**Design Rationale**

**Dictionary**

The use of the dictionary was because of its unique property to store a collection of key value pairs and there mutability allowing for unordered collection of information where each unique key maps to a specific value. This structure allows for efficient retrieval of values by referencing their corresponding keys. And there for it was a great fit for the library management app

**List**

Similar to the dictionary, a list serves as an ordered, mutable collection of items. It is a fundamental data structure used to store multiple items in a single variable. And making use of this property we are able to store many different names of books or total copies of a particular book all under one variable.

**Tuples**

For the use of the tuple we took advantage of its immutability property which is very suitable for things like genre that we don’t want people to add too or change in anyway shape or form.

**Operation.py Explanation**

In Summary every operation needed a function for it to be possible the file **operation.py** was created to build a simple Library Management System using Python’s basic data structures and functions. The goal was to manage books and members by allowing adding, searching, updating, deleting, borrowing, and returning of books. A dictionary was used for books because each book has a unique ISBN, making it easy to look up and update details quickly. Each book record stores information such as title, author, genre, total copies, and available copies. Members were stored in a list of dictionaries since there can be many members, and a list makes it easy to loop through and find a specific member. Each member has an ID, name, email, and a list of borrowed books. The genres were stored in a tuple to keep them fixed and prevent any changes during the program’s execution.

I created Helper functions such as **find\_member** and **book\_exists** to reduce repetition and make the code easier to read. The main functions handle all operations like adding, updating, deleting, borrowing, and returning books, with checks to make sure data stays valid. For example, the system prevents adding duplicate ISBNs, borrowing more than three books, or deleting a book that is still borrowed. Instead of printing messages, each function returns **True** or **False** to make testing easier. This makes it easier to use test scripts to **assert** statements to check if the system works correctly.

Overall, I designed the **operation.py** to be clear, reliable, and easy to test. Dictionaries make managing books efficient, lists make handling members flexible, and tuples keep the genres fixed. The code structure ensures that the system runs smoothly, is easy to understand, and can be expanded or improved later if needed.