***Name- Munaif Hussain***

***Sys.id*-2022530294(B. TECH-IT)**

**STUDENT MANAGEMENT SYSTEM**

**Problem Statement**

Educational institutions often struggle with the complexities of managing student information, course registrations, attendance records, examination schedules, and academic results through traditional, manual processes. These outdated systems are prone to human error, time-consuming administrative tasks, data duplication, and limited accessibility. Furthermore, the lack of real-time insights and centralized data management hampers decision-making, student engagement, and overall academic efficiency. Therefore, there is a critical need for an automated, scalable, and user-friendly **Student Management System** that can streamline academic workflows, enhance data accuracy, and provide real-time access to essential information for students, faculty, and administrators.

**Objective**

The Student Management System aims to develop a robust Java-based console application that automates academic administration by enabling efficient student registration, course enrollment, and exam management while providing administrators with comprehensive tools for content management and delivering students seamless access to learning materials and assessments through an integrated platform.

**Technologies Used**

* Java (OOP, Collections, Abstraction)
* Java Console Input/Output
* Data Structures: Array List, HashMap
* Package Structure: Student\_Management\_System

**User Roles**

**Admin:**

• Create, update, and delete courses  
• Add training materials (PDF names, web links)  
• Add MCQ exam questions to courses  
• View all available courses  
• Protected by hardcoded credentials: admin@example.com / admin123

**Student:**

• Register with name, email, password  
• View and enrol in available courses  
• Access course-specific materials  
• Attempt timed MCQ exams (10 questions/course)  
• View results with percentage and pass/fail status

**System Architecture / Class Design**

**Core Classes:**

1. **User (Abstract)**
   * Base for all users
   * Stores: name, email, password
   * Requires: showMenu() implementation
2. **Admin (Extends User)**
   * Manages courses (create/delete)
   * Adds materials & exam questions
3. **Student (Extends User)**
   * Enrolls in courses
   * Takes exams, views results
4. **Course**
   * Contains: title, description, materials
   * Has one Exam
5. **Exam**
   * Holds list of Questions
   * Manages test-taking
6. **Question**
   * Stores: question text, 3 options, correct answer
7. **Result**
   * Tracks: score, percentage, pass/fail

**Program Workflow**

**1. System Initialization**

* **Preloaded Data:**
  + Admin Account: admin@example.com / admin123
  + Sample Courses:
    - * Java Basics (2 materials, 10 MCQs)
      * Python Fundamentals (2 materials, 10 MCQs)
      * Web Development (2 materials, 10 MCQs)

**2. Role-Based Execution**

**Admin Flow:**

* Login with credentials
* Manage courses (add/delete)
* Upload materials (PDFs/links)
* Configure exam questions

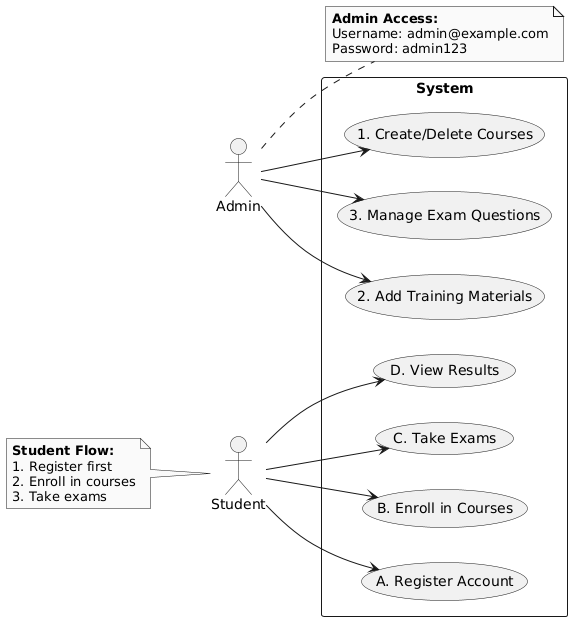
**Student Flow:**

* Register (name, email, password)
* Browse & enrol in courses
* Access materials → Take exams
* View instant results

