Student Management System Project Report

**1. Problem Statement**

Educational institutions often require a centralized **Student Management System (SMS)** to manage student data, course registrations, examination scores, and result generation. Manual handling of such operations leads to inefficiencies, data inconsistencies, and difficulty in scalability.

Hence, a software solution is needed that:

* Allows an admin to register students and courses.
* Assigns courses and scores to students.
* Allows students to view their enrolled courses and results.
* Ensures secure and structured access control (admin vs. student).

**2. Proposed Solution**

## The proposed solution is a Java-based Student Management System that supports:

The implemented Java program is an **object-oriented Student Management System** designed using multiple classes such as Student, Course, Result, Admin, and Main. It fulfills the above requirements using clear modular design and interactions between components.

#### 1. ****Student Class****

* **Attributes:**
  + Unique student ID (auto-incremented)
  + Name and Email
  + List of registered courses
  + Map of exam scores (course → score)
* **Key Methods:**
  + registerCourse(Course course) – adds a course to the student.
  + assignScore(Course course, int score) – stores exam score.
  + viewCourses() – displays registered courses.
  + viewResults() – displays scores and grades using Result.calculateGrade().

#### 2. ****Course Class****

* **Attributes:**
  + Course ID (String)
  + Course Name (String)
  + Credits (int)
* **Purpose:** Represents an academic course.

#### 3. ****Result Class****

* **Static Method:** calculateGrade(int score)
* **Logic:** Converts numeric score into a letter grade:
  + 90+ → A
  + 80-89 → B
  + 70-79 → C
  + 60-69 → D
  + Below 60 → F

#### 4. ****Admin Class****

* Handles **authentication** using a predefined username (admin) and password (1234).
* Maintains lists of all students and courses.
* **Functionality:**
  + Registering students
  + Adding courses
  + Assigning courses and scores
  + Viewing all students

#### 5. ****Main Class (UI/Interface Layer)****

* Provides a **console-based interface** with a menu-driven approach.
* Divides access into:
  + **Admin Panel** – with full CRUD privileges.
  + **Student Panel** – limited to viewing registered courses and results.

**3. System Architecture**

## The system follows a modular object-oriented architecture with the following core components:

* - Admin Module: Handles all administrative functions including authentication, student registration, course creation, and assigning courses/scores.
* - Student Module: Allows students to view their registered courses and exam results.

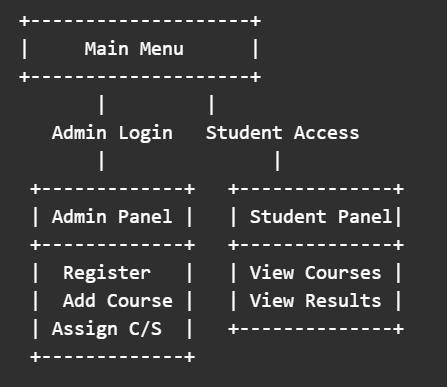
## - Data Classes:

* - Student: Stores student details, courses registered, and exam scores.
* - Course: Represents a course with ID, name, and credit details.
* - Result: Utility class to calculate grade based on score.

**4. Class Descriptions**

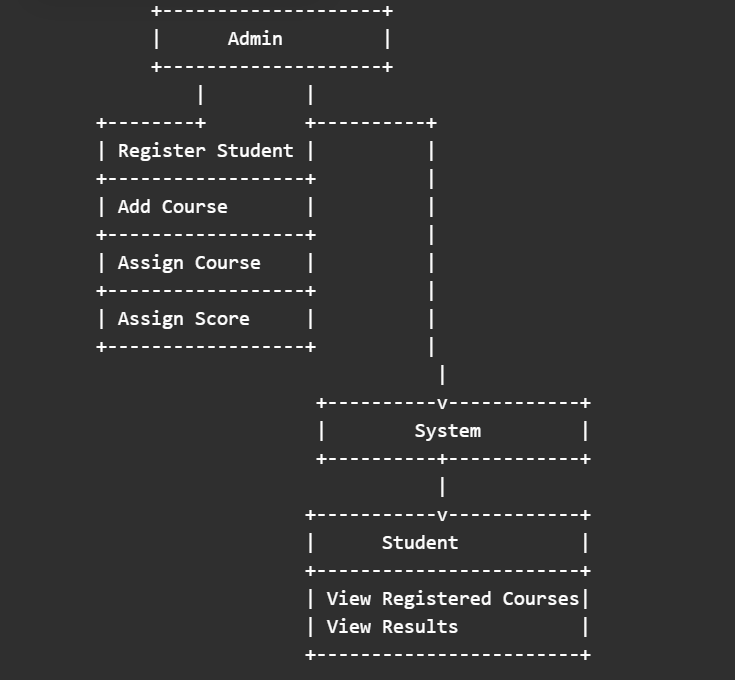
* - Student.java
* - Attributes: studentId, name, email, registeredCourses, examScores
* - Methods: registerCourse(), assignScore(), viewCourses(), viewResults()
* - Course.java
* - Attributes: courseId, courseName, credits
* - Methods: Getters for attributes
* - Result.java
* - Static Method: calculateGrade(score) returns grade A-F based on score
* - Admin.java
* - Attributes: students, courses, and predefined admin credentials
* - Methods: login(), registerStudent(), addCourse(), assignCourseToStudent(), assignScore(), and accessors
* - Main.java
* - Provides console-based UI for Admin and Student functionalities using menu-driven interactions

**5. Flow Diagram (Logical Workflow)**



**6. UML Diagrams**

## Use Case Diagram:



## Class Diagram:



**7. Key Features**

1. **Object-Oriented Design**
   * Uses classes like Student, Course, Admin, and Result for modularity and clarity.
2. **Student Auto-ID Generation**
   * Each student is automatically assigned a unique ID starting from 1000.
3. **Course Management**
   * Admin can add courses with ID, name, and credits.
4. **Secure Admin Login**
   * Simple username-password authentication for admin tasks.
5. **Course & Score Assignment**
   * Admin can assign courses and exam scores to students.
6. **Automatic Grade Calculation**
   * Converts scores into grades (A-F) based on standard criteria.
7. **Student Panel**
   * Students can view their courses and results using their ID.
8. **Admin Panel**
   * Full control to register students, add courses, and manage data.
9. **Console-Based Interface**
   * Menu-driven UI for interaction via the command line.
10. **Data Handling with Lists and Maps**

* Efficient in-memory storage using ArrayList and HashMap.

**8. Future Enhancements**

* - GUI-based interface using JavaFX or Swing
* - Persistent storage using databases
* - Email notifications for scores and course registration
* - Role-based access control and logging

**9. Conclusion**

The Student Management System developed in Java demonstrates a functional and modular approach to managing academic data within an educational institution. By applying core object-oriented programming principles, the system effectively handles student registration, course management, score assignment, and grade generation.

The admin interface provides full control over data entry and updates, while the student interface allows limited access for viewing courses and results. The use of data structures like ArrayList and HashMap ensures efficient data handling during runtime.

Although currently console-based and memory-resident, the system forms a solid foundation for future enhancements such as database integration, secure login systems, and graphical user interfaces. Overall, the project achieves its goal of offering a simple, structured, and extendable solution for academic administration.