Assignment-1

CSE201 Advanced Programming

Release date: 9th Sep
Due by: 18th Sep,11:59 P.M.

This assignment is a take-home lab assignment. No extensions whatsoever will be provided. Any submission after the deadline will not be evaluated. If there is any ambiguity or inconsistency in a question, please seek clarification from the teaching staff. Please read the entire text below very carefully before starting its implementation.

Plagiarism: All submitted lab assignments are expected to result from your effort. You should never misrepresent someone else's work as your own. In case any plagiarism case is detected, it will be dealt with as per IIITD plagiarism policy and without any relaxations:

Academic Dishonesty in Courses Homework / Project-report / Submissions
Please note that you cannot discuss the lab assignment's design/solution (e.g., classroom page discussions). Anyone who is found doing this will be treated as a plagiarism case.
AI plagiarism will also be checked strictly.No Excuses!

NOTE that we will ONLY respond to valid questions. Make sure you ask all your doubts in advance. **Doubts on the day of submission will not be entertained**.
With doubts coming, some parts of the assignment may get updated. So, make sure you regularly follow up on doubts in the classroom. If the assignment is updated in between, it will be announced.

Problem Statement

Our university is transitioning to a new course registration system to better manage student enrollments, schedules, and academic records. You are tasked with developing this new system using object-oriented programming (OOP) concepts covered in class.

Your application must ensure that all fundamental OOP concepts—such as classes, interfaces, inheritance, polymorphism, encapsulation, and abstraction—are applied effectively. The rubrics will strictly check the implementation of all the concepts covered in class.

The assignment is up to you to your creativity, and there is no one solution. Multiple methods can exist, and you won't be penalized if they make sense.

PART 1 - Writing the Program (45 marks)

General Requirements

1. **User Roles**: The system should support three types of users—Students, Professors, and Administrators. Each user type has specific functionalities; they must log in to access these features. Students and professors can log in/sign up using their email and password. (They must create the password first to sign up). For Administrators, the password is fixed (you may choose anything)

Student Functionalities (24 marks)

1. **View Available Courses**: (2 marks)
    - Students should be able to see all available courses for the current semester, including course code, title, professor, credits, prerequisites, and timings.
2. **Register for Courses**: (8 marks)
    - Students should be able to register only for courses available in their current semester. They must select their semester first and then the course from the list.
    - Ensure that all prerequisites for a course are met before registration. Prerequisites should be courses completed in previous semesters.
    - Any student must start from semester one, and a semester will be completed only when the grade is assigned to that student by the admin.
    - To maintain the credit limit, we are fixing credits to 20, and courses can be either four or two credits.
    - Again, for simplicity, there is no complication of mandatory or electives. However, if you want to implement anything extra, we don't have any issues, but it must be correct.
3. **View Schedule**: (3 marks)
    - Students should be able to view their weekly course schedule, including details such as class timings, locations, and professor names.

4. **Track Academic Progress**: (5 marks)
    - Students should be able to view their grades for completed courses and calculate their SGPA and CGPA.
    - The GPA should be computed based on the grades from completed courses. This can only happen when the semester is complete (which will happen when all the professors assign all the grades)
5. **Drop Courses**: (2 marks)
    - A student should be able to drop a course anytime in the ongoing semester.
    - It should be within a deadline, but to keep it simple, it can be done anytime. However, feel free to implement an add/drop deadline if you wish.
6. **Submit Complaints**: (4 marks)
    - Students who encounter issues like schedule clashes or other problems should be able to submit a complaint.
    - Complaints should include a description and be assigned a status (`Pending`, `Resolved`). Students should be able to view the status of their complaints. The status can be changed by admin only.

Professor Functionalities (10 marks)

1. **Manage Courses**: (7 marks)
    - Professors should be able to view and manage their courses, including updating course details like syllabus, class timings, credits, prerequisites, enrollment limits, office hours, and anything else if required.
      Note: The professor can only read and update the course. (Adding and deleting the course is handled by admin)
2. **View Enrolled Students**: (3 marks)
    - Professors should be able to view a list of students enrolled in their courses, including their academic standing and contact details.

Administrator Functionalities (16 marks)

1. **Manage Course Catalog**: (4 marks)
    - Administrators should be able to view, add, and delete courses from the course catalog.
2. **Manage Student Records**: (4 marks)
    - Administrators should be able to view and update student records, grades, and personal information.
3. **Assign Professors to Courses**: (4 marks)
    - Administrators should assign professors to courses based on their expertise and availability.
4. **Handle Complaints**: (4 marks)
    - Administrators should view all submitted complaints and update their status (`Pending`, `Resolved`).

    - The system should allow administrators to filter complaints based on status or date and provide resolution details.

Application Flow - A terminal-based menu-driven program.

(5 marks for a smooth application flow)

1. **Start the Application**:
    - When starting the application, the user should be prompted to log in as a student, professor, or administrator.
2. **Role-based Interface**:
    - Depending on the role, present the appropriate interface for students, professors, or administrators, where they can perform the respective queries.
3. **Exit the Application**:
    - The application should provide an option to log out and exit safely, ensuring all changes are saved.

Input/Output Examples

- **Welcome to the University Course Registration System**
    - Enter the Application
    - Exit the Application
- **Login as**:
    - Student
    - Professor
    - Administrator
- **Student Mode**:
    - List of all the functionalities and logout button
- **Professor Mode**:
    - List all the functionalities and the logout button
- **Administrator Mode**:
    - List of all the functionalities and logout button

Of course, there can be multiple functionalities inside a functionality. That implementation is not fixed. However, all the functionalities must be handled because there will be set rubrics that check everything, which must be implemented strictly to handle everything.

What do functionalities inside functionalities mean?
Understand with an example:

1. I enter the app
2. I log in as a professor
3. I want to update my course details, so I selected that functionality.

4. Now I can view the details, and there are further options to update a particular field, suppose credits.
5. I select update credits, and another option asks me to choose 2 or 4.

The menu structure does not need to be fixed; it is up to you to create it in any manner. Ensure all the functionalities are available; beyond this, it is an open field for you to develop.

PART 2 (5 marks)

Make a UML Diagram with proper notation. (All five marks of Part 2 are for the UML diagram)

Submission Guidelines

- **Code**: Submit your code in a single `.zip` file with separate directories for each role's functionalities (Student, Professor, Administrator).
- **Documentation**: Include a README file explaining how to run your code, any assumptions made, and how each OOP concept is applied.
- **Demonstration**: You must demonstrate how your application works with at least three students, two professors, one administrator, and five courses.
- **Plagiarism**: Ensure all the work is your own. Plagiarism will be strictly penalized as per the university's policy.

Note: Currently, marks are assigned according to functionalities only, but using all OOP concepts covered in class is essential. If concepts are missing, marks will be deducted, even if functionality works.

You must be clear with the explanations. Otherwise, we would think you copied the code, and marks will be deducted.

The solution to complex problems is simple, mostly simple.

ALL THE BEST!!