VIT BHOPAL

CAMPUS COURSE & RECORDS MANAGER (CCRM)

COURSE: PROGRAMMING IN JAVA
COURSE CODE: CSE2006
CLASS NO.: BL2025260400011
FACULTY: CHOUR SINGH RAJPOOT
SLOT: C11+C12+C13+F11+F12

ANSHUL KANODIA
REGISTRATION NO.: 24BSA10254

Campus Course & Records Manager (CCRM)

1. Introduction

The education sector relies heavily on data accuracy. The **Campus Course & Records Manager (CCRM)** is a Java-based application developed to digitize the core academic processes of a university. By leveraging Object-Oriented Programming (OOP) principles, the project provides a scalable and reliable platform for students to manage their enrolments and for faculty to handle grading and course management.

2. Problem Statement

Traditional methods of managing university records are prone to human error. Physical files can be lost, and spreadsheets do not enforce business rules (like course capacity limits). The lack of a centralized system causes delays in grade reporting and confusion regarding course availability. This project aims to eliminate these inefficiencies by providing a unified digital interface.

3. Functional Requirements

- **User Authentication:** Users must be able to register as either a Student or Professor and log in securely.
- **Course Management:** Professors must be able to create courses with defined credits and student capacity.
- **Enrolment System:** Students must be able to view courses and enrol. The system must validate that the course is not full.
- **Grading:** Professors must be able to assign grades to enrolled students.
- **Reporting:** The system must generate a transcript for students showing their GPA.
- **Data Persistence:** The system must save all data to the disk automatically.

4. Non-Functional Requirements

- **Security:** Access control prevents unauthorized users from accessing professor-specific menus.
- **Usability:** The command-line interface uses clear prompts and formatted tables for readability.
- **Reliability:** Data is serialized and saved after every state change (enrollment, grading), ensuring no data loss on crash.
- **Maintainability:** The code uses a modular structure (Model-View-Controller pattern), making it easy to debug and extend.

5. System Architecture

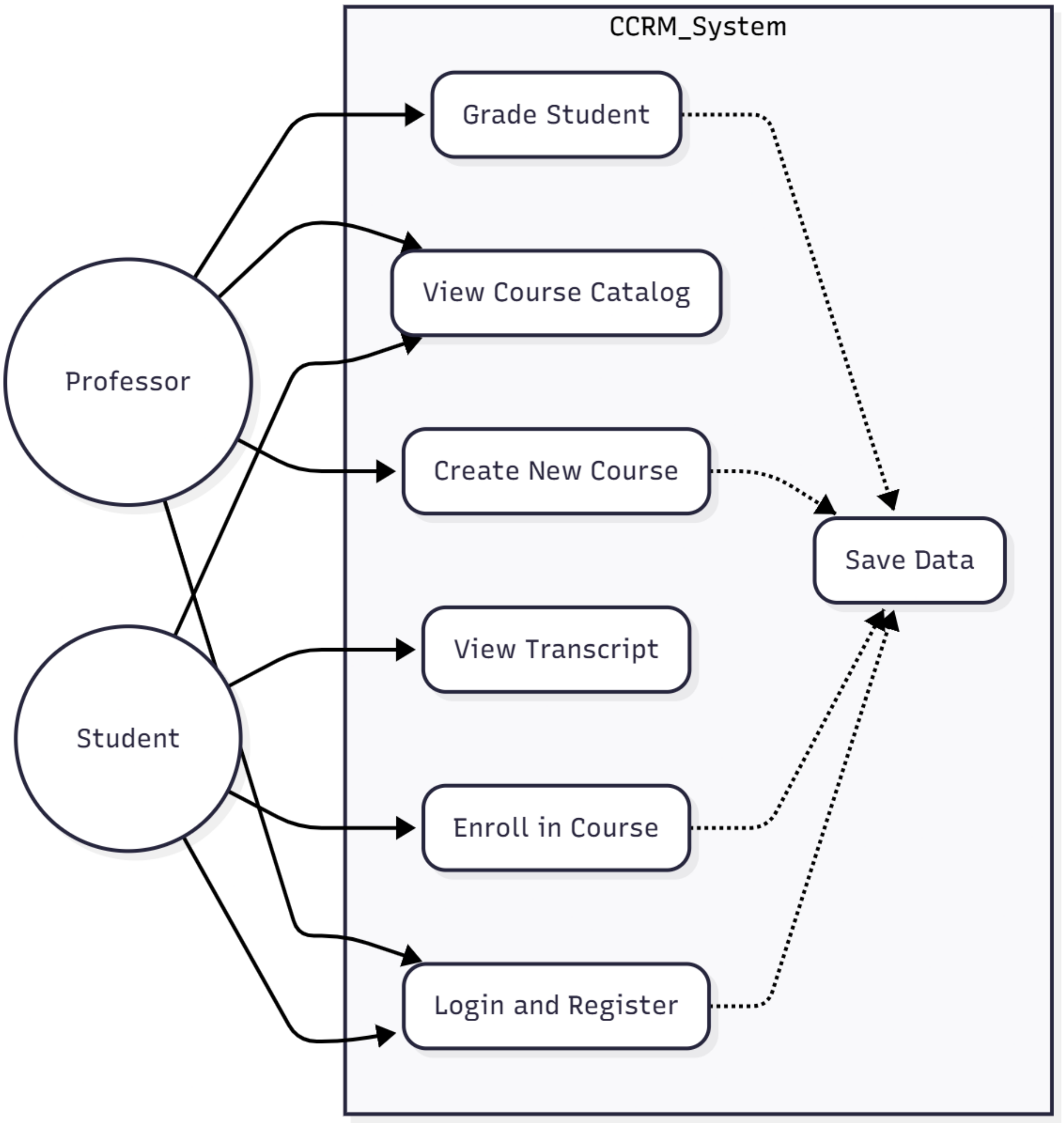
The system is designed with a layered architecture:

