**Assumptions – Librarian Module**

* **Librarian login functionality:**
* The assumption in this requirement is about which module is managing the logging in functionality for librarians, which was decided to be inside of the Admin module itself, where it will have access to the database of registered Librarians. This method calls the “libLogin” function from the Admin instance, which will let us know if it was successful or not. The form validation will be done inside of the “libLogin” function, and we will make use of the returned value.
* **Adding a new book in the system (Book Information):**
* The main assumption in this method was to take enough information about the book, that it suites the database and makes it unique enough. I decided to take the book ISBN (a unique identifier of the book), the name, the author, total pages and the stock number (provides how many books that can be rented). This form gives us enough information about the book, to be unique and if the book is unique with only 1 stock, it cannot be issued to multiple students at the same time. Whereas, if the stock is over 1, we assume that book has multiple copies to be issued.
* Another assumption in this method was to specialize it for the addition of books and books only. This would not add any magazines or videos, as the library can only hold them, but cannot do anything with it yet as there was nothing related to it inside of the entire requirements list for every module. We added data about magazines and videos, but it is not used anywhere in the program, but can still be fetched if we have to add functionality about it later.
* **View all the books in the system:**
* We are fetching data from the main database in this requirement, as we parse all the books only and nothing else. This method will return all the books inside of the current library catalog, but it will not generate an output file because we already have the main database to reference from and to. Which will reduce redundant code and also provide current information to be made available instantly.
* **Issuing a book:**
* My understanding for this requirement was such that I should create an automated system to issue books, which also runs a validation that if the book does not have any other issues, it will basically move the student to the waiting list. This also takes the student into consideration, of which if a book is returned, the person on the waiting list queue will be appended to the borrowed students, and whenever the student logs in, they will see their new book. This also contains a status method, which the Student instance can use on login to see if any new books are issued for that specific student. Instead of creating a frontend interface of approving the books and putting students in the waiting list manually, I took inspiration of current library systems and their automations. So I did not design a frontend because it was not necessary when all the automation is happening in the backend using the borrow Book method.
* This same exact assumption is being made about the returns as well, because in the real world, a librarian is not taking books to return manually. The system does it by itself by taking the ISBN and processing that information automatically, while also enabling the book to be borrowed by the next student right away. This basically allows the librarian to focus on adding and deleting books, while the system takes care of the returns and borrows, simultaneously running validation checks.