Library Circulation system



عمل الطالبات:

لميس العلفي هديل القاضي

فاطمة القديمي رهان السلامي

1. Functional Requirements

**Functional Requirements**

1. **Search Collection**
   1. Library’s collection database should be available to potential borrowers on terminals via the Web.
   2. The Borrower search the books by title, author, keywords, and ISBN.
   3. A valid borrower requests the checked-out book to be returned.
   4. The Librarian notifies the borrower of the book’s availability.
2. **Borrow Books**
3. There are three types of borrowers (student, stuff, guest)
4. The Borrower provides Librarian their ID card.
5. The Librarian checks the validity of the ID Card.
6. The Librarian checks whether the Borrower has any overdue books and or fines
7. The Borrower checks out the books.
8. The system must update the library’s collection database to reflect the book’s new status.
9. **Return Books**
10. The Borrower Return Books.
11. The Librarian update the library’s collection database to reflect the book’s new status.
12. **Maintain Book Collection**
13. The Librarian checks damaged Books.
14. The Librarian removed damaged Books from the library’s book collection.
15. The borrower pays a fine**.**
16. **Process Overdue Books**
17. the Librarian sends reminder emails to borrowers.
18. The borrower pays a fine.

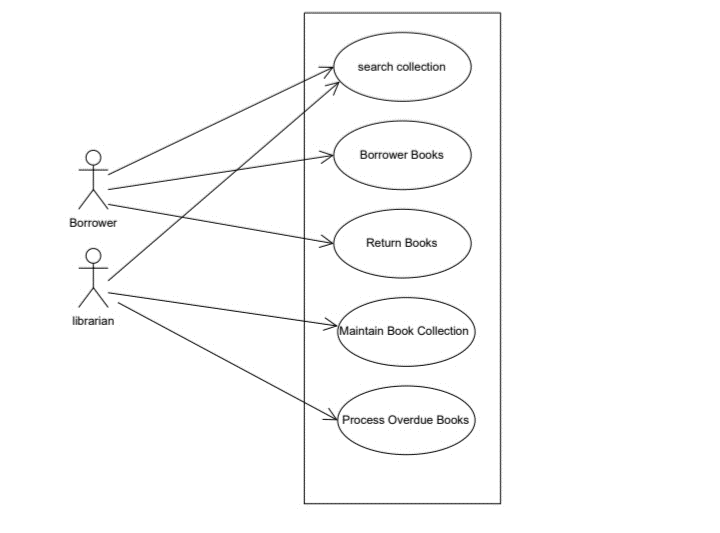
2.Non\_ Functional Requirements

**Nonfunctional Requirements**

* 1. **Operational Requirements**

1. The system will operate in Windows environment.
2. The system should automatically back up at the end of each day.
3. **Performance Requirements**
4. The system will store a new borrower info in 5 seconds or less.
5. The system will retrieve the daily checks out schedule in 5 seconds or less.
6. **Security Requirements**
7. Only Librarian can set their availability.
8. Only a manager can produce a schedule.
9. **Cultural and Political Requirements**
10. No special cultural and political requirements are anticipated.

3. Use Case Diagram:



4. Use Case Description:

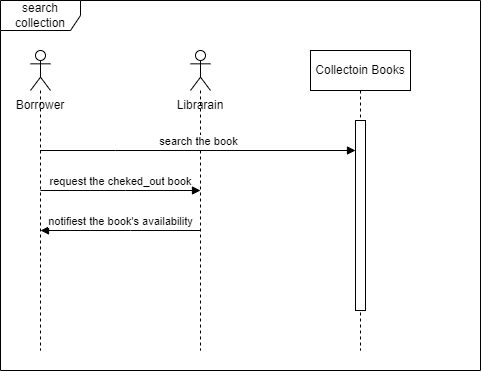
|  |  |
| --- | --- |
| **search collection** | **Use case name** |
| Borrower, librarian | **Participating actor** |
| 1. Library Collection Database should be available. | **Entry condition** |
| 2. The Borrower search the books by title, author, keywords, and ISBN  3. A valid borrower requests the checked-out book to be returned | **Event flow** |
| 4. The Librarian notifies the borrower of the book’s availability | **Exit condition** |
| E1 No books found  1.The system informs the borrower that the books are not found | **Exceptional flow** |

