# Library Management System - Project File Explanations

## include/LibraryItem.h

This is the base abstract class for all library items. It contains common attributes like title, author, and itemID, and defines two pure virtual functions:  
- displayInfo(): to show item details  
- isBorrowable(): to check if the item can be borrowed

## include/Book.h & src/Book.cpp

The Book class inherits from LibraryItem. It adds a borrowed flag and implements how books behave:  
- Can be borrowed and returned  
- Displays if the book is available or not

## include/Magazine.h & src/Magazine.cpp

The Magazine class also inherits from LibraryItem. It cannot be borrowed, so isBorrowable() always returns false. Used to show magazines in the catalog.

## include/User.h & src/User.cpp

The User class represents a library user. It stores the user's ID, name, and a list of borrowed book IDs. It has functions to:  
- Borrow and return books  
- Check if a book is already borrowed  
- Display the user's information

## include/LibrarySystem.h & src/LibrarySystem.cpp

This is the main manager class for the system. It holds all the books, magazines, and users using maps. It handles actions like:  
- Adding books/magazines  
- Registering users  
- Borrowing/returning books  
- Viewing all items and users

## include/LibraryException.h & src/LibraryException.cpp

This is a custom exception class used for error handling. If something goes wrong (like book not found), we can throw LibraryException with a message. It inherits from runtime\_error. This file is optional, but helps make the code more professional.

## main.cpp

This is the main file of the program. It shows a menu using the console, and lets the user:  
- Add books or magazines  
- Register users  
- Borrow/return books  
- View everything  
It uses the LibrarySystem class to perform the actions and also logs each action to a file using logTransaction().

## logs/transactions.txt

This text file contains a history of what happened in the system: Who registered, which book was borrowed or returned, etc. It’s created automatically when actions happen.

## Makefile

This script compiles and runs the project. Using `make`, it compiles all files, links them, and runs the program. `make clean` deletes all build files and logs.

## Summary Explanation for the Professor

I structured the project using object-oriented programming principles. Each class has its own header and implementation file. I used inheritance for library items, and polymorphism to allow books and magazines to behave differently. To keep the code clean, I created a LibrarySystem class to manage all logic, and I used a custom exception for error handling. I also added logging to a file for every action, and made a Makefile to compile and run everything smoothly.