1. **Presentation Layer (Main.java):** Handles user input and displays menus.
2. **Logic Layer (RecordManager.java):** Contains the core business logic (finding users, validating capacity).
3. **Data Layer (FileHandler.java):** Manages reading and writing objects to the ccrm_data.ser file.

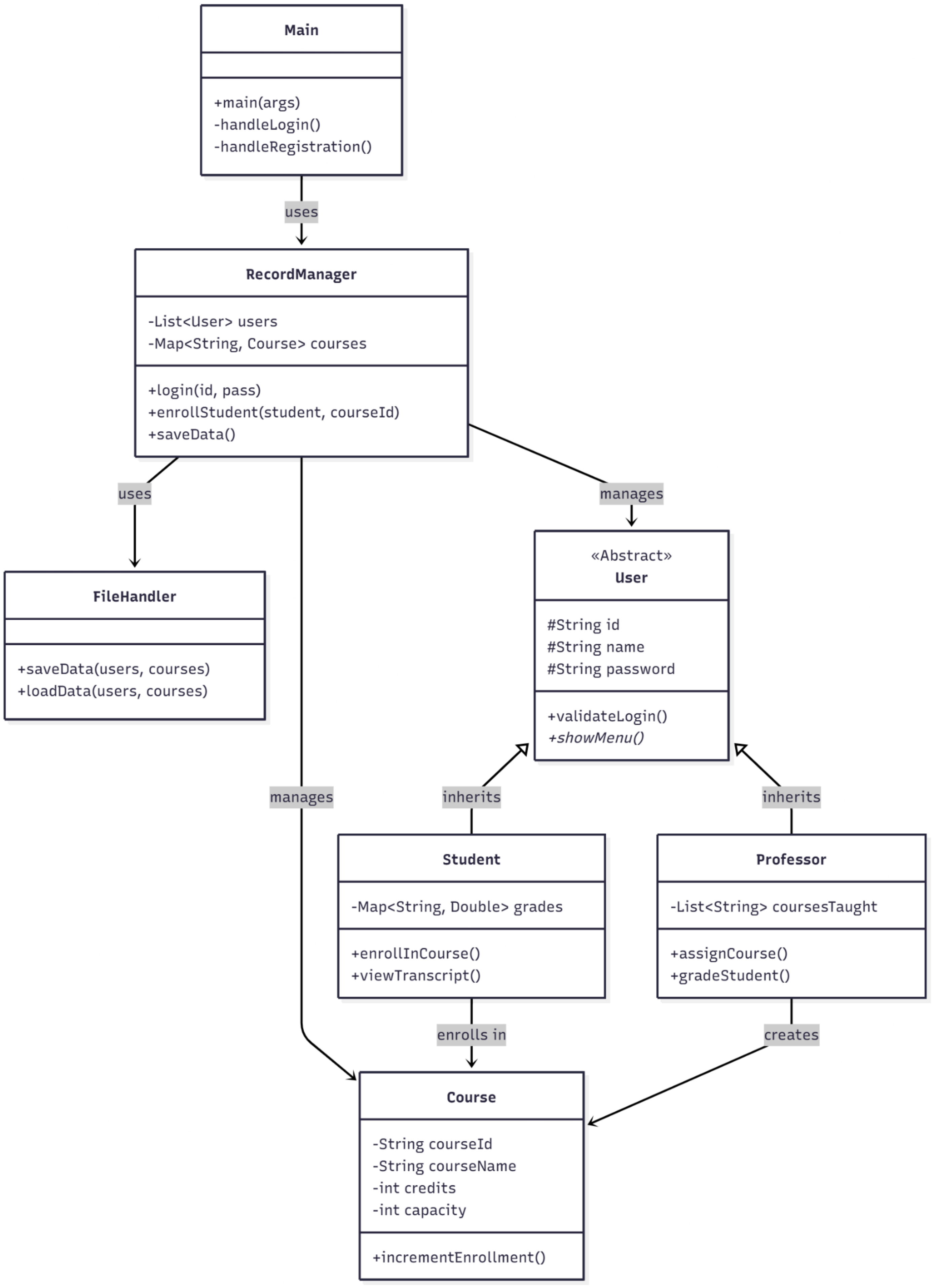
$$\boldsymbol{Flow}: User\ Input \longrightarrow Main\ Menu \longrightarrow RecordManager \longrightarrow FileHandler \longrightarrow Disk.$$

