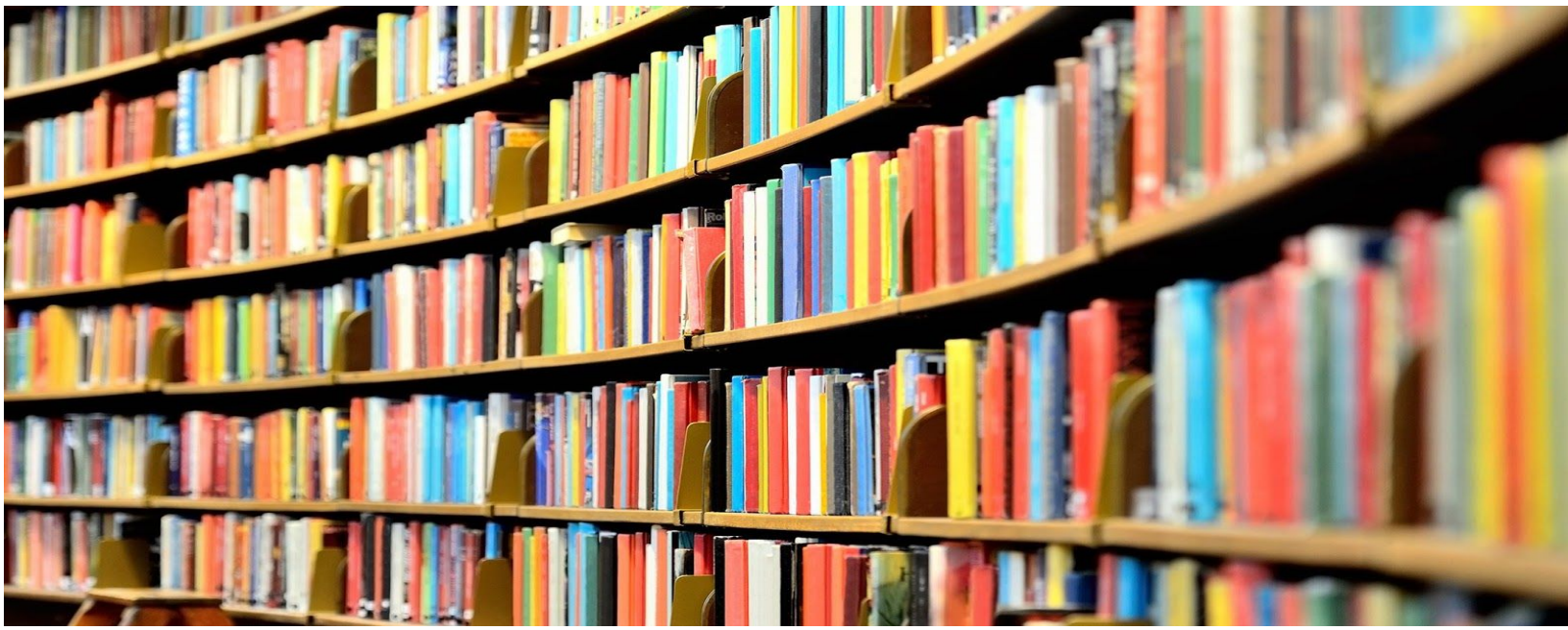
EDYODA

December 27, 2019

Library Management System



Objective

1. Develop use case diagram (Tool - draw.io). Major components of Library - Member, Librarian & System. Actions take care of each of these. This is decomposition.
2. Identify similar things (issue & return book)& bundle them together - pattern recognition.
3. Derive classes out of these. Abstraction - Taking only the necessary information. Develop a class diagram using draw.io.
4. Complete the Library Management Prototype Project, all the unimplemented functions. (50 marks). Prototype Code - https://github.com/edyoda/Python-Django-Program/blob/master/Library%20Mgmt%20System.zip
5. Flow chart for issue book & return book.
6. Testing the application & show the results
    a. A user issues a book should change the inventory.
    b. The same user returns the book back should change the inventory.

Use Case Diagram:

Use case diagrams are mainly used to analysing requirements and showing functionalities of our system. First, we need to identify the role of the user. Then we will analyse requirements for specific functionality. We are using some relationships between use cases ex. Include and extend.

Include relation means that without a sub-use case, base use case is incomplete. Related cases will execute every time whenever the base use case will be executed. Example: - whenever librarian add any book it will automatically add book item and update catalog without updating catalog book can't be added.

Extend relation is inserting an additional sequence into a base use case sequence. Example: - whenever a member search for a book in the catalog, he can search by name or author.

Functionalities:

1.Add/Remove/Edit book: updating inventory by adding, removing or editing a book item.

2.Search catalog: To search books by title and author.

3.Create new account/cancel membership: To add a new member or cancel the membership of an existing member.

