# VITYARTHI PROJECT REPORT


## Submitted by:

NAME: ROHIT KUMAR
REG NO: 24BCE10466


## Course Name:

PROGRAMMING IN JAVA


## Institution:

VELLORE INSTITUTE OF TECHNOLOGY BHOPAL

## Guide:

DR. ANAND MOTWANI

## FACULTY:

KOMARASAMY.G


## Date of Submission:

24-09-25

## Abstract

The Campus Course Records Manager is a Java-based desktop application designed to manage and maintain student and course records efficiently. By providing a simple console interface, it automates the tasks of adding, updating, deleting, and viewing student and course information. This system reduces manual errors and provides easy access to academic data with strong modularity and extensibility using Object-Oriented Programming concepts.

---

## Introduction

Managing student and course data manually is inefficient, error-prone, and labor-intensive, especially in educational institutions. This project offers an automated solution to store, update, and retrieve information about students and courses within a campus through a console-based system written in Java.

Key objectives of this project include:

- To design a modular and extensible system using Java OOP principles.
- To provide CRUD operations for student and course data.
- To create a menu-driven console user interface for ease of operation.
- To ensure data integrity and easy maintenance.

---

## System Requirements

## Software Requirements

- Java SE Development Kit (JDK) version 11 or later
- Eclipse IDE or any Java compatible IDE for development and testing

## Hardware Requirements

- Standard computer running Windows, Linux, or macOS
- Minimum 4 GB RAM
- Approximately 200 MB disk space for project files

## System Design

The system is designed following object-oriented design principles to ensure modularity, ease of maintenance, and future extensibility.

## Architecture

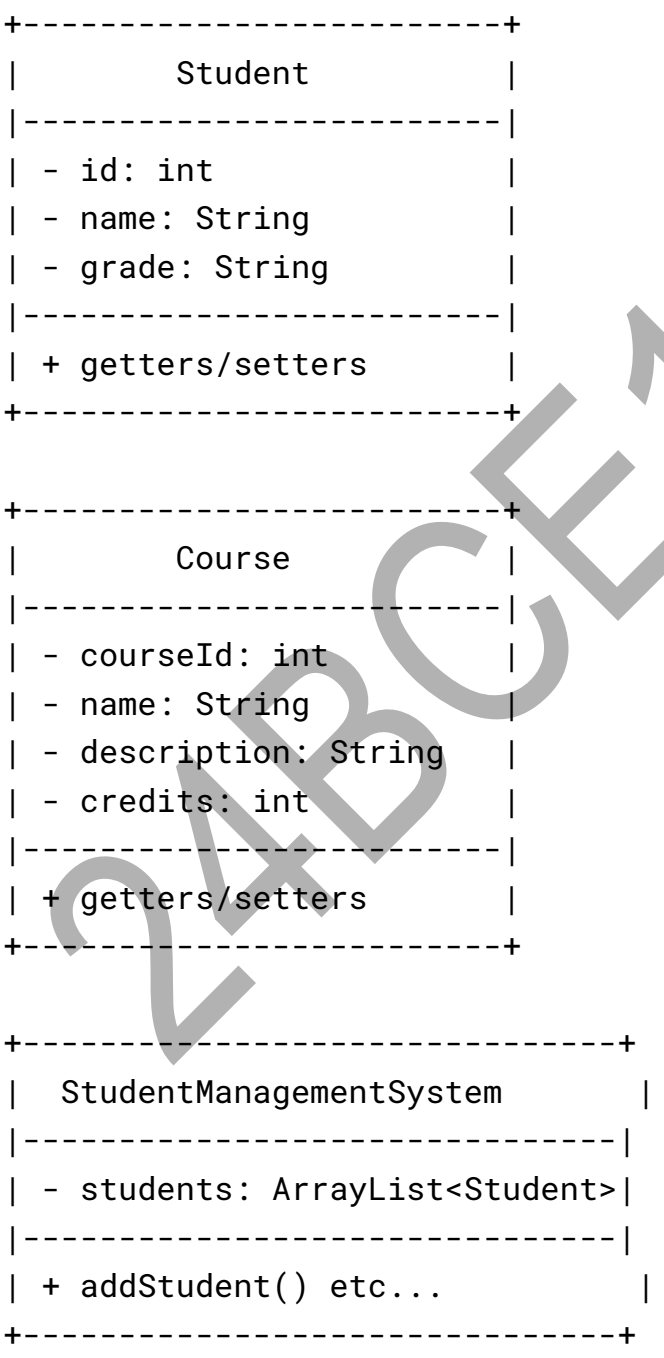
The project is divided into three logical modules packaged as follows:

