BASAVARAJESWARIGROUPOFINSTITUTIONS

BALLARIINSTITUTEOFTECHNOLOGY&MANAGEMENT

AUTONOMOUS INSTITUTEUNDER VISVESVARAYA TECHNOLOGICALUNIVERSITY

JNANASANGAMA,BELAGAVI590018

DEPARTMENTOFDATASCIENCEENGINEERING

Report on

"Student Academic Management System"

For the course:

DATABASE MANAGEMENT SYSTEM–22CD43

Submitted by

S GAYATHRI-3BR23CD080

Max Marks	**:**	**04**
Marksallotted	**:**	

Signatureofthestaff

BALLARIINSTITUTEOFTECHNOLOGY&MANAGEMENT

NACCAccreditedinstitution*
(RecognisedbyGovt.ofkarnataka,approvedbyAICTE,NewDelhi&AffiliatedtoVisvesvaraya Technological University, Belgavi)
"JnanaGangotri"Campus,No.873/2,Ballari-HospetRoad,Allipur,Ballari-583104
Karnataka,India.
Ph:08392-237100/23719,Fax:08392-237197

2024-2025

TABLE CONTENTS

1. Abstract

The Student Academic Management System (SAMS) is a software application designed to streamline and automate the management of academic data within educational institutions. This system aims to replace traditional paper-based or manual methods by providing a centralized digital platform to manage student records, subjects, attendance, grades, and academic performance efficiently anaccurately.

The system is built with a user-friendly interface and supports role-based access to ensure data privacy and integrity. Administrators can manage academic calendars, generate reports, and oversee institutional performance, while instructors can upload grades, track attendance, and communicate with students.

Students benefit from the ability to access their academic records, enroll in courses, and receive notifications related to academic activities**.**

SAMS aims to reduce manual workload, eliminate redundant data handling, and enable realtime access to academic information. The system's scalable and user-friendly design makes it a valuable tool for modern educational institutions striving for digital transformation and operational excellence.

2. Introduction

1.Definition: A Student/Academic Management System is a digital tool that automates and manages academic and administrative tasks in educational institutions.

2.Users: The system is used by administrators, teachers, students, and sometimes parents for seamless communication and data handling.

3.Features: Common features include student registration, timetable scheduling, grade tracking, attendance monitoring, examination management, and reporting tools.

4.Benefits:Reduces paperwork and manual errors Enhancecommunication between stakeholder Saves time and administrative effortAllows quick access to academic records

5. Technology: Can be implemented as web-based or mobile applications, often using modern technologies like cloud storage, databases, and user authentication systems.

6.Goal: To digitize academic operations and support data-driven decision-making in educational institutions.

3.Justification

- **Data Accuracy and Integrity**

Manual systems are prone to human errors. A digital system ensures more accurate data entry, minimizes duplication, and maintains data integrity through validation mechanisms.

- **Centralized Information Management**

All academic records are stored in a centralized database, making it easy to access, update, and manage student information. This supports informed decision-making and improved coordination among departments.

- **Transparency and Accessibility**

SAMS provides role-based access to students, faculty, and administrators. Students can view grades, attendance, and academic history, while faculty can manage class records and communicate with students more effectively.

- **Scalability and Adaptability**

As institutions grow in size, the system can scale to accommodate more users and functionalities. It can also be adapted to support new academic policies or structures without major disruptions.

- **Environmental Sustainability**

Reducing dependence on paper records contributes to environmental sustainability and aligns with the growing push for green and digital campuses.

- **Improved Communication and Monitoring**

Notifications, academic alerts, and reports help stakeholders stay informed and respond quickly to academic issues, improving the overall quality of education and administration

2. Problem Statement

- Inefficiency and errors in manual record keeping

- Difficulties in managing large student databases
- Lack of quick access to academic data
- There is a lack of centralized systems to store and manage student records such as personal details, grades, attendance, and course information.
- Existing systems (if any) may not be scalable, user-friendly, or adaptable to changing institutional needs.
- Lack of real-time access to academic information hinders quick decisionmaking.
- Communication between students, faculty, and administration is often delayed or fragmented.