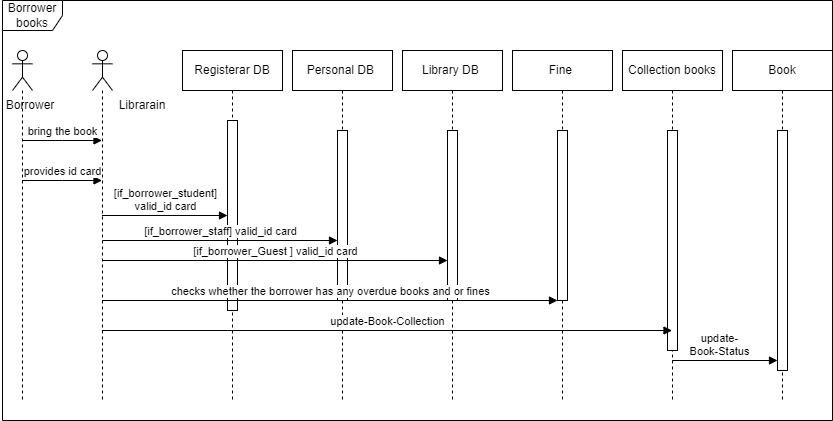
|  |  |
| --- | --- |
| **Borrower Books** | **Use case name** |
| * Borrower | **Participating actor** |
| 1. The borrower bring the book to the librarian at the check-out desk | **Entry condition** |
| 2. The Borrower provides Librarian their ID card.  3. The Librarian checks the validity of the ID Card.  -If the Borrower is a Student Borrower, Validate ID Card against Registrar's Database.  - If the Borrower is a Faculty/Staff Borrower, Validate ID Card against Personnel Database.  - If the Borrower is a Guest Borrower, Validate ID Card against Library's Guest Database.  4. The Librarian checks whether the Borrower has any overdue books and or fines  5. The Borrower checks out the books. | **Event flow** |
| 6. The system does update the library’s collection database to reflect the book’s new status. | **Exit condition** |
| **E1: if the borrower`s ID card is invalid**  1.The book request is rejected.  E2: borrower has overdue or fine  1 .If the Borrower either has overdue books, fines, or both, the book request is rejected. | **Exceptional flow** |

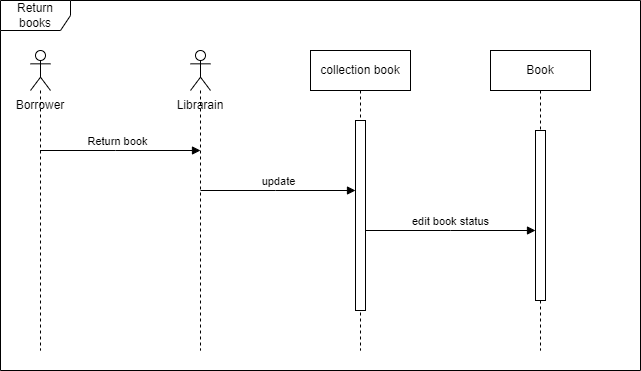
|  |  |
| --- | --- |
| **Return Books** | **Use case name** |
| * Borrower | **Participating actor** |
| 1. The Borrower brings books to return desk | **Entry condition** |
| 2. The Borrower Return Books. | **Event flow** |
| 3. The Librarian update the library’s collection database to reflect the book’s new status | **Exit condition** |
| \_\_\_\_\_\_\_\_\_\_\_ | **Exceptional flow** |

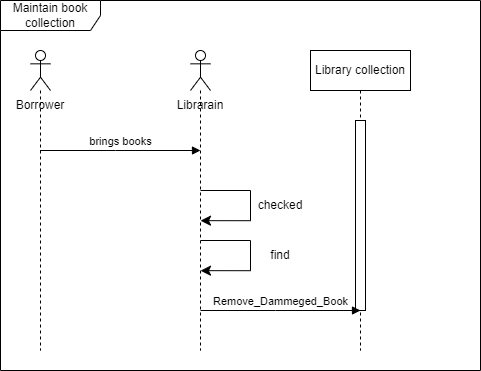
|  |  |
| --- | --- |
| **Maintain Book Collection** | **Use case name** |
| * Librarian | **Participating actor** |
| 1. Borrower brings books to return desk | **Entry condition** |
| 1. The Librarian checks damaged Books. 2. The Librarian find damaged book | **Event flow** |
| 4. The borrower pays a fine**.** | **Exit condition** |
| E1 found damaged books  1. The Librarian removed damaged Books from the library’s book collection | **Exceptional flow** |