6. Design Diagrams

6.1 Use Case Diagram



6.2 Class Diagram



7. Design Decisions & Rationale

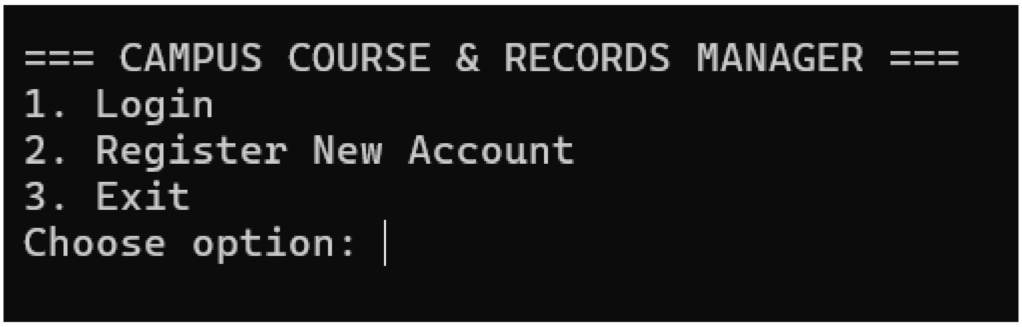
- **Why Java Serialization?** For a single-user desktop application, setting up a SQL database requires extra configuration. Java Serialization (Serializable interface) allows us to save complex object graphs (Users linked to Grades linked to Courses) directly to a file with minimal code.
- **Why Abstract Class User?** Both Student and Professor share attributes like ID, Name, and Password. Using an abstract parent class reduces code duplication and allows us to store all users in a single List<User>.
- **Why Custom Exceptions?** We implemented CourseFullException to cleanly separate the "happy path" logic from error conditions, improving code readability.

8. Implementation Details

The project consists of 10 Java classes. Key components include:

- **RecordManager.java:** The "brain" of the application. It holds the users list and courses map in memory and syncs them to the file system.
- **Student.java:** Uses a HashMap<String, Double> to store grades. The Key is the Course ID, and the Value is the grade. A value of -1.0 represents an ungraded course.
- **TranscriptGenerator.java:** A utility class that iterates through a student's grades and calculates the GPA using the formula: Sum(Grade * Credits) / Sum(Credits).