3. Objectives

- ❖ Efficient Student Information Management
- ❖ Streamlined Academic Processes
- ❖ Attendance and Discipline Tracking
- ❖ Communication and Collaboration
- ❖ Resource Management
- ❖ Reporting and Analytics
- ❖ User Access Control
- ❖ Integration and Scalability
- ❖ Promote Eco-Friendly Practices
- ❖ Ensure Security and Privacy

4. System Requirements

▶ **Functional Requirements:**These describe what the system should do:User ManagementRegister/login for students, teachers, and adminsRole-based access controlProfile management

- Student Management:Add, edit, delete student recordsTrack attendance and gradesView academic progress reports
- Course & Class Management:Create/manage courses and subjectsAssign teachers to coursesSchedule classes and exams

- Enrollment Management:Student registration for coursesView course catalogPrerequisite and eligibility checks
- Reports & Analytics:Academic performance dashboardsAttendance and fee reportsExport data (CSV, PDF, etc.)

▶ **Non-Functional Requirements:**These describe how the system should perform:PerformanceFast response times(e.g.,<2sfor most queries)Support concurrent users (scale based on institution size)

- Security:Role-based permissionsData encryption (especially for sensitive student data)Regular backups and audit trails
- Scalability:Scalable to handle more students, courses, and campuses
- Usability: Intuitive UI/UX for all rolesAccessible on mobile and desktop devices
- Maintainability: Modular codebase for easy updates Well-documented API and backend
- Availability: 99.9% uptime with failover mechanismsScheduled maintenance notifications

6.Methodology

Methodology: The development of the Student/Academic Management System followed a structured software development lifecycle to ensure a robust and user-friendly application. The methodology is divided into the following phases

1.Requirement Analysis:This phase involved gathering and analyzing the needs of end-users including administrators, faculty, and students.

2.System Design:A detailed design was created using UML diagrams, ER diagrams (for database structure), and wireframes for the user interface. This helped in visualizing the system architecture and data flow between modules such as:

- ➢ Student Registration
- ➢ Attendance Tracking

- Timetable Scheduling
- Course Management
- Grade Management

3. Development: The system was developed using [insert technologies here, e.g., Python/Django, PHP/Laravel, Java/Spring Boot, etc.] for the backend, and HTML/CSS/JavaScript (with or without frameworks) for the frontend. A relational database (e.g., MySQL or PostgreSQL) was used to store academic data securely.

4.Testing: The system underwent multiple levels of testing:Unit Testing for individual componentsIntegration Testing to ensure modules work togetherUser Acceptance Testing (UAT) with actual users to gather feedback

5. Evaluation: Effectiveness was measured by user satisfaction surveys and comparing manual processes vs. automated processes in terms of time efficiency and accuracy.

