OOP in C++ PROJECT:

Aim:

- Ensure the students are able to apply the concepts they studied in the course.
- To learn how to build and evolve programs of reasonable size using object-oriented programming, and work in teams (**where teams are of (1-3) students each**) learning from each other.

General Rules:

- A group can be formed from 1 to maximum 3 students per project.
- Groups are not allowed to repeat ideas, first come first served basis, once a group selects a project title, the project idea cannot be chosen by another group.

- There is a set of suggested ideas from which you can choose:
    - Bank Management System
    - Library Management System
    - Electric Bill Project
    - Shopping Management System
    - Hospital Management System
    - University Management System
    - Gym Management System
    - School Management System

Task: programming a simple project in OOP using C++, which is divided into two phases:

Phase 1: Designing

1. Describe the project idea (PowerPoint presentation)
2. Design UML class diagram.
3. Use of inheritance is highly recommended.

Phase 2: Implementation

1. Build all classes and their members
2. The code must contain the major OOP concepts:
    a. Structs
    b. arrays
    c. pointers
    d. classes
    e. constructors (default, and parameterized)
    f. Destructors
    g. Static members (variables and functions)
    h. Inheritance and composition relationships

i. Overriding and overloading
j. Template
k. Exception Handling

Grading Criteria:

The project is worth 15% of your final mark. These marks are graded individually by asking each student in the presentation for their contribution and testing their understanding. These are broken into:

- 5 marks for correctness (no compilation or run-time errors).
- 6 marks for the application of course concepts, where feasible, (i.e. modularity, inheritance, encapsulation, method overriding and overloading, abstract classes, …)
- 2 mark for GUI interfaces, presentation, documentation, and teamwork.
- 2 marks for all other concepts you self-studied outside the learning objectives of the course.

Submission Details:

All project files are zipped and submitted named as "proj_LeaderStudentID.zip", where "LeaderStudentID" is replaced by a leader team member chosen by all team members. The zip file should contain:

- All source code used to develop the project, either as a file, or a folder of the project packages and files.
- A Design document containing a class diagram describing the project classes, and associations, and one paragraph describing the methods, one paragraph as a user manual, and one paragraph describing team members individual contributions.
- The project must be submitted by **15/January/2026**, and the project will not be received after that.
- When there are any issues or questions, the team leader communicates with the instructor.

	Poor (0.5 pt)	Good (1 pt)	Excellent (1.5 pt)
Organization	Program is extremely unorganized. It is difficult to understand what any particular portion does.	Program is not necessarily well organized, but it is evident what each portion does. May be the result of commenting.	Program is well organized and comments describe what each major portion of the program does. Full points in this category can also be rewarded if the quality of the resulting program is such that one does not need to understand what a certain part of the code does.
Runtime	Does not execute due to errors. User prompts are misleading or nonexistent. No testing has been completed, or no input validation.	Executes without errors. User prompts contain little information, poor design. Some testing or input validation has been completed.	Executes without errors excellent user prompts, good use of symbols, spacing in output. Thorough and organized testing or input validation has been completed.
Effort	Little to no effort was put into the programming. The programmer didn't even care.	A moderate amount of effort was put into the program. However, the programmer didn't seem to care about his/her work.	Much thought was obviously put into the program. It was well crafted--and not done in a night.
Use of Input Output	No usage of Input Output	Fair use of Input Output	codes are appropriate and well done
Appropriate Data Types and Variable	Data types not suitable, variables names are ambiguous	Data types are appropriate and variables names lack clarity	Data types are appropriate with meaningful variable names
Use of Class, Pointers, Constructor ,	No usage of Class, Pointers, Constructor,	Fair use Class, Pointers, Constructor ,	codes are appropriate and well done
Use of Inheritance and composition relationships	No usage of Inheritance and composition relationships	Fair use of Inheritance and composition relationships	codes are appropriate and well done

Use of Overriding and overloading	No usage of Overriding and overloading	Fair use of Overriding and overloading	codes are appropriate and well done
Result	Program is finished, but the usability and performance leave much to be desired.	Program works fairly well, but lacks any sort of polish.	Program performs the desired function and does it well.