Evaluation Criteria for Attendance and Leave Management System Project

Grading Breakdown:

- **70% Marks**: Adherence to Software Design Principles, including correct application of design patterns.
- **30% Marks**: Working project functionality, based on the requirements provided (attendance/leave management, reporting, file handling).

---

1. Adherence to Software Design Principles

- **Design Principles:**
    - The implementation should demonstrate a clear understanding and application of **Software Design Principles** (SRP, OCP, LSP, ISP, DIP, Encapsulation, Composition over Inheritance, and Polymorphism).
    - Each class and component of the system should follow the **Single Responsibility Principle (SRP)**, ensuring that each class handles only one concern (e.g., employee records, leave management, report generation).
    - The system should be **open for extension but closed for modification (OCP)**, which means that adding new features (such as new leave types) should not require modifying existing code.
    - **Liskov Substitution Principle (LSP)** should be followed by ensuring that subclasses of leave types (e.g., CasualLeave, EarnedLeave) can be substituted where their base class (Leave) is used without affecting functionality.
    - **Interface Segregation Principle (ISP)** should be maintained by using client-specific interfaces (e.g., separate interfaces for leave management and report generation).
    - **Dependency Inversion Principle (DIP)** should be implemented so that high-level modules (e.g., leave approval process) depend on abstractions rather than low-level modules.
    - **Encapsulation** should be used to hide internal data and logic within classes, exposing only necessary methods to interact with the system.
    - Preference for **Composition over Inheritance** in class relationships
    - The system should favor **polymorphism** for handling different types of leaves or leave approval strategies.

2. Application of Design Patterns

- The system should effectively use appropriate **design patterns** to achieve scalability, flexibility, and maintainability:
    - **Factory Method Pattern** to create different types of leave objects dynamically.
    - **Strategy Pattern** for implementing different leave approval workflows (e.g., supervisor approval, director approval).
    - **Observer Pattern** for notifying employees about leave status changes.
    - **Command Pattern** to encapsulate leave-related actions (e.g., submitting, approving, or rejecting leaves).
    - **Template Method Pattern** to define the skeleton of leave request processing with specific implementations for different leave types.
    - **State Pattern** to manage the different states of a leave request (e.g., pending, approved, rejected).
    - Proper usage of these patterns will demonstrate your understanding of both **design principles** and **pattern implementation** in real-world systems.

3. Code Quality & CCCC Report

- The project should be accompanied by a **CCCC tool report**, which will calculate the **average metrics values of class** and provide insights into code quality.
    - **Cyclomatic Complexity**: Lower cyclomatic complexity is preferred, showing that the code is modular and simple to understand.
    - **Class Cohesion**: Classes should exhibit high cohesion, meaning that related functionality is grouped together in a class.
    - **Coupling**: The system should aim for **low coupling** between components to improve flexibility and reduce dependencies.
    - **Code Duplication**: Code duplication should be minimized. Refactoring to reduce redundancy is a sign of clean, maintainable code.
    - **Class Responsibility**: The classes should have clear and well-defined responsibilities based on **SRP**.
- **Marking Based on CCCC Metrics:**
    - Marks will be awarded based on the **average CCCC metrics** values of your class.
    - Projects will be graded on **overall code quality** based on these metrics, with a higher focus on **maintaining a clean design** and **low complexity**.

4. Working Project

- The system must be **fully functional**, meaning it should meet all the requirements listed in the project description.
    - It should correctly track attendance and leave records.
    - The leave application and approval process should function as described.

- Reports should generate accurately based on the available data (e.g., employees with less than a specified percentage attendance, employees with outstanding leaves).
- Data should be properly saved and retrieved from files, as specified in the project description (no database management systems allowed).
- The project must be **bug-free** and demonstrate smooth user interactions if a graphical interface is implemented (optional but bonus points may be awarded for additional effort).