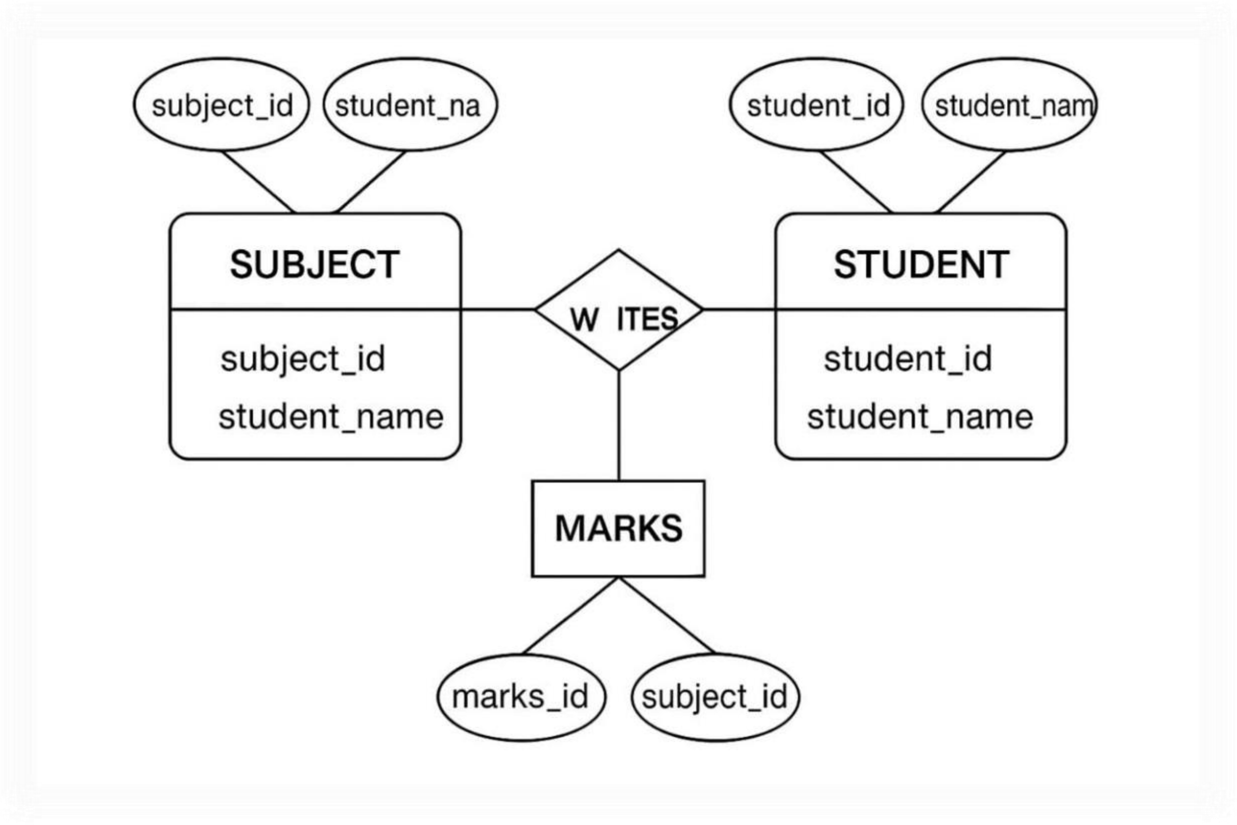
7.ER Diagram

1. Student
   StudentID (PK), FirstName, LastName , DOB , Email ,Phone ,Address, EnrollmentDate
2. Course
   CourseID (PK), CourseName, CourseDescription ,Credits
3. Instructor
   InstructorID (PK), FirstName, LastName, Email, Department
4. Department
   DepartmentID(PK), DepartmentName, OfficeLocation
5. Enrollment
   EnrollmentID(PK), StudentID(FK), CourseID(FK), EnrollmentDate, Grade 6.
ClassSchedule
   ScheduleID(PK), CourseID(FK), InstructorID(FK), Semester, Time, Location

Relationships:

- A Student can enroll in many Courses (Many-to-Many via Enrollment).
- A Course can have many Students (Many-to-Many via Enrollment).
- A Course is taught by one or more Instructors (Many-to-Many via ClassSchedule).
- An Instructor belongs to one Department (Many-to-One).
- A Department can have many Instructors (One-to-Many).

ER Diagram:


subject_id
student_na
student_id
student_nam
SUBJECT
subject_id
student_name
W ITES
STUDENT
student_id
student_name
MARKS
marks_id
subject_id

