**ProManage - Project Management System**

**Introduction**

ProManage is a robust Project Management System designed to streamline collaboration between Admins and Clients. It provides functionalities like **Task Management**, **Meeting Management**, **Notifications**, and **File Upload/Download**. Built using Object-Oriented Programming principles, the system ensures scalability and maintainability, while a SQL database backend manages data securely and efficiently.

**System Overview**

**Actors**

1. **Admin**: Responsible for managing users, assigning tasks, scheduling meetings, and monitoring progress.
2. **Client**: Interacts with assigned tasks, attends meetings, and receives updates through notifications.

**Features**

* **Task Management**: Allows Admins to assign, manage, and track tasks. Clients can view, submit, and mark tasks as complete.
* **Meeting Management**: Provides a structured way to schedule, manage, and view meetings between Admins and Clients.
* **Notifications**: Ensures real-time updates for task assignments, meeting reminders, and task status changes.
* **File Upload/Download**: Facilitates document exchange between Admins and Clients, enhancing collaboration and task completion.
* **SQL Database Integration**: All data, including user credentials, tasks, meetings, and uploaded files, is securely stored and retrieved using an SQL database.

**OOP Principles in ProManage**

**1. Encapsulation**

Encapsulation is applied by keeping data (like user credentials, task details, and meeting data) private within their respective classes. Public getter and setter methods are used to access or modify this data in a controlled way.

* Example: The User class contains private attributes for username, password, and role. These attributes can only be accessed through designated getter and setter methods, preventing unauthorized modifications.

**2. Inheritance**

Inheritance is used to eliminate redundancy and promote code reuse. The User class serves as a base class for Admin and Client, allowing shared properties (e.g., username, ID) and methods to be inherited, while enabling role-specific customizations.

**3. Polymorphism**

Polymorphism allows the system to dynamically execute role-specific behavior at runtime. For instance, both Admin and Client objects may use the same viewTasks() method, but the implementation varies depending on the user's role.

**4. Abstraction**

Abstraction hides implementation details and provides a simplified interface for complex functionalities. For instance, controllers abstract the intricate details of task assignments, meeting scheduling, and database operations, presenting a clean interface to the UI.

**Design Patterns in ProManage**

**1. Factory Pattern**

The Factory Pattern is used for creating objects dynamically based on user roles. For example:

* When a user logs in, the system identifies their role (Admin or Client) and creates the appropriate object instance. This ensures that object creation is consistent and centralized.

**2. Singleton Pattern**

The Singleton Pattern ensures that critical components, like the database connection handler, have only one instance throughout the application's lifecycle. This avoids redundant connections, optimizes resource usage, and provides a global point of access for database operations.

**3. Adapter Pattern**

The Adapter Pattern is used to integrate components with incompatible interfaces. For instance, during file upload/download processes, adapters may convert data between the system’s internal format and external formats, ensuring compatibility and smooth functionality.

**System Structure**

**Three-Layered Architecture**

ProManage adheres to a well-structured three-layered architecture to ensure separation of concerns:

1. **UI Layer**: Includes JavaFX-based graphical interfaces for Admins and Clients. These interfaces handle user interactions like logging in, viewing tasks, and managing meetings.
2. **Business Logic (BL) Layer**: Manages the core functionalities of the system, such as assigning tasks, scheduling meetings, processing notifications, and handling file uploads/downloads.
3. **Database (DB) Layer**: Handles secure storage and retrieval of all system data, including users, tasks, meetings, notifications, and files, using an SQL database.

**Class Diagram Summary**

Key classes in the system include:

* **User**: A base class for Admin and Client, encapsulating shared properties and methods.
* **Controllers**: Manage interactions between the UI and Business Logic layers. For example, LoginController, AdminTaskController, and MeetingController.
* **Task Management**: Handles creation, assignment, tracking, and submission of tasks.
* **Meeting Management**: Facilitates scheduling and viewing of meetings.
* **DatabaseConnection**: Implements the Singleton Pattern to manage a single, global instance of the database connection.

**Core Features**

**1. Task Management**

Admins can:

* Create and assign tasks to Clients.
* Track task progress (e.g., pending, in progress, completed).

Clients can:

* View assigned tasks.
* Submit tasks with relevant documents or mark them as complete.

**2. Meeting Management**

Admins can:

* Schedule meetings and notify Clients.
* View and manage meeting details.

Clients can:

* View scheduled meetings.
* Join meetings as notified.

**3. Notifications**

* Notifications are sent in real-time to inform users about:
  + New task assignments.
  + Meeting updates or reminders.
  + Task completion acknowledgments.

**4. File Upload/Download**

* The system supports file uploads (e.g., documents related to tasks) and downloads for enhanced collaboration.
* All files are securely stored in the SQL database and can be accessed by authorized users.

**SQL Database Integration**

ProManage uses a relational SQL database for data persistence. Key database entities include:

1. **Users**: Stores credentials, roles (Admin/Client), and profile information.
2. **Tasks**: Tracks task details, assignment status, and associated files.
3. **Meetings**: Records meeting schedules, participants, and updates.
4. **Notifications**: Logs notifications sent to users, ensuring clarity and traceability.

**Key Benefits of ProManage**

1. **Centralized Project Management**: Combines task, meeting, and notification management in one platform.
2. **Role-Based Access**: Differentiates functionalities for Admins and Clients, ensuring streamlined operations.
3. **Scalability**: OOP principles and design patterns enable easy feature additions and modifications.
4. **Seamless Collaboration**: File upload/download functionality fosters efficient collaboration between Admins and Clients.
5. **Efficiency**: The Singleton Pattern optimizes database operations, while the Factory Pattern ensures consistent object creation.

**Future Enhancements**

1. **Advanced File Management**: Add features like version control for uploaded files and larger file storage capabilities.
2. **Mobile Application**: Develop a mobile version for easier access and updates on the go.
3. **Enhanced Notifications**: Introduce support for email or SMS notifications for critical updates.
4. **Meeting Integration**: Add functionality to integrate with third-party tools (e.g., Zoom or Google Meet) for virtual meetings.
5. **Task Analytics**: Include basic analytics for Admins to track overall project progress.

**Conclusion**

ProManage is an intuitive and scalable solution for project collaboration. By leveraging OOP principles, SQL database integration, and design patterns like Factory, Singleton, and Adapter, it provides robust task and meeting management while ensuring a smooth user experience. With its existing features and scope for future enhancements, ProManage is well-suited for project teams of all sizes.