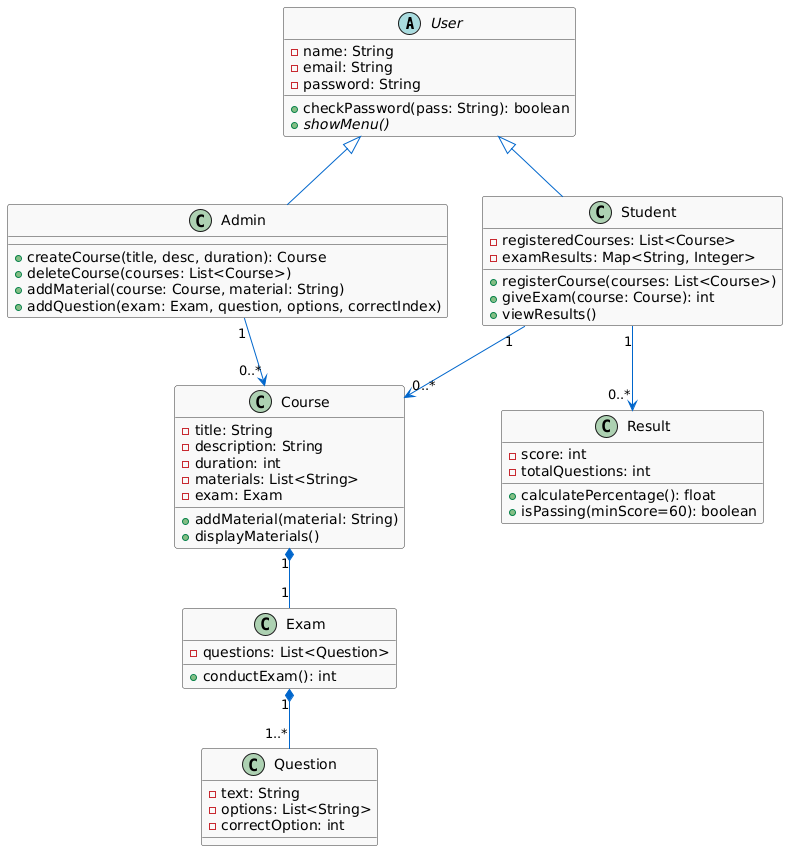
**How the System Works Together**

1. **Admin creates courses** with titles, descriptions, and duration.
2. **Admin uploads** training materials and exam questions.
3. **Student enrols** in available courses.
4. **Student takes** timed MCQ exams.
5. **System auto-grades** and displays results instantly.
6. **Results save** to student profiles for future access.

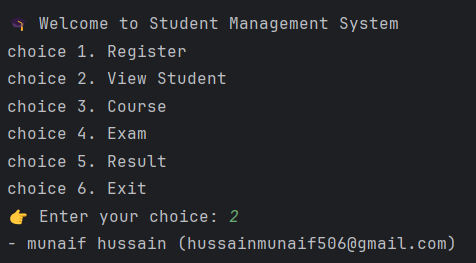
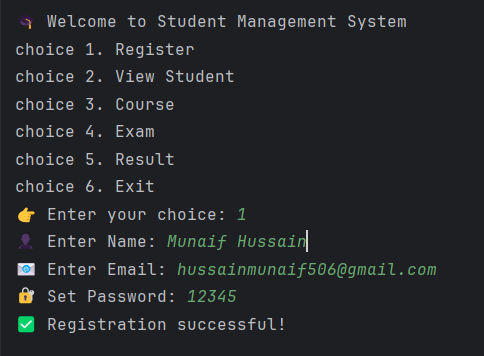
**Use Case Diagram**