PROGRAM

```
      import sqlite3

      #        Connect        s)        conn        =
   sqlite3.connect('student_manement.db')  cursor
   = conn.cursor()

   # Create tables
   cursor.execute('''
   CREATE TABLE IF NOT EXISTS students (     student_id
   INTEGER PRIMARY KEY AUTOINCREMENT,     name
   TEXT NOT NULL
   )
   ''')
   cursor.execute('''
   CREATE TABLE IF NOT EXISTS subjects (    subject_id INTEGER PRIMARY KEY
   AUTOINCREMENT,   name
      TEXT NOT NULL
      )
   ''')
   cursor.execute('''
   CREATE TABLE IF NOT EXISTS marks (    mark_id
      INTEGER PRIMARY KEY AUTOINCREMENT,
      student_id INTEGER,     subject_id INTEGER,     marks
      INTEGER,
         FOREIGN KEY(student_id) REFERENCES students(student_id),
   FOREIGN KEY(subject_id) REFERENCES subjects(subject_id)
      )
   ''')
   # Function to add a student def add_student(name):     cursor.execute("INSERT
   INTO students (name) VALUES (?)", (name,))     conn.commit()
     print(f"Student '{name}' added.")
   #  Function  to  add  a  subject  def
   add_subject(name):
     cursor.execute("INSERT INTO subjects (name) VALUES (?)", (name,))
      conn.commit()
         print(f"Subject '{name}' added.")
```

```
# Function to record marks def record_marks(student_id, subject_id, marks):
cursor.execute("INSERT INTO marks (student_id, subject_id, marks) VALUES (?, ?, ?)",
               (student_id, subject_id, marks))
conn.commit()
   print(f"Marks recorded for Student ID {student_id} in Subject ID {subject_id}: {marks}
marks.")
# Function to display student performance
def show_student_performance(student_id):
cursor.execute('''
   SELECT students.name, subjects.name, marks.marks
   FROM marks
   JOIN students ON marks.student_id = students.student_id
WHERE students.student_id = ?
   ''', (student_id,))     records =
cursor.fetchall()     if not
records:         print("No records
found.")
      return
 print(f"\nPerformance for Student ID {student_id} ({records[0][0]}):")
for subject, marks in [(r[1], r[2]) for r in records]:
print(f"- {subject}: {marks} marks")
# Sample usage if __name__
== "__main__":
   add_student("Alice")
   add_student("Bob")

   add_subject("Math")
add_subject("Science")

   record_marks(1, 1, 85)  # Alice - Math
record_marks(1, 2, 92)  # Alice - Science
record_marks(2, 1, 78)  # Bob - Math

   show_student_performance(1)
show_student_performance(2)

   conn.close()
```

OUTPUT:

```
Student 'Alice' added. Student
'Bob' added.
```

```
Subject 'Math' added.
Subject 'Science' added.
Marks recorded for Student ID 1 in Subject ID 1: 85 marks.
Marks recorded for Student ID 1 in Subject ID 2: 92 marks.
Marks recorded for Student ID 2 in Subject ID 1: 78 marks.
Performance for Student ID 1 (Alice):
- Math: 85 marks
- Science: 92 marks
Performance for Student ID 2 (Bob):
- Math: 78 marks
```

10.Conclusion

The Student/Academic Management System is a vital tool for modern educational institutions,enabling efficient handling of student data, academic records,attendance,and examination results.By automating routine tasks and centralizing information,the system improves administrative efficiency,enhances accuracy,and supports better decision-making. The successful development and implementation of SAMS demonstrate the importance of integrating technology into academic environments to streamline operations, support academic planning, and enhance the educational experience for all stakeholders. With scalability and adaptability at its core, SAMS is well-positioned to evolve alongside institutional needs, making it a valuable asset for modern education systems.

Through centralized data management and role-based access, SAMS ensures that students, faculty, and administrators can interact with the system securely and effectively. The system not only promotes transparency and accountability but also supports informed decision-making through real-time reporting and analytics.

Overall, SAMS serves as a reliable, user-friendly, and efficient solution that improves the quality of academic administration and contributes to the broader goal of digital transformation in education.

11.REFERENCES

- ▶ Moodle – Open-source Learning Management System (LMS).
  URL: https://moodle.org

- ▶ EDUCAUSE – https://www.educause.edu/
  A leading resource on IT in higher education including SIS (Student Information Systems).

- ▶ W3Schools. (2024). *HTML, CSS, JavaScript Tutorials*. Retrieved from: https://www.w3schools.com

- ▶ IEEE Xplore Digital Library. (2023). *Academic Management System Designs and Implementations*. Retrieved from: https://ieeexplore.ieee.org

▶ GitHub. (2024). *Student Management System Projects (Open Source Repositories)*. Retrieved from: https://github.com