Class Diagram:

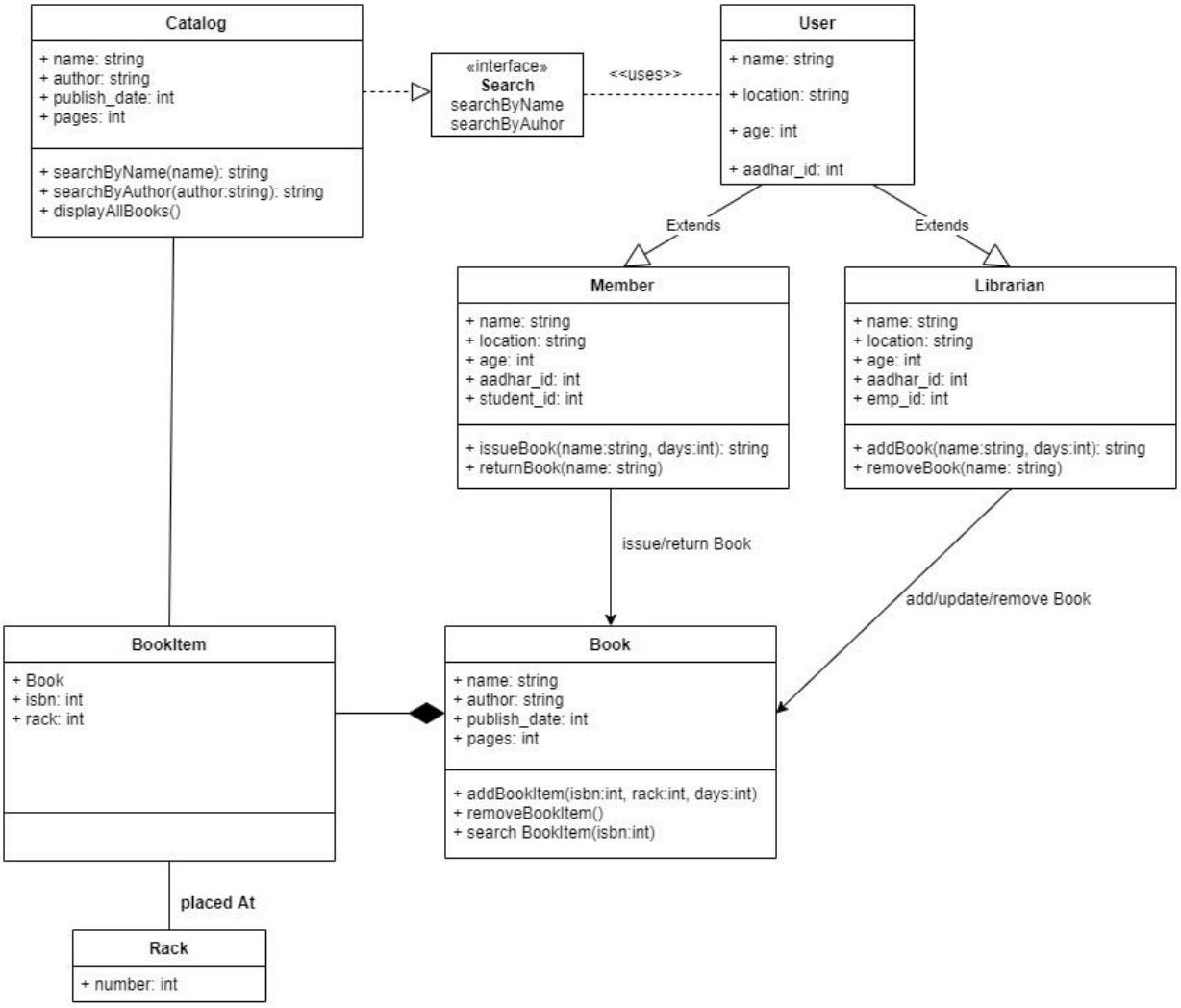
These are our Main Classes:

Catalog : Catalogs contain list of books. We can search catalogs by: Name and Author.

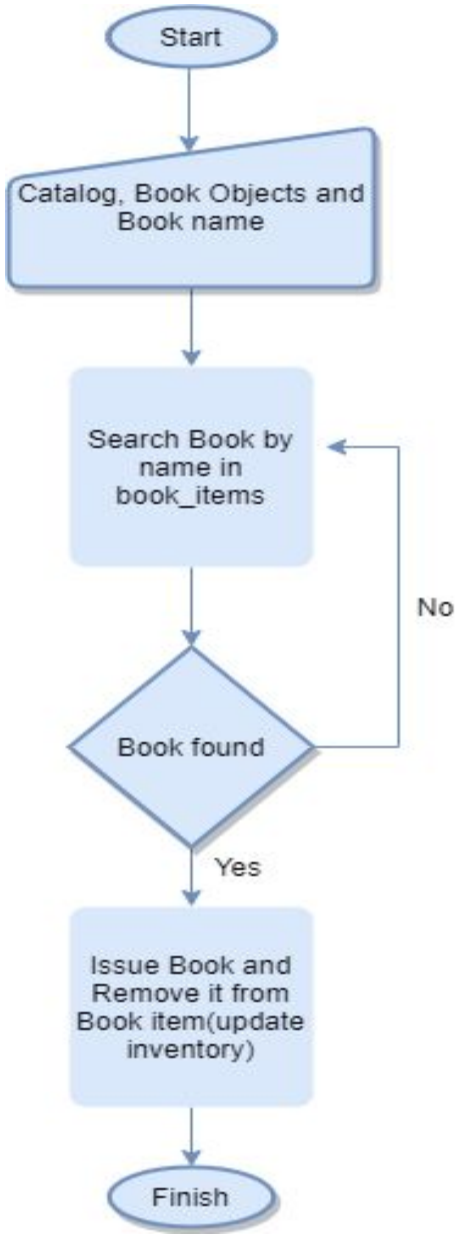
Book: The basic building block of the system. Every book will have Name, Subject, Published date, Number of pages etc.