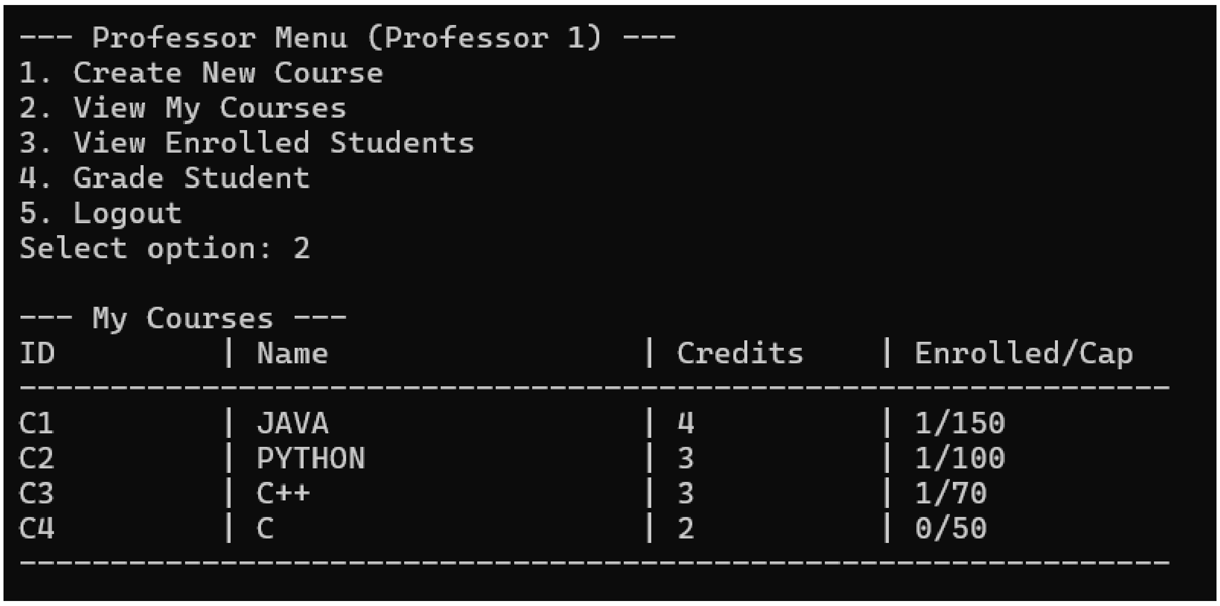
9. Screenshots / Results



```
=== CAMPUS COURSE & RECORDS MANAGER ===
1. Login
2. Register New Account
3. Exit
Choose option:
```

The Login/Registration Menu

The initial screen allowing users to sign in or sign up.