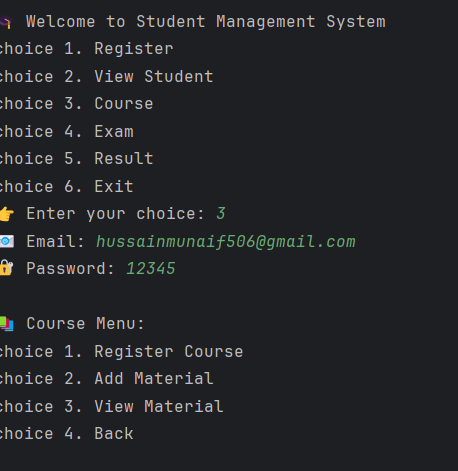
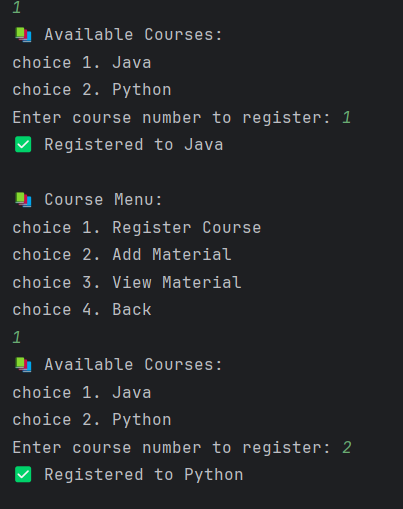


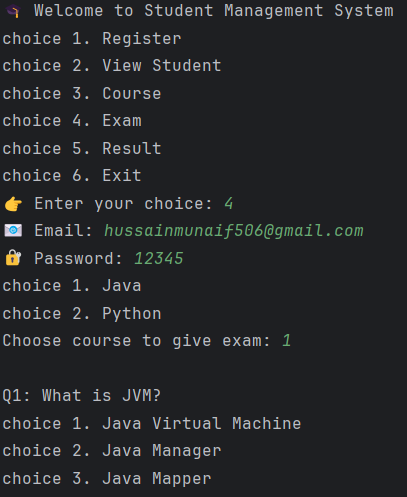
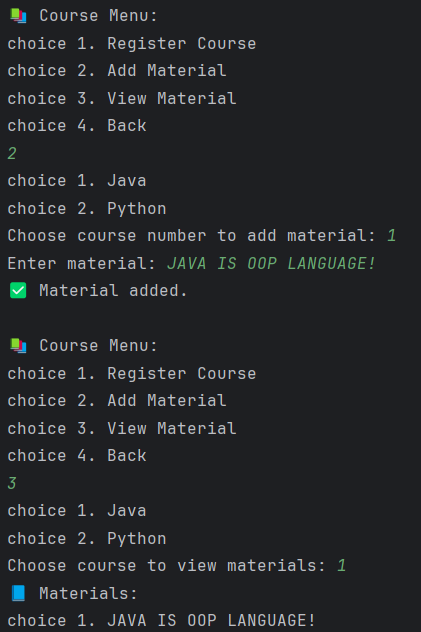
**Class Diagram**

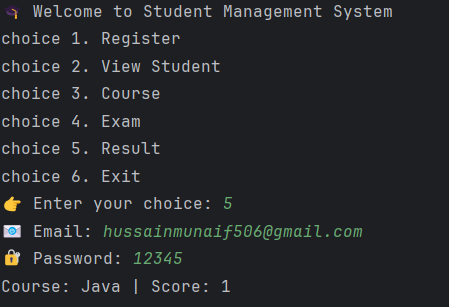
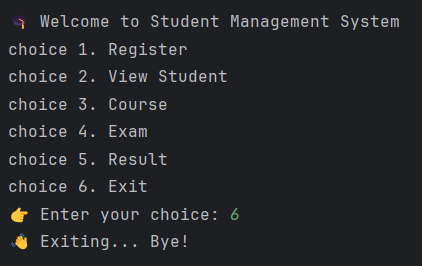


**Screenshots**



****

****

**** ****

**Conclusion**

This Java-based Student Management System effectively applies Object-Oriented Programming principles to streamline academic operations. Its modular structure ensures easy maintenance and scalability, while features like automated course management, exam handling, and result tracking reduce manual workload and errors. Overall, it offers a solid, efficient, and user-friendly solution for managing academic tasks, with great potential for future enhancements like database support and GUI integration.