**Object Oriented Programming**

**Project Report**

**Learning Management System**



# Submitted by:

Affan Tahir 2023-CS-709

Muhammad Abdul Rehman Wahla 2023-CS-717

# Submitted to:

Dr. Qurat ul Ain

**Dated**: 06th May, 2024

**Department of Computer Science**

**University of Engineering and Technology Lahore, New Campus**

**Learning Management System**

**Introduction**

The Learning Management System (LMS) project aims to create a comprehensive platform for managing academic activities within educational institutions. It encompasses course management, enrollment, assignment submission, grading, resource management, and student feedback. By providing tools for faculty and students to interact efficiently and effectively, the LMS aims to streamline academic processes, enhance learning outcomes, and improve the overall educational experience.

**Description**

The Learning Management System (LMS) project is a comprehensive solution designed to streamline various aspects of educational management within an institution. With a user-friendly interface, it offers administrators, faculty, and students’ efficient tools for managing courses, students, faculty, assignments, grades, feedback, and library resources.

Administrators have the ability to add new courses to the system, specifying details such as course ID, name, instructor, and credit hours. They can also manage faculty members, enabling them to add new faculty details such as name, email, and phone number. Moreover, administrators can enroll students in courses, facilitating the seamless organization of class rosters.

For faculty members, the system offers functionality to create and manage assignments. They can input assignment topics, descriptions, due dates, and associated course IDs. Additionally, faculty can input grades for student assignments and view enrollment details for their courses. They can also collect feedback from students, enhancing the overall teaching and learning experience.

Students benefit from features allowing them to enroll in courses, submit assignments, and view their grades. They can enroll in courses by providing their student ID, name, course ID, credit hours, and instructor name. Once enrolled, they can access course materials, submit assignments, and track their progress. Furthermore, students can provide feedback on courses, helping to improve teaching quality and course content.

The LMS also includes a library management module, enabling users to add and search for library resources such as books. Users can input resource details like title, author, and availability, making it easier for students and faculty to access relevant materials for their studies.

Overall, this project aims to enhance the efficiency and effectiveness of educational processes within an institution, providing a centralized platform for managing courses, assignments, grades, feedback, and library resources. By leveraging technology, the LMS empowers administrators, faculty, and students to optimize their learning and teaching experiences.

**Classes**

1. **Person**:

* id: array of integers (size 50) - stores the ID of each person
* age: array of integers (size 50) - stores the age of each person
* name: array of strings (size 50) - stores the name of each person
* email: array of strings (size 50) - stores the email of each person
* occupation: array of strings (size 50) - stores the occupation of each person

**Description:**

This class is called "person" and it represents a collection of person objects, each with attributes like ID, age, name, email, and occupation. The class uses arrays to store these attributes for up to 50 persons. It provides methods to set and get values for each attribute for a specific person at a given index in the array. The constructor initializes all arrays to default values. The destructor doesn't do anything, it's there because it's good practice to include it even if it's empty. Overall, this class allows for the management of information related to multiple persons in a program.

1. **Student**:

* student\_id: integer - stores the unique ID of the student
* name: string - stores the name of the student
* email: string - stores the email of the student
* phone\_number: string - stores the phone number of the student

**Description:**

The "student" class is an extension of the "person" class, designed to manage student-related information in a program. It inherits attributes such as ID, name, age, email, and occupation from the "person" class, and adds specific features for student management. Each student object holds details such as their student ID, name, email, and phone number. Additionally, it tracks counts for enrolled courses, submitted assignments, and joined clubs. With methods like save\_data\_to\_file(), it can store student data in files, and with setters and getters, it can manipulate and retrieve this data. The class offers functionalities like submitting assignments and joining clubs, with virtual methods ready to be overridden for specific implementations. This class is designed to streamline the management of student information, providing a foundation for handling student-related tasks within a program.

1. **Event**:

* event\_id: integer - stores the unique ID of the event
* title: string - stores the title of the event
* description: string - stores the description of the event
* date: string - stores the date of the event
* venue: string - stores the venue of the event

**Description:**

The "Event" class extends the functionality of the "student" class to manage information related to events within the program. This class contains attributes specific to events, including an event ID, title, description, date, and venue. Each event object holds details about the event such as its unique identifier, title, a brief

description, date, and venue. The class provides methods for setting and getting

these attributes. Additionally, it has virtual methods getEvent() and organize\_event(), which are intended to be overridden for specific implementations. The getEvent() method is likely to be used for retrieving event information, while organize\_event() could be utilized for organizing new events. The class utilizes dynamic memory allocation for storing strings and integers, ensuring efficient memory usage and deallocation through its destructor. This class serves as a foundation for managing event-related tasks within the program, offering functionality to create, retrieve, and organize events.

