first_name");
             String Lname = resultSet.getString("last_name");
             String PostCode = resultSet.getString("postal_code");
             String District = resultSet.getString("district");
             String Division = resultSet.getString("division");
             String DOB = resultSet.getString("DOB");
             String Password = resultSet.getString("password");
             int DeptID = resultSet.getInt("dept_ID");
             int HostelID = resultSet.getInt("hostel_ID");
             System.out.printf("|%-5s|%-14s|%-13s|%-6s|%-12s|%-12s|%-14s|%-
14s|%-6s|%-6s|\n", ID, Fname, Lname, PostCode, District, Division, DOB,
Password, DeptID, HostelID);
             System.out.println("+-----+--------------+-------------+------+------------+-----------
-+--------------+--------------+------+------+");
          }

       }catch (SQLException e)
```

```
{
         e.printStackTrace();
      }
   }

   public void getStudentByID(int id){
      String query = "select * from student where st_ID = ?";
      try {
         PreparedStatement preparedStatement =
connection.prepareStatement(query);
         preparedStatement.setInt(1, id);
         ResultSet resultSet = preparedStatement.executeQuery();
         if(resultSet.next())
{
            int ID = resultSet.getInt("st_ID");
            String Fname = resultSet.getString("first_name");
            String Lname = resultSet.getString("last_name");
            String PostCode = resultSet.getString("postal_code");
            String District = resultSet.getString("district");
            String Division = resultSet.getString("division");
            String DOB = resultSet.getString("DOB");
            String Password = resultSet.getString("password");
            int DeptID = resultSet.getInt("dept_ID");
            int HostelID = resultSet.getInt("hostel_ID");

            System.out.println("Student ID: " + ID);
            System.out.println("Student's Name: " + Fname + " " + Lname);
            System.out.println("Address: " + PostCode + "-" + District + ", " + Division);
            System.out.println("Date of Birth: " + DOB);
            System.out.println("Department ID: " + DeptID);
            System.out.println("Hostel ID: " + HostelID);
         } else
         {
                System.out.println("Student not found!");
              }
            }
catch(SQLException e)
{
        e.printStackTrace();
    }
  }

}
```

Faculty class:

```
package com.feroj.sms;

import java.sql.*;

public class Faculty {
   private final Connection connection;

   public Faculty(Connection connection)
{
       this.connection = connection;
   }


   public void viewFaculty()
{
       String query = "select * from faculty";

       try{
          PreparedStatement preparedStatement =
connection.prepareStatement(query);
          ResultSet resultSet = preparedStatement.executeQuery();
          System.out.println("Faculties: ");
          System.out.println("+-----+--------------+------------+-----------+-------------+");
          System.out.println("|ID   | First Name   | Last Name   | Dept_ID   |
Password     |");
          System.out.println("+-----+-------------+------------+----------+-------------+");
          while (resultSet.next()){
             int ID = resultSet.getInt("f_ID");
             String Fname = resultSet.getString("first_name");
             String Lname = resultSet.getString("last_name");
             int DeptID = resultSet.getInt("department");
             String Password = resultSet.getString("password");
             System.out.printf("|%-5s|%-14s|%-13s|%-11s|%-15s|\n", ID, Fname,
Lname, DeptID, Password);
             System.out.println("+-----+-------------+------------+----------+-------------+");
          }

       }catch (SQLException e){
          e.printStackTrace();
       }
   }

   public void getFacultyByID(int id){
       String query = "select * from faculty where f_ID = ?";
       try {
          PreparedStatement preparedStatement =
connection.prepareStatement(query);
          preparedStatement.setInt(1, id);
          ResultSet resultSet = preparedStatement.executeQuery();
          if(resultSet.next()){
             int ID = resultSet.getInt("f_ID");
             String Fname = resultSet.getString("first_name");
```

```
                String Lname = resultSet.getString("last_name");
                int DeptID = resultSet.getInt("dept_ID");
                String Password = resultSet.getString("password");

                System.out.println("Faculty ID: " + ID);
                System.out.println("Faculty's Name: " + Fname + " " + Lname);
                System.out.println("Department ID: " + DeptID);
            } else
{
                System.out.println("Faculty not found!");
            }
        }catch(SQLException e){
            e.printStackTrace();
        }
    }
}
```

StudentManagementSystem class:

```
package com.feroj.sms;

import java.sql.Connection;
import java.sql.DriverManager;
import java.util.Scanner;

public class StudentManagementSystem {
   private static final String url =
"jdbc:mysql://127.0.0.1:3306/student_management_system";
   private static final String username = "root";
   private static final String password = "admin";

   public static void main(String[] args) {
      try {
         Class.forName("com.mysql.cj.jdbc.Driver");
      } catch (Exception e){
         e.printStackTrace();
      }
      Scanner scanner = new Scanner(System.in);
      try {
         Connection connection = DriverManager.getConnection(url, username,
password);
         Student student = new Student(connection, scanner);
         Faculty faculty = new Faculty(connection);
```

