Zaimplementowane Wzorce Projektowe:

1. Singleton – zaimplementowany w pliku **library_catalog.py**

Zapewnia, że istnieje tylko jedna instancja klasy LibraryCatalog, która przechowuje wszystkie książki. Zapewnia spójność danych w systemie: wszystkie moduły korzystają z tego samego katalogu. Nie ma potrzeby tworzenia wielu instancji katalogu. Gwarantuje jedno źródło prawdy o książkach w systemie.

2. Adapter – zaimplementowany w pliku **data_adapter.py**

Konwertuje dane książek z różnych formatów (JSON, XML, CSV) na standardowy format używany w LibraryCatalog. Został użyty ponieważ dane o książkach mogą pochodzić z różnych źródeł i w różnych formatach. Adapter ujednolica te dane, co pozwala na łatwą integrację z systemem. Dzięki temu system jest bardziej elastyczny – można łatwo dodać obsługę nowych formatów danych.

3. Factory – zaimplementowany w pliku **user_factory.py**

Tworzy różne typy użytkowników, np. Student, Teacher, Librarian, na podstawie danych wejściowych. Pozwala na uproszczenie procesu tworzenia obiektów użytkowników. Ułatwia dodawanie nowych typów użytkowników w przyszłości. Kod jest bardziej czytelny i elastyczny.

4. Observer – zaimplementowany w pliku **observer.py**

Umożliwia subskrybowanie powiadomień przez użytkowników. Powiadomienia są wysyłane, gdy w katalogu pojawia się nowa książka. Ułatwia informowanie użytkowników o zmianach w systemie (np. dostępności książki). Zapewnia automatyzacje powiadomień, co poprawia doświadczenie użytkownika.

5. Iterator – zaimplementowany w pliku **iterator.py**

Umożliwia przeglądanie kolekcji książek w katalogu w ustrukturyzowany sposób. Zapewnia spójny sposób iteracji po danych (np. książkach lub użytkownikach). Ułatwia przeglądanie dużych zbiorów danych w katalogu.

6. Facade - zaimplementowany: W pliku **library_interface.py**

Zapewnia uproszczony interfejs do wykonywania typowych operacji, takich jak wyszukiwanie książek, dodawanie książek, wyświetlanie książek i zarządzanie użytkownikami. Fasada ukrywa złożoność systemu, zapewniając prosty i ujednolicony sposób interakcji z systemem. Łączy funkcjonalności z różnych podsystemów, takich jak LibraryCatalog, ObserverCatalog, UserFactory i Iterator. Dzięki niemu można dodać nowe operacje bez modyfikacji kodu w podsystemach.