1. **Club**:

* club\_id: string - stores the unique ID of the club
* name: string - stores the name of the club
* description: string - stores the description of the club
* roll: integer - stores the roll number of club members

**Description:**

The "Club" class extends the functionalities of both the "Event" and "student" classes, providing a comprehensive structure for managing club-related activities within the program. This class encompasses attributes and methods specific to clubs, including club ID, name, description, member count, event count, and details about its members and events. Each club object holds details such as its unique identifier (club ID), name, and description. It also tracks the number of members and events associated with the club. The class includes methods for adding members to a club, organizing events, retrieving club and event information, and joining clubs.

1. **Assignment**:

* session\_id: int - ID of the attendance session.
* session\_date: string - Date of the attendance session.
* num\_students\_present: int - Number of students present in the session.

**Description:**

The "Assignment" class extends the functionality of the "student" class to handle tasks related to student assignments. This class manages information about assignments, including details such as assignment topic, ID, course ID, due date, and description. It provides methods to set and get information for each assignment, enabling manipulation and retrieval of assignment data. The class also includes functionalities to save assignment data to a file, display assignments, and submit assignments.

1. **Attendance**:

* id: array of integers - stores the ID of each person
* age: array of integers - stores the age of each person
* name: array of strings - stores the name of each person
* email: array of strings- stores the email of each person
* occupation: array of strings- stores the occupation of each person

**Description:**

The "Attendance" class is designed to manage and record attendance for various sessions within a program. It includes attributes to store session ID, session date, and the number of students present. This class provides methods

to set and get information about each session's attendance, as well as functionalities to mark attendance and display attendance records This class ensures accurate recording and management of attendance data, providing a systematic approach for keeping track of students' presence in various sessions. With input validation and file I/O functionalities, it ensures reliability and efficiency in handling attendance records within the program.

1. **Faculty**:

* faculty\_id: int - ID of the faculty member.
* stud\_id: int - ID of the student.
* name: string - Name of the faculty member.
* email: string - Email address of the faculty member.
* phone\_number: string - Phone number of the faculty member.

**Description:**

The "Faculty" class is designed to manage information about faculty members within a program. It includes attributes to store faculty ID, student ID (presumably the ID of the student being supervised), name, email, and phone number. This class provides essential functionalities for managing faculty-related tasks, such as teaching courses and supervising students, within the program. It integrates file I/O operations to save and retrieve faculty information, ensuring efficient management of faculty data.

1. **Course**:

* id: int - ID of the course.
* course\_id: string - ID of the course.
* name: string - Name of the course.
* credits: int - Credits of the course.
* instructor: string - Name of the course instructor.
* enrolled\_students: student\* - Pointer to enrolled students.

**Description:**

The "Course" class extends the functionalities of the "student" class to manage courses within a program. It includes attributes to store course details such as course ID, name, credits, instructor, and enrolled students. This class provides essential functionalities for managing courses, including enrolling students, adding assignments, and displaying course details. It maintains data integrity by ensuring proper memory management and file I/O operations.

1. **Enrollment**:

* enrollment\_id: int - ID of the enrollment.
* student\_id: int - ID of the student.
* course\_id: int - ID of the course.
* enrollment\_date: string - Date of enrollment.

**Description:**

The "Enrollment" class is designed to manage the enrollment of students in courses. It extends the functionality of the "student" class and includes attributes to store enrollment details such as enrollment ID, student ID, course ID, and enrollment date. This class provides essential functionalities to manage

student enrollments in courses, allowing for the addition and retrieval of enrollment data. It ensures proper memory management and file I/O operations to maintain data integrity and accessibility.

1. **Department**:

* department\_id: int - ID of the department.
* name: string - Name of the department.
* head\_of\_department: string - Head of the department.
* courses\_offered: Course- Array of offered courses.
* faculty\_members: Faculty - Array of faculty members.

**Description:**

The "Department" class manages the details of a department within the university. It includes attributes to store the department ID, name, head of the department, the courses offered, and faculty members. This class provides functionality to manage courses and faculty members within a department. It allows for the addition and retrieval of course and faculty data, ensuring proper memory management and file I/O operations.

1. **Grade**:

* namee: string - Name of the student.
* student\_id: int - ID of the student.
* course\_id: string - ID of the course.
* grade: string - Grade of the student

**Description:**

The "Grade" class is designed to manage and store grade information for students. It includes attributes to store the student's name, ID, course ID, and grade, as well as pointers to corresponding student and course objects. This class facilitates the management and storage of grades for students. It allows for inputting new grades and retrieving existing grades, providing essential functionality for grading and record-keeping.