```
while(true){
    System.out.println("STUDENT MANAGEMENT SYSTEM");
    System.out.println("1. Add Student");
    System.out.println("2. See Student List");
    System.out.println("3. Find Student by ID");
    System.out.println("4. See Faculty List");
    System.out.println("5. Find Faculty by ID");
    System.out.println("0. Exit");

    System.out.print("Enter your choice: ");
    int choice = scanner.nextInt();

    switch (choice){
        case 1:
            //add student
            student.addStudent();
            break;
        case 2:
            //see student list
            student.viewStudent();
            break;
        case 3:
            //find student by id
            System.out.print("Enter the Student's ID: ");
            int id = scanner.nextInt();
            student.getStudentByID(id);
            break;
        case 4:
```

```
                    //see faculty list
                    faculty.viewFaculty();
                    break;
                case 5:
                    //find faculty by id
                    System.out.print("Enter the Faculty's ID: ");
                    int f_id = scanner.nextInt();
                    faculty.getFacultyByID(f_id);
                    break;
                case 0:
                    System.out.println("Thank you for using our system!");
                    System.exit(0);
                default:
                    System.out.print("Enter your choice: ");
            }
        }
    }catch (Exception e){
        e.printStackTrace();
    }
  }
}
```

Conclusion

- *Enhanced Efficiency and Accessibility:*

  The implementation of the Student Management System (SMS) significantly improves the efficiency of administrative processes by centralizing coordination and automating manual tasks. This streamlines operations, reduces errors, and ensures timely access to accurate student information for administrators, teachers, and students.

- *Improved Data Security and Compliance:*

  With the SMS's implementation, educational institutions benefit from enhanced data security measures, including encryption, access controls, and regular security testing. These measures safeguard sensitive student information, ensuring compliance with regulatory requirements and mitigating the risk of data breaches or unauthorized access.

- *Empowering Student Success:*

  By providing a user-friendly interface, comprehensive functionalities, and timely access to relevant information, the SMS empowers students to take control of their educational journey. It facilitates communication between students and educators, supports academic planning, and fosters a conducive learning environment, ultimately contributing to student success and satisfaction.

Limitations and Future Works

Limitations:

1. *Scalability Challenges:*

   The current implementation of the Student Management System (SMS) may face scalability challenges as the volume of student data and user base grows. As the system scales, it may encounter performance issues or resource constraints that need to be addressed to maintain optimal functionality.

2. *Limited Integration Capabilities:*

   SMS may have limited integration capabilities with other systems or platforms commonly used in educational institutions, such as learning management systems or financial management software. This limitation could hinder the seamless flow of data and information across different systems, leading to data silos and inefficiencies.

3. *User Adoption and Training:*

   Adoption of the SMS by administrators, teachers, and students may face resistance or challenges due to unfamiliarity with the system or reluctance to change existing workflows. Adequate training and support mechanisms may be required to ensure smooth adoption and utilization of the SMS functionalities.

Future Work:

1. *Enhanced Scalability:*

   Future work could focus on enhancing the scalability of the SMS to accommodate growing data volumes and user loads. This may involve optimizing database performance, implementing distributed architectures, or leveraging cloud computing resources to ensure scalability without compromising performance.

2. *Integration with External Systems:*

Further development could focus on enhancing the integration capabilities of the SMS with external systems and platforms used in educational institutions. This could involve implementing standardized APIs, data exchange formats, or middleware solutions to facilitate seamless data integration and interoperability.

3. *Advanced Analytics and Reporting:*

   Future work could include the development of advanced analytics and reporting functionalities within the SMS. This could enable administrators and educators to gain deeper insights into student performance, trends, and behaviors, facilitating data-driven decision-making and personalized support interventions.

4. *Enhanced Security Features:*

   Continuous improvement of security features is essential to address emerging threats and ensure the protection of sensitive student data. Future work could focus on implementing advanced security measures such as threat detection, anomaly detection, and behavior analysis to enhance the SMS's resilience against cyber threats.

5. *User Experience Enhancements:*

   Future work could involve refining the user interface and experience of the SMS to improve usability, accessibility, and user satisfaction. This could include conducting user feedback sessions, usability testing, and iterative design improvements to create a more intuitive and user-friendly interface.

References

1. *https://www.nveo.org/index.php/journal/article/download/5238/4150/5313*

2. *https://www.researchgate.net/publication/353650862_Design_and_Implementation_of_Student_Management_System_Based_on_Internet_Big_Data*

3. *https://www.javatpoint.com/er-diagram-for-student-management-system*