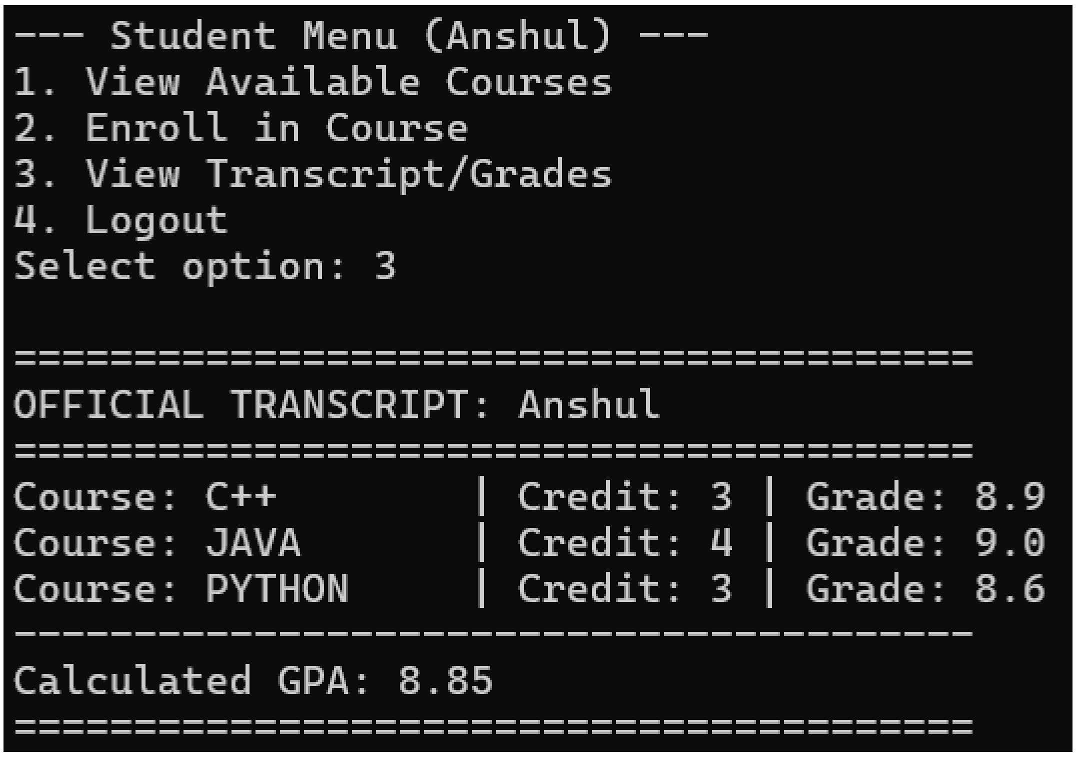


```
--- Professor Menu (Professor 1) ---
1. Create New Course
2. View My Courses
3. View Enrolled Students
4. Grade Student
5. Logout
Select option: 2

--- My Courses ---
ID         | Name                   | Credits     | Enrolled/Cap
---------------------------------------------------------------
C1         | JAVA                   | 4           | 1/150
C2         | PYTHON                 | 3           | 1/100
C3         | C++                    | 3           | 1/70
C4         | C                      | 2           | 0/50
---------------------------------------------------------------
```

The Professor's "View My Courses" Table

Description: The formatted table showing course names and enrolment counts.



```
--- Student Menu (Anshul) ---
1. View Available Courses
2. Enroll in Course
3. View Transcript/Grades
4. Logout
Select option: 3

========================================
OFFICIAL TRANSCRIPT: Anshul
========================================
Course: C++         | Credit: 3 | Grade: 8.9
Course: JAVA        | Credit: 4 | Grade: 9.0
Course: PYTHON      | Credit: 3 | Grade: 8.6
----------------------------------------
Calculated GPA: 8.85
========================================
```

A Student's Transcript/GPA

Description: The generated report showing grades and calculated GPA.

10. Testing Approach

We performed **Manual Black-Box Testing**:

1. **Positive Testing:** Registered a valid user, enrolled in an open course, and verified the grade was saved.
2. **Negative Testing:**
    - Tried to enroll in a full course -> Verified CourseFullException message.
    - Entered "ABC" for course credits -> Verified NumberFormatException handling.
    - Tried to login with wrong password -> Verified "Invalid Credentials" message.

11. Challenges Faced

- **Challenge:** Implementing the "View Enrolled Students" feature required linking students back to courses.
    - *Solution:* We wrote a filter method in RecordManager that iterates through all students and checks if their grade map contains the specific Course ID.
- **Challenge:** Windows PowerShell threw errors with the wildcard compiler command.
    - *Solution:* We utilized a script to pass file paths explicitly to the Java compiler.

12. Learnings & Key Takeaways

- **OOP in Practice:** I learned how Polymorphism allows User references to behave differently depending on whether the object is a Student or Professor.
- **File I/O:** Understanding Object Streams was crucial for making the application useful (persistent data).
- **MVC Pattern:** Separating the "View" code (Main.java) from the "Logic" (RecordManager.java) made the code much cleaner.

13. Future Enhancements

- **GUI:** Implementing a JavaFX interface for a more modern look.
- **Search:** Adding a search bar to find courses by name.
- **Admin Role:** Adding a super-user who can delete users or reset passwords.

14. References

- Oracle Java Documentation (Collections Framework).
- GeeksForGeeks (Java Serialization tutorials).
- Course Lecture Notes on Object-Oriented Programming.