Here's how you can incorporate some design patterns into your library system features:

1. Singleton:

- Use a Singleton for the **Configuration Manager**: This class can hold library-wide settings like fines, borrowing limits, and renewal periods. There would only ever be one instance of this class accessible throughout the system.

2. Factory:

- Implement a **User Factory**: This factory can create different user objects depending on the user type (patron, librarian, administrator). This promotes code reusability and simplifies user creation logic.

3. Abstract Factory:

- Create an **Abstract Item Factory**: This factory can have subclasses for creating specific types of items (books, audiobooks, DVDs). This allows for easy addition of new item types in the future without modifying existing code.

4. Decorator:

- Use the Decorator pattern for **Fines**: A base Fine class can be extended with decorators for different overdue scenarios (late fees, replacement cost, etc.). This allows for flexible fine calculation based on specific conditions.

5. Adapter:

- Implement an Adapter for **Legacy Data Import**: If your library has existing data in a non-standard format, an adapter can convert that data into a format compatible with the library system.

Remember, using design patterns should be done thoughtfully to improve code maintainability and flexibility. Not all features require a design pattern. Choose the pattern that best suits the specific functionality you're trying to achieve.