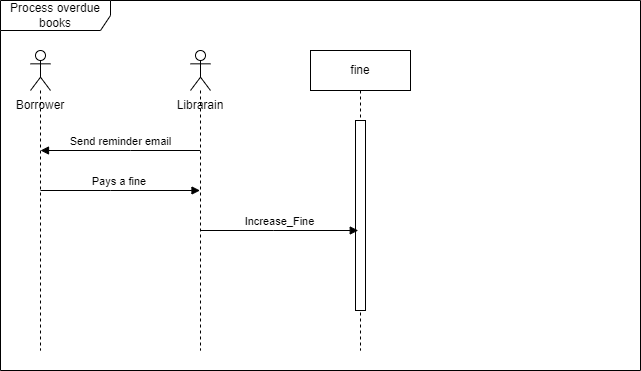
|  |  |
| --- | --- |
| **Process Overdue Books** | **Use case name** |
| * Librarian | **Participating actor** |
| 1. The borrower was late in returning the books | **Entry condition** |
| 2. The Librarian sends reminder emails to borrowers. | **Event flow** |
| 3. The borrower pays a fine | **Exit condition** |
| E1 Did not pay the fine  1. The Librarian sends reminder emails to borrowers and increases the fine. | **Exceptional flow** |

5. Sequence Diagram: 

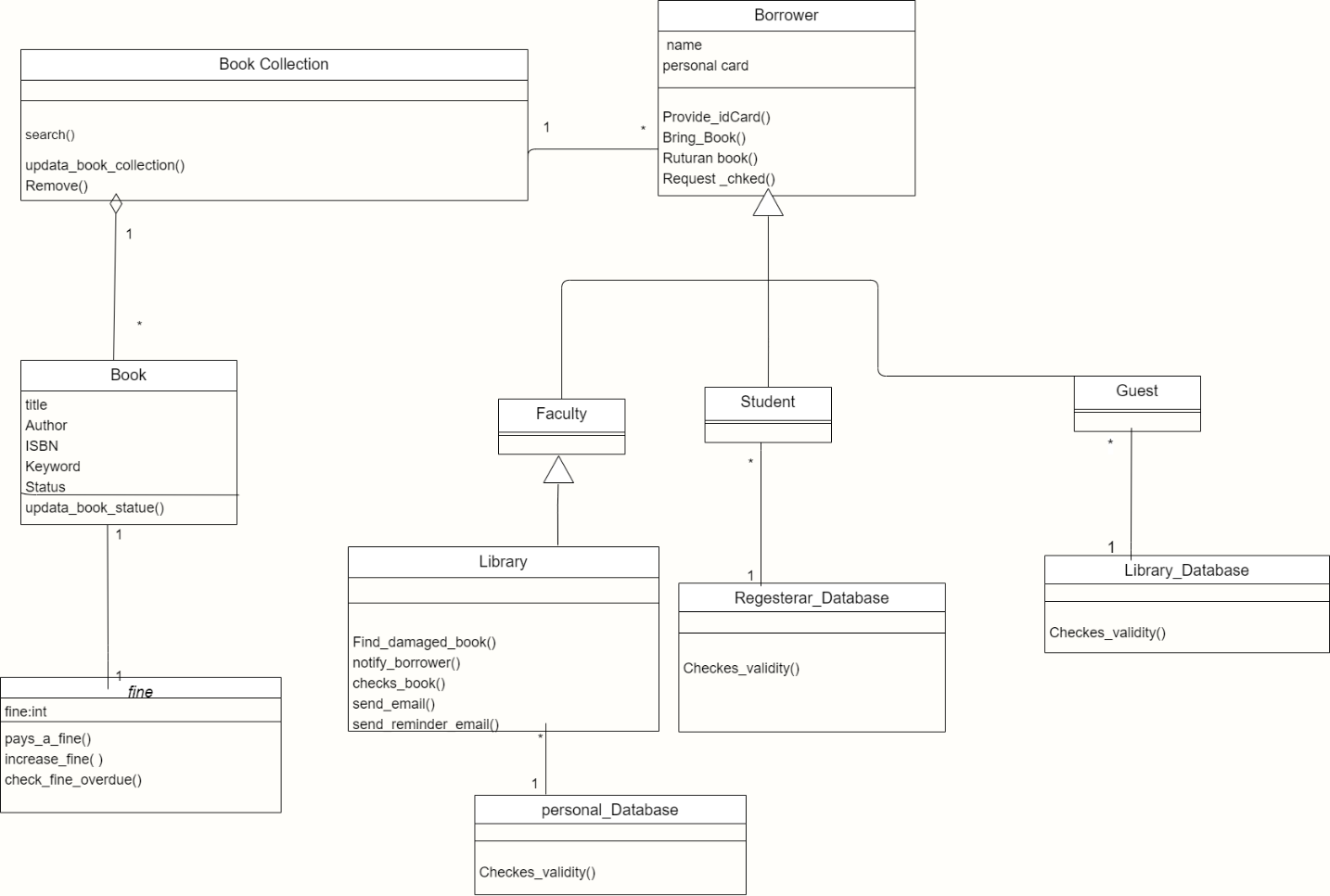




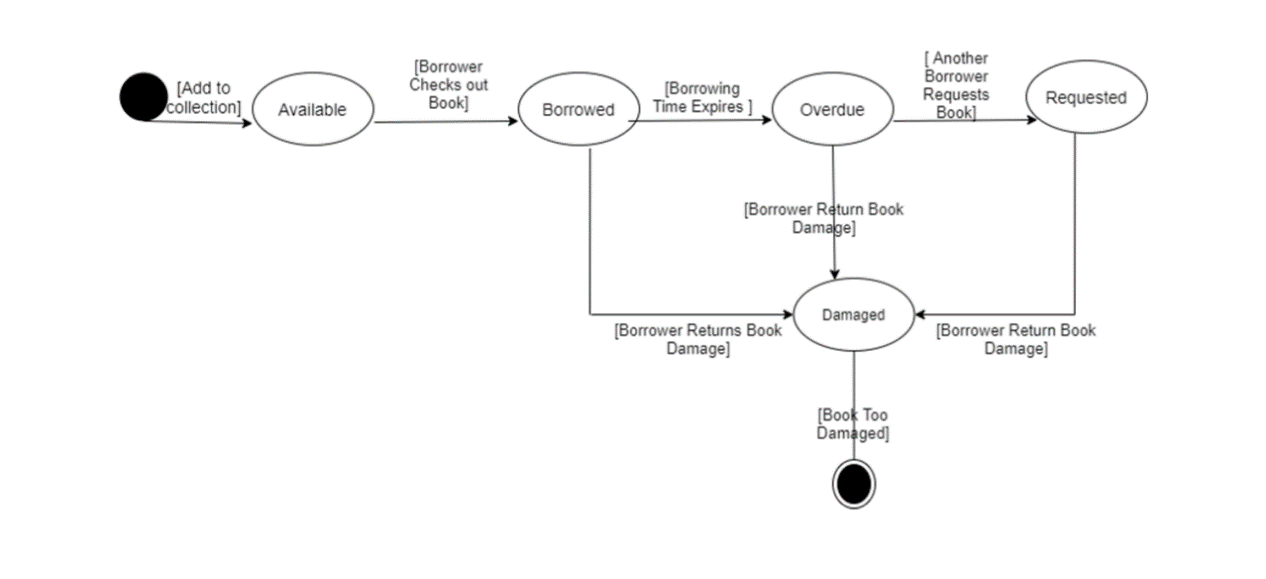




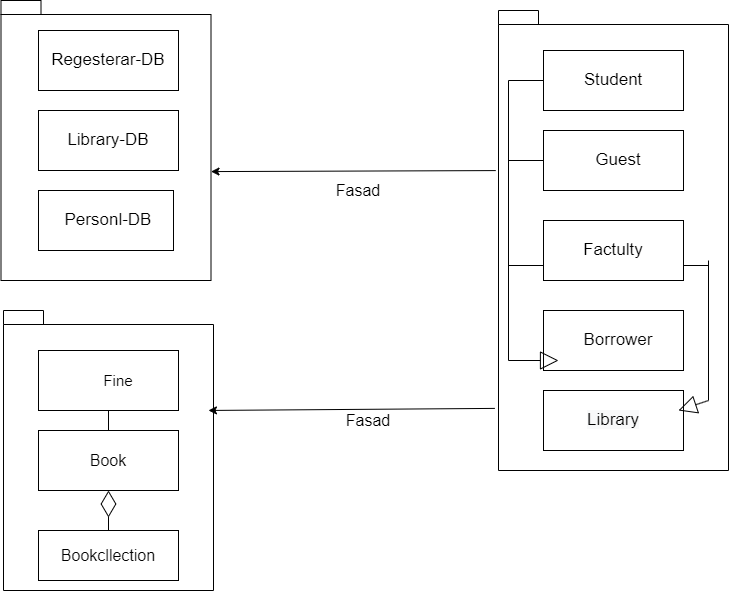
6. Class Diagram:



7. State chart Diagram:

****

8. Package Diagram:

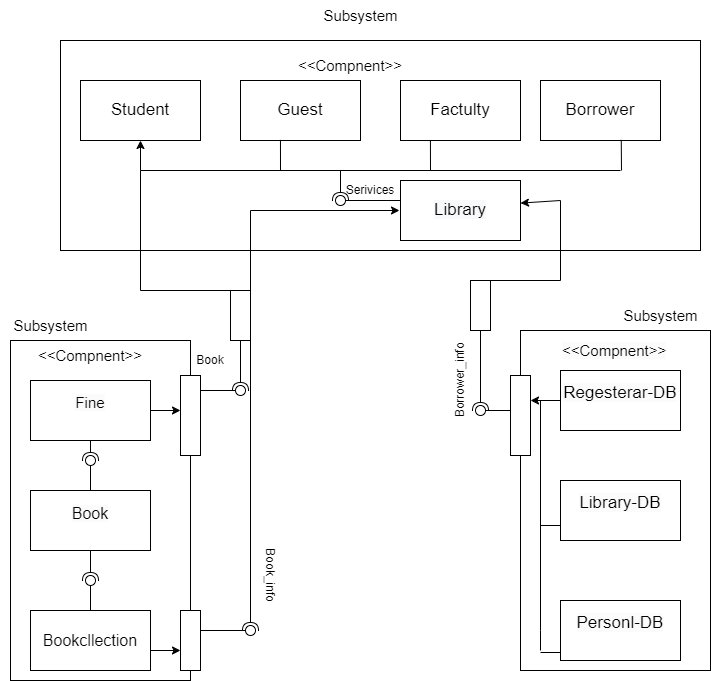


Book

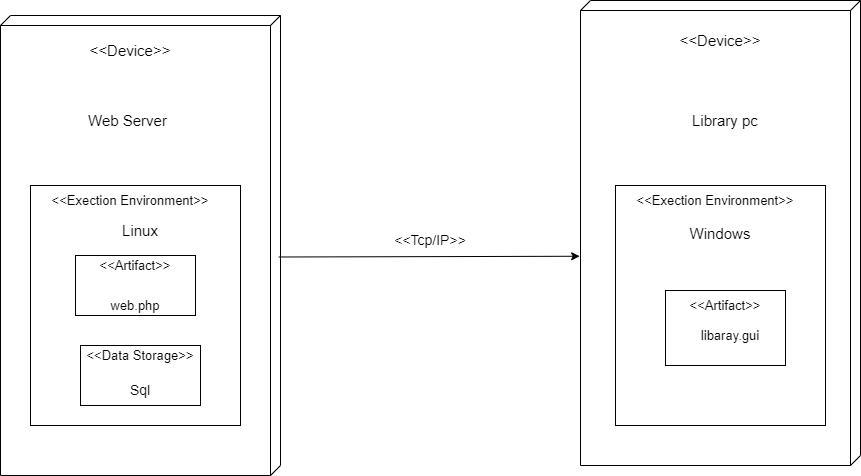
Book-DB\_ bBook\_

Borrower

9. Component Diagram:



10. Deployment Diagram:



11. HW/SW Mapping:

**(HW\_SW) Mapping:**

* **Processor: Intel Core i5 processors**
* **Memory:** **8GB of RAM**
* **I/O devises:** **Printer,** **Monitor,** **Barcode scanner,** **Keyboard and mouse**
* **OS: Windows**
* **Commination protocol:** **TCP/IP**
* **Data storage type: MySQL DB**