1. **Library**:

* resource\_id: int - ID of the resource.
* title: string - Title of the resource.
* author: string - Author of the resource.
* availability: bool - Availability status of the resource.

**Description:**

The Library class serves as a comprehensive system for managing library resources, offering a range of functionalities tailored to streamline book management processes. These functionalities collectively provide a robust library management system, enabling efficient tracking, search, borrowing, and returning of library resources. With this class, users can easily manage and access books within the library, enhancing overall library efficiency and user experience.

1. **Project**:

* name: string - Names of the project members.
* project\_id: int - ID of the project.
* title: string - Title of the project.
* description: string - Description of the project.

**Description:**

The Project class facilitates the management of project details, allowing users to add and display information about various projects. These functionalities enable efficient management and tracking of project information, allowing users to easily add and view project details. The class supports the organization of projects with relevant information, facilitating effective project management.

1. **Feedback**:

* feedback\_id: int - ID of the feedback.
* courseid: string - ID of the course.
* content: string - Content of the feedback.
* rating: int - Rating given in the feedback.

**Description:**

The Feedback class enables students to provide feedback on courses, allowing them to share their opinions, ratings, and comments. Each feedback includes a unique ID, the name and ID of the student giving the feedback, the course ID or name, the content of the feedback, and a rating ranging from 1 to 5. Users can submit feedback through the givefeedback() method, which prompts for the student's details, the course, feedback content, and rating, saving the feedback to a file named "feedback.txt". The displayfeedback() method retrieves and displays feedback stored in the file, presenting it in a formatted manner with student names, IDs, feedback content, and ratings. This class provides a valuable tool for students to contribute their perspectives on courses, aiding in course improvement and evaluation processes.

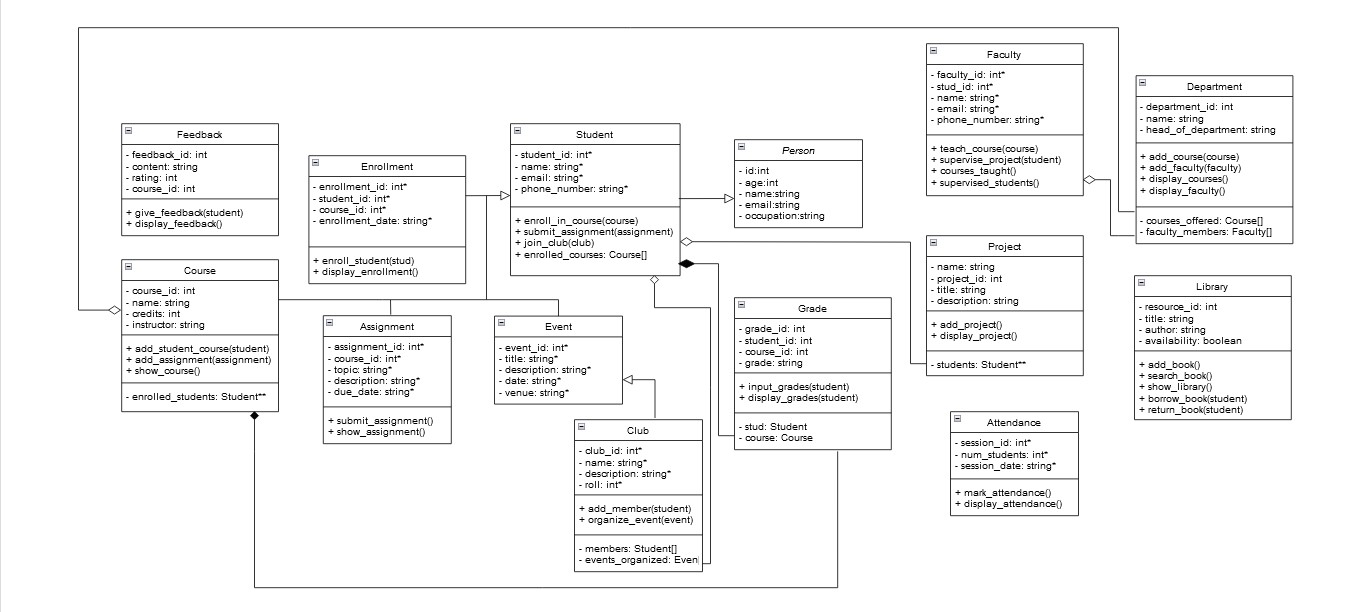
**Class Diagram**

Figure 1: Class Diagram