Campus Course & Records Manager (CCRM) – Project Report

1. Introduction

The Campus Course & Records Manager (CCRM) is a Java-based command-line system designed for educational institutions to manage students, courses, enrollments, grades, reports, and backups.
The system demonstrates object-oriented programming, layered architecture, file handling, builder patterns, custom exceptions, and Java collection frameworks.

---

2. Problem Statement

Educational institutions that manage academic records manually face several challenges:

- Inconsistent and missing data
- Difficulty in tracking students and courses
- Errors in enrollments and grading
- No automated reporting
- No backup and recovery
- Slow administrative work

A centralized, menu-driven, and robust Java system is needed to:

- Maintain student and course data
- Handle enrollments and grades
- Import/export academic records
- Provide backups and reports

CCRM solves all these problems.

---

3. Functional Requirements

Student Management

- Add student
- Update details
- Deactivate student
- List all students
- View student profile
- Generate transcript

Course Management

- Add course (Builder Pattern)
- Update course
- Deactivate course
- List courses
- Filter by instructor/department/semester

Enrollment & Grades

- Enroll student
- Unenroll student
- Record marks
- Prevent duplicate enrollment

- Enforce max credit limit

Import/Export

- Import students via CSV
- Import courses via CSV
- Export all data

Backup

- Create backup folder
- Save exported files
- Show backup size

Reports

- Top 5 students by GPA
- GPA distribution (High / Mid / Low)

---

4. Non-Functional Requirements

1. **Performance:** Fast in-memory operations
2. **Usability:** Easy menu-driven interface
3. **Maintainability:** Layered design for easy updates
4. **Reliability:** Custom exceptions and validations
5. **Scalability:** Can be upgraded to database-based system in the future
6. **Portability:** Works on any OS with Java installed

---

5. System Architecture

The system uses these layers:

- **CLI Layer:** Main.java
- **Service Layer:** StudentService, CourseService, EnrollmentService, etc.
- **Domain Layer:** Student, Course, Enrollment, Semester
- **Value Layer:** CourseCode
- **Exceptions:** Custom domain-based exception classes
- **IO Layer:** Import/Export service
- **Backup Layer:** File backup operations
- **Config Layer:** Central AppConfig with in-memory storage

Architecture Diagram

```
flowchart LR
    UI[CLI - Main.java] --> SVC[Service Layer]

    SVC --> STU[StudentService]
    SVC --> CRS[CourseService]
    SVC --> ENR[EnrollmentService]
    SVC --> IO[ImportExportService]
    SVC --> BAK[BackupService]

    STU --> DOM[Domain Models]
    CRS --> DOM
    ENR --> DOM

    IO --> FILES[(CSV Files)]
    BAK --> ZIP[(Backups)]

    DOM --> CFG[AppConfig]
```

---

6. Use Case Diagram

```
usecaseDiagram
actor Admin
actor Faculty

Admin --> (Manage Students)
Admin --> (Manage Courses)
Admin --> (View Reports)
Admin --> (Backup Data)

Faculty --> (View Courses)
Faculty --> (Record Marks)
Faculty --> (View Student Performance)
```

---

7. Workflow Diagram

```
flowchart TD
    A[Start Application] --> B[Show Main Menu]
    B --> C[Manage Students]
    B --> D[Manage Courses]
    B --> E[Enrollment & Grades]
    B --> F[Import/Export]
    B --> G[Backup]
    B --> H[Reports]
    B --> Z[Exit]

    C --> C1[Add/Update/List/Deactivate Student] --> B
    D --> D1[Add/Update/List/Filter/Deactivate Course] --> B
    E --> E1[Enroll/Unenroll/Record Marks] --> B
    F --> F1[Import CSV / Export All] --> B
    G --> G1[Create Backup & Show Size] --> B
    H --> H1[Top Students & GPA Distribution] --> B
```

---

8. Class Diagram

```
classDiagram

    class Student {
        - String regNo
        - String name
        - String email
        - boolean active
        + profile()
        + gpa()
    }

    class Course {
        - CourseCode code
        - String title
        - int credits
        - String instructor
        - Semester semester
        - String department
    }

    class Enrollment {
        - Student student
        - Course course
        - Semester semester
        - double marks
    }

    class StudentService
    class CourseService
    class EnrollmentService
    class ImportExportService
    class BackupService
    class AppConfig

    StudentService --> Student
    CourseService --> Course
    EnrollmentService --> Enrollment
```

```
    AppConfig --> Student
    AppConfig --> Course
```

---

9. Sequence Diagram

```
sequenceDiagram
    participant User
    participant Main
    participant StudentService
    participant CourseService
    participant EnrollmentService

    User->>Main: Select Enrollment Option
    Main->>User: Request RegNo
    User->>Main: Enter RegNo
    Main->>StudentService: find(regNo)
    StudentService-->>Main: Student

    Main->>User: Ask Course Code
    User->>Main: Input Code
    Main->>CourseService: find(courseCode)
    CourseService-->>Main: Course

    Main->>User: Ask Semester & Marks
    User->>Main: Input

    Main->>EnrollmentService: recordMarks()
    EnrollmentService-->>Main: Success/Error

    Main-->>User: Display result
```

---

10. Implementation Details

- **Language:** Java 17

- **Architecture:** Layered
- **Key Concepts:**
    - Classes & Objects
    - OOP Principles
    - Enums
    - Builder Pattern
    - Exception Handling
    - Collections + Streams
    - File I/O
- **Data Flow:**
  CLI → Service → Domain → Export/Backup

---

11. Screenshots

(Add your CLI screenshots here)

---

12. Testing Approach

- Unit-based manual testing
- Boundary testing for:
    - Duplicate enrollment
    - Invalid student/course
    - Over-credit conditions

- Import/Export file tests
- Backup folder creation tests
- GPA calculation validation

---

13. Challenges Faced

- Correctly designing layered architecture
- Handling exception flow
- Managing relationships between students, courses, enrollments
- Building transcript and GPA logic

---

14. Learnings

- Clean OOP architecture
- Practical Java service-based development
- Handling file processing and backups
- Designing multi-module CLI applications
- Writing maintainable Java code

---

15. Future Enhancements

- Full graphical UI or web application
- Database integration

- Role-based access (admin/faculty)
- Attendance system
- Automatic email-based reporting