**Student Management System (OOP Project)**

**Project Overview**

This project is a **Student Management System** implemented using Object-Oriented Programming (OOP) principles in C++. It aims to provide functionalities for managing various school-related activities, including user management (students, teachers, and admin), course management, and handling academic records such as quizzes, assignments, lectures, reports, and notifications.

**Class Descriptions**

**1. User Class**

* **Purpose**: Acts as a base class for all types of users (students, teachers, and admin) within the system.
* **Attributes**:
  + username (string): The username of the user.
  + password (string): The password of the user.
  + role (string): The role of the user (e.g., student, teacher, admin).
  + id (int): A unique identifier for the user.
* **Methods**:
  + getUsername(): Returns the username of the user.
  + getRole(): Returns the role of the user.
  + getId(): Returns the user ID.
  + getPassword(): Returns the user's password.
  + portal(): A pure virtual function to be implemented by derived classes.
  + operator<<: A friend function for displaying user information.

**2. Teacher Class**

* **Purpose**: Represents a teacher and inherits from the User class.
* **Attributes**:
  + school (pointer to School): Pointer to the school object.
* **Methods**:
  + portal(): Provides the main interface for teachers to interact with the system.
  + addNotification(): Adds a notification for a student.
  + addQuizMarks(): Adds quiz marks for a student.
  + addReport(): Adds a report for a student.
  + addLecture(): Adds a lecture for a student.
  + addAssignment(): Adds an assignment for a student.
  + addQuiz(): Adds a quiz for a student.
  + getSchool(): Returns the pointer to the school object.

**3. Student Class**

* **Purpose**: Represents a student and inherits from the User class.
* **Attributes**:
  + school (pointer to School): Pointer to the school object.
* **Methods**:
  + portal(): Provides the main interface for students to interact with the system.
  + displayNotifications(): Displays notifications for the student.
  + displayQuizMarks(): Displays quiz marks for the student.
  + displayReport(): Displays reports for the student.
  + displayLectures(): Displays lectures for the student.
  + displayAssignments(): Displays assignments for the student.
  + displayQuiz(): Displays quizzes for the student.
  + getSchool(): Returns the pointer to the school object.

**4. Course Class**

* **Purpose**: Represents a course in the school.
* **Attributes**:
  + id (int): Unique identifier for the course.
  + name (string): Name of the course.
* **Methods**:
  + operator<<: A friend function for displaying course information.

**5. Admin Class**

* **Purpose**: Represents an admin and inherits from the User class.
* **Attributes**:
  + school (pointer to School): Pointer to the school object.
* **Methods**:
  + portal(): Provides the main interface for admins to interact with the system.
  + addNotification(): Adds a notification (not implemented in the provided code).

**6. School Class**

* **Purpose**: Represents the school, handling the storage and management of users, courses, and academic records.
* **Attributes**:
  + users (map<string, User\*>): Stores users by their username.
  + courses (vector<Course>): Stores a list of courses.
  + quizMarks, lectures, assignments, reports, quizzes, notifications (map<string, map<string, string>> and map<string, map<string, int>>): Stores various academic records by student username and subject.
* **Methods**:
  + ~School(): Destructor to clean up dynamically allocated user objects.
  + addUser(User\* user): Adds a user to the school.
  + addCourse(const Course& course): Adds a course to the school.
  + displayCourses() const: Displays all courses.
  + addQuiz(), addReport(), addLecture(), addAssignment(), addQuizMark(), addNotification(): Methods to add various academic records.
  + getNotifications(), getQuizMarks(), getReports(), getLectures(), getAssignments(), getQuizzes(): Methods to retrieve various academic records for a student.
  + login(): Handles user login and redirects to the respective user portal.
  + clearConsole() const: Clears the console (platform-dependent).

**Main Function**

* **Initialization**: The main function displays an ASCII art banner, initializes a School object, and adds some test users and courses.
* **Menu**: Provides a menu interface for logging in or displaying courses. The user can log in by entering their username and password. If the credentials are correct, the user is redirected to their respective portal (student, teacher, or admin).

**Key Features and Functionality**

* **User Management**: Allows the creation and management of different user types (students, teachers, admins).
* **Course Management**: Supports adding and displaying courses.
* **Academic Records Management**: Teachers can add lectures, assignments, quizzes, reports, and notifications for students. Students can view their respective academic records.
* **Admin Management**: Admins can add courses, teachers, and students.
* **Authentication**: Provides a simple login mechanism to authenticate users.

**Improvements and Enhancements**

* **Error Handling**: Add proper error handling and validation for user inputs.
* **Data Persistence**: Implement data persistence (e.g., using files or a database) to save and load users, courses, and academic records.
* **User Interface**: Enhance the user interface for better usability and user experience.
* **Role-Based Access Control**: Implement more granular role-based access control to ensure users can only access authorized sections.
* **Function Implementation**: Implement missing functions like Admin::addNotification().

**Conclusion**

This Student Management System project demonstrates a solid application of OOP principles in C++. It covers the fundamental aspects of managing users, courses, and academic records in a school environment. With further enhancements and improvements, this system could be developed into a comprehensive solution for managing school operations.