SC20002 Group Assignment

Mohor, Aishwarya, Mustafa

LAB SCSX - GROUP 6

CAMS APPLICATION SYSTEM

CAMs Overview: A comprehensive software designed for managing camps, student registrations, and operations in educational settings.

SOLID Principles Compliance: Demonstrates adherence to **SOLID** design principles within its codebase.

Key OOP Features: Emphasizes loose coupling, high cohesion, code reusability, and generalization relationships, showcasing effective object-oriented programming (OOP) design.

MVC

Utilizes the Model-View-Controller (MVC) framework, with classes divided into Models, Views, and Controllers for high cohesion and loose coupling.

Class Segmentation:
Models: Include entities like “*User*”, “*Student*”, “*Staff*”.
Controllers: Comprise “*UserHandler*”, “*MessageManager*”.
Views: Consist of UI elements like “*StudentInterface*”, “*StaffInterface*”.

SOLID PRINCIPLES

Single Responsibility Principle (SRP)

"*ExcelFileManager*" for file operations

"*DataManager*" for data management

"*DatabaseSearchManager*" for database searching

Open/Closed Principle (OCP)

Extendable interfaces like “*MessageManager*”, allowing easy addition of new message types

Liskov Substitution Principle (LSP)

Subclasses like “*Enquiries*” and “*Suggestions*” are interchangeable with their base class without breaking functionality

Interface Segregation Principle (ISP)

Specific interfaces like "*CampsViewer*" for targeted functionalities

Dependency Injection Principle (DIP)

Abstract class “UserInterface”

Employed the “MessageManager” interface

Further Enhancements to our Design

Password Integrity: The 'PasswordManager' class mandates strong passwords and secures them using PBEKeySpec encryption, enhancing data security.

Custom Search Filters: 'SearchFilters' allows detailed filtering for camps and suggestions, integrated into Viewer classes for advanced search options.

Mailboxes: The 'Mailbox' feature provides immediate notifications to users about the status of their suggestions or enquiries.

DEMONSTRATION OF GOOD OOP PRACTICES

OBJECT-ORIENTED PRINCIPLES

Abstraction: Highlights unique attributes of entities like Camps and Students, focusing on specific features such as name and location.

Encapsulation: Protects data by restricting access to class members, ensuring user information security through getters and setters.

Inheritance: Implements efficient code reuse; '*Student*' and '*Staff*' classes derive from '*User*', and '*CampCommitteeMember*' extends '*Student*'.

Polymorphism: Facilitates dynamic role-based object binding; '*CurrentUserInstance*' dynamically links to appropriate user class (*Student, Staff, or CampCommitteeMember*), ensuring system flexibility and scalability.

Loose Coupling and High Cohesion

Loose Coupling through Interfaces: CAMs utilizes interfaces like '*CampCommitteeMemberHandler*' and '*EnquiryManager*', with classes such as '*CampCommitteeMemberHandler*' and '*StaffEnquiryManager*' implementing these for interaction via abstractions, reducing class dependencies.

High Cohesion in Class Design: Ensured by assigning specific, focused responsibilities to classes; '*ExcelFileManager*' manages registrations, and '*ReportGenerator*' exclusively handles report generation, enhancing manageability and maintainability.

Reusability of Code

Code Reusability through Generalization: '*Student*' and '*Staff*' classes inherit from '*User*', sharing common attributes and behaviors, which minimizes code duplication.

Interface Utilization: Interfaces like '*CampViewer*' and '*SuggestionManager*' enable various classes to implement the same methods differently, enhancing system-wide reusability.