Course: Java Programming

Submitted by: Sai Krishna Reddy

Submitted to: VITyarthi

Institution : VIT

Year: 2025

📓 PROJECT REPORT – Student Course Registration System

1. Introduction

The Student Course Registration System is a Java-based console application developed to simplify the process of course registration for students. In traditional academic environments, course registration often involves manual form filling or coordination with administrators, which makes the process slow and error-prone.
This project aims to provide an interactive, easy-to-use system that allows students to view available courses, enroll in them, drop them, and maintain their own academic course list. It also demonstrates the principles of object-oriented programming, modular design, and basic software engineering practices.

---

2. Problem Statement

In many institutions, course registration is still handled manually or through unorganized digital methods. Students often experience difficulties such as:

- Inability to track course availability
- Errors in manual entries
- Lack of immediate confirmation of enrollment
- Difficulty in modifying their selections

To tackle these issues, a structured software solution is required. The proposed system automates the course registration workflow, ensuring faster processing, minimal errors, and improved user experience.

---

3. Objectives

The main objectives of this software project are:

- To design a simple and user-friendly registration system
- To model the course registration workflow using Java OOP concepts
- To implement separate modules for students, courses, and registration services
- To provide a menu-driven interface that guides the user through each operation
- To create a lightweight system that can be extended or upgraded easily in the future

---

4. Functional Requirements

The system includes the following functional components:

4.1 Student Management

- Register a new student with ID, name, and password
- Authenticate existing students through login
- Provide access to their registration dashboard

4.2 Course Management

- Add new courses (admin or system initialization)
- Display all available courses
- Search and retrieve course information based on ID

4.3 Registration Module

- Allow students to enroll in selected courses
- Prevent duplicate course registration
- Display a student’s registered courses
- Allow students to drop previously enrolled courses

These modules together enable smooth and complete course registration functionality.

---

5. Non-Functional Requirements

5.1 Usability

The interface is simple, text-based, and easy to navigate using numeric menu options.

5.2 Maintainability

The project is developed using modular Java classes such as `StudentService`, `CourseService`, and `RegistrationService`, making it easy to update or add features later.

5.3 Reliability

Validation checks are implemented to avoid issues like duplicate enrollments or invalid course IDs.

5.4 Performance

Java Collections such as ArrayList and HashMap ensure quick data retrieval and modification, even when multiple students or courses are added.

5.5 Scalability

The system is designed in such a way that it can later be upgraded to a GUI application or linked to a real database without large design changes.

---

6. System Architecture

The system follows a multi-module architecture where each module performs a specific task:

- **Main.java** – Starts the application and initializes sample data
- **Menu.java** – Acts as the controller for user interaction
- **Student.java** – Represents a student entity
- **Course.java** – Represents a course entity
- **StudentService.java** – Handles student registration and authentication
- **CourseService.java** – Manages course list and retrieval
- **RegistrationService.java** – Manages enrollment, dropping courses, and viewing lists

This architecture separates business logic from data representation, which is ideal for maintainable code.

---

7. UML & Design Diagrams

You can add diagrams in your final report. Here's what each diagram represents:

7.1 Use Case Diagram

Shows actions like Login, View Courses, Enroll, Drop Course and the main actor: Student.

7.2 Class Diagram

Includes classes such as Student, Course, StudentService, CourseService, RegistrationService, and Menu.
Shows their attributes and methods.

7.3 Sequence Diagram

Demonstrates the flow of operations when a student logs in and interacts with the registration functions.

You can create these diagrams using tools like draw.io, Lucidchart, or StarUML.

---

8. Implementation Details

The system is implemented using object-oriented concepts such as:

- **Encapsulation** (private attributes with getters/setters)
- **Abstraction** (services hiding logic from main class)
- **Modularity** (separate files for each function)

The application uses:

- `HashMap` for storing students and registrations
- `ArrayList` for storing courses
- Simple validation and looping mechanisms for user input handling

The program runs entirely on the console, offering a clean and understandable execution flow.

---

9. Results and Screenshots

You can add your own screenshots such as:

- Login screen
- Course list
- Enrollment confirmation
- Registered courses display

These will improve your report quality.

---

10. Testing Approach

The system was tested manually using black-box testing. Different test cases were performed:

- Valid and invalid login attempts
- Trying to enroll in the same course twice
- Dropping a non-registered course
- Viewing courses with multiple added entries
- Verifying system behavior with no registrations

All core functionalities worked correctly during testing.

---

11. Challenges Faced

- Designing a clean structure that separates logic clearly
- Ensuring proper validation for student login and course enrollment
- Making the menu flexible without making the code messy
- Managing multiple lists and maps efficiently

---

12. Learnings & Key Takeaways

- Practical understanding of Java OOP
- Experience in building a modular console-based project
- Improved debugging and testing skills
- Better understanding of system workflows and architecture

---

13. Future Enhancements

In future versions, the system can be extended with:

- Database connectivity using MySQL
- A graphical interface using JavaFX or Swing
- Admin login with features like editing or deleting courses
- Course availability limits to prevent over-enrollment
- Notification and email system for students

---

14. References

- Oracle Java Documentation
- Official Java Tutorials
- OOP Principles – Textbook References
- Java Collections Framework Guide