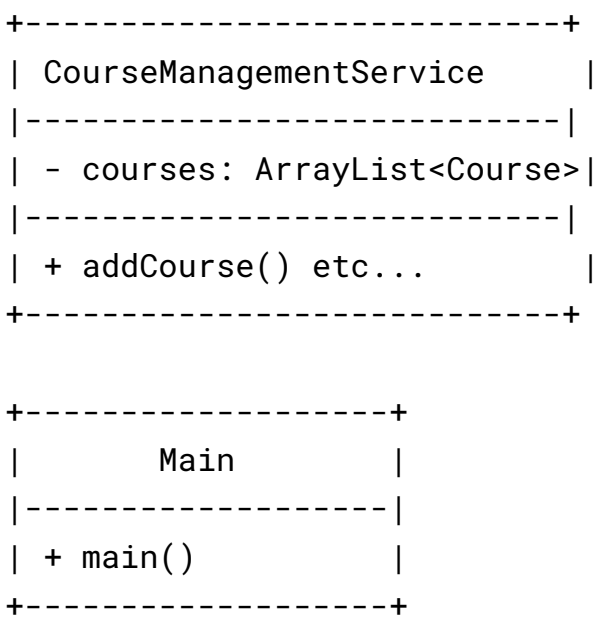
- edu.ccrm.domain: This package defines the core domain classes `Student` and `Course`. They encapsulate the data fields and provide appropriate getters/setters.
- edu.ccrm.service: This layer contains `StudentManagementSystem` and `CourseManagementService` classes which implement CRUD operations using Java collections like ArrayList to manage dynamic data.
- edu.ccrm.cli: This package contains the `Main` class which provides a console-based menu interface that interacts with the service classes to perform required operations.

## Class Diagrams

(Diagram Placeholder)

Below is a simplified UML class diagram showing classes and their relationships:




CourseManagementService
- courses: ArrayList<Course>
+ addCourse() etc...
Main
+ main()

## Implementation

The project is implemented in Java using standard Java SE libraries. The core logic is encapsulated in service classes that use `ArrayList` to store data dynamically. Each entity like `Student` and `Course` is modeled as a separate class with data encapsulation.

The main features implemented are:

- Student Management:
    - Add a new student record with ID, name, and grade.
    - View all student records.
    - Update existing student details.
    - Delete a student record by ID.
- Course Management:
    - Add a new course record with course ID, name, description, and credits.
    - View all courses.
    - Update details for existing courses.
    - Delete a course by ID.

All CRUD operations are accessible through a menu-driven console interface implemented in the `Main` class in the `edu.ccrm.cli` package. Users can repeatedly select options and provide data inputs for streamlined interaction.

## Code Snippet Examples

Student.java

```java
package edu.ccrm.domain;

public class Student {
    private int id;
    private String name;
    private String grade;

    public Student(int id, String name, String grade) {
        this.id = id;
        this.name = name;
        this.grade = grade;
    }

    // Getters and Setters
}
```

StudentManagementSystem.java (Partial)

```java
package edu.ccrm.service;

import java.util.ArrayList;
import edu.ccrm.domain.Student;

public class StudentManagementSystem {
    private ArrayList<Student> students = new ArrayList<>();

    public void addStudent(Student student) {
        students.add(student);
    }

    // Other CRUD methods here
}
```

## Testing

The system was tested extensively to ensure functionality and robustness. Test cases focused on all CRUD operations for both students and courses. Common scenarios tested include:

- Adding new students and courses with valid data.
- Attempting to update or delete records that do or do not exist.
- Viewing records to confirm accurate additions or modifications.
- Handling invalid inputs gracefully.

A sample session involves adding students, updating grades, viewing all records, and deleting entries successfully using the console menu.

Screenshots of test cases and outputs are included in the appendix (or can be added here).

---

## Conclusion and Future Scope

The Campus Course Records Manager project successfully demonstrates core Java programming concepts, including data encapsulation, collections, user input handling, and modular design.

While functional as a console application, the project can be enhanced by:

- Implementing a graphical user interface (GUI) for better user experience.
- Integrating persistent storage using file handling or a database like MySQL.
- Adding authentication and role-based access features.
- Allowing export/import of data as CSV or Excel files.
- Deploying as a web application for remote access.



## References


- Oracle Java Documentation: https://docs.oracle.com/javase/
- Eclipse IDE User Guide: https://www.eclipse.org/documentation/
- Java Tutorials by Oracle
- Various online Java programming tutorials

# THANK YOU