BookItem: Any book can have multiple copies, each copy will be considered a book item in our system. Each book item will have an ISBN and rack number where it is kept.

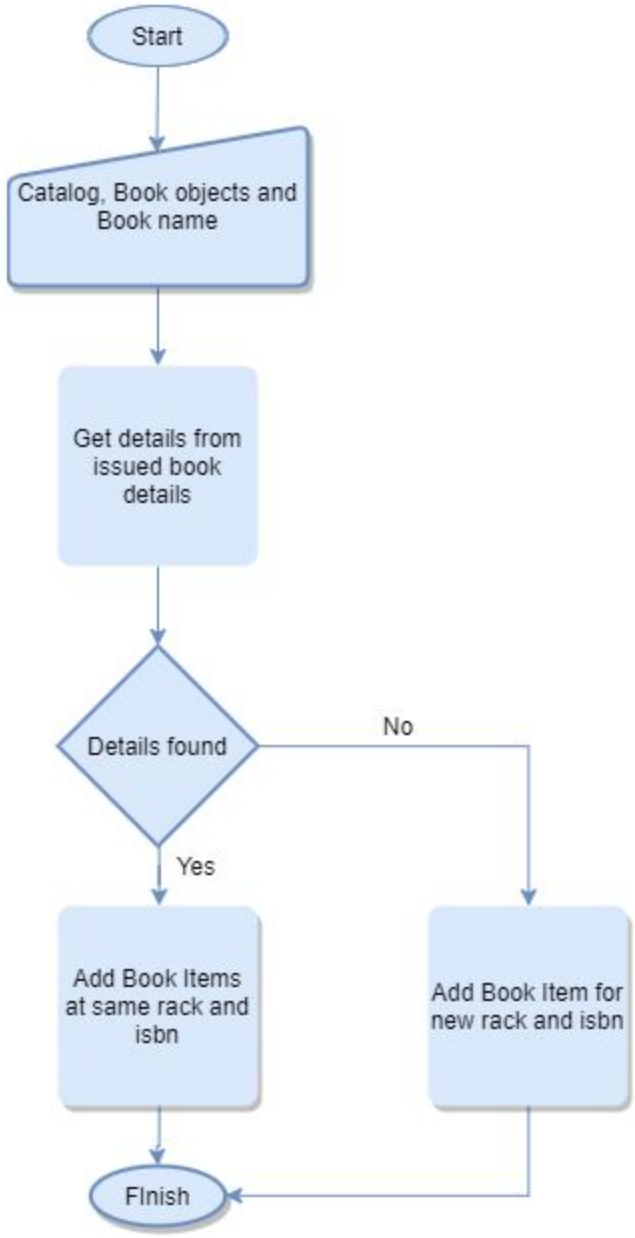
Account: We will have two types of accounts in the system, one will be a member, and the other will be a librarian.


Catalog
+ name: string
+ author: string
+ publish_date: int
+ pages: int
+ searchByName(name): string
+ searchByAuthor(author:string): string
+ displayAllBooks()
«interface»
Search
searchByName
searchByAuhor
<<uses>>
User
+ name: string
+ location: string
+ age: int
+ aadhar_id: int
Extends
Extends
Member
+ name: string
+ location: string
+ age: int
+ aadhar_id: int
+ student_id: int
+ issueBook(name:string, days:int): string
+ returnBook(name: string)
Librarian
+ name: string
+ location: string
+ age: int
+ aadhar_id: int
+ emp_id: int
+ addBook(name:string, days:int): string
+ removeBook(name: string)
issue/return Book
add/update/remove Book
BookItem
+ Book
+ isbn: int
+ rack: int
Book
+ name: string
+ author: string
+ publish_date: int
+ pages: int
+ addBookItem(isbn:int, rack:int, days:int)
+ removeBookItem()
+ search BookItem(isbn:int)
placed At
Rack
+ number: int

Flow chart for issue a book:



Flow chart for return a book:



Memory Layout Diagram

As we are not using any database in this project. This is our memory layout diagram, like how we are storing our books and book items. Catalog has information about each book and book

items as well. Each has info about its items. Finally, our book items are all about how many copies of a book we have and where it is kept in our library.

Below is the image of our memory layout diagram.