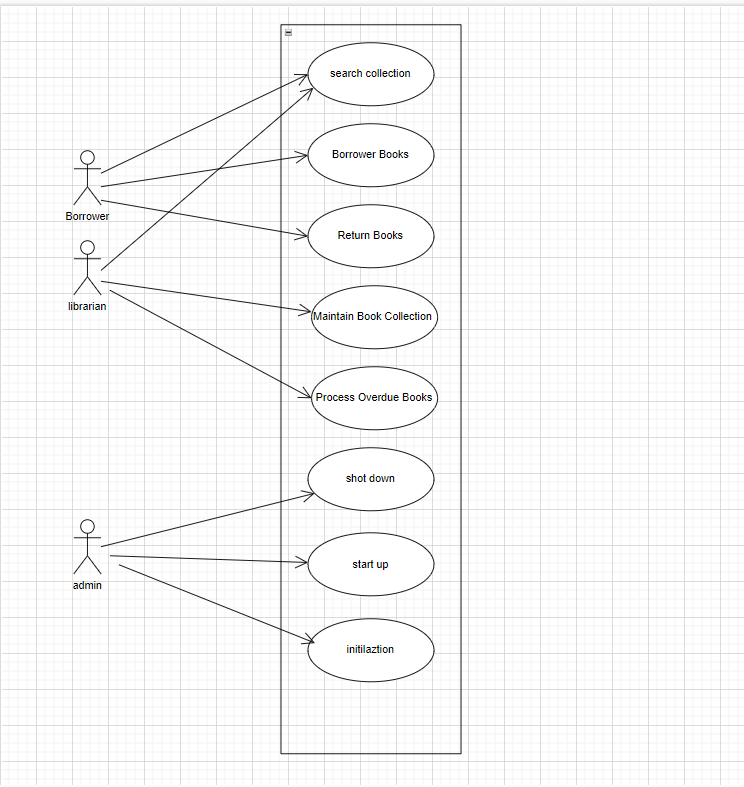
12. Access Matrix:

**Access Matrix**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Library\_Book\_Collection | Registrar DB | Personal DB | Library DB | Fine |
| Borrower | search\_for\_book() |  |  |  |  |
| Librarian | update-Book-Collection()  remove\_  Dammeged\_Book() | checks the\_ validity\_ of \_the id \_card() | checks the\_ validity\_ of \_the id \_card() | checks the\_ validity\_ of \_the id \_card() | checks\_overdue\_books \_fines()  increase\_Fine() |

13. Boundary condition:

**Boundary condition**



14. Control Flow

Flow control Decentralization

15. Concurrency

Concurrent Objects:

Regesterar\_DB

Personal\_DB

Library\_DB

Book Collection\_DB

16. Architecture:

In this structure, roles and responsibilities are divided between the client and the server, and each device performs specific functions.

The structure was chosen for several reasons, including:

1. Separation of data and logic: Dividing roles between server and client allows data to be separated from processes and logic. This facilitates software development and maintenance and increases reusability.

2. Centralized management: Centralized server management facilitates control, maintenance, and updating operations, and this provides better standardization and security

3. Security and protection: Security can be enhanced through client servers by separating sensitive data and logic from the client and executing it on the server.

4. Network distribution: The client server allows applications to be distributed over the network, allowing users to access the application from anywhere over the Internet

5. Scalability and customization: Customer applications can be developed to suit specific requirements and customizations can be made as needed

6. Cross-device presence: The client server can support a variety of devices and platforms, making the application available to a greater number of users.

7. Session and Connection Management: The client server allows users to connect to services, manage sessions, and interact easily

8. Providing an enhanced user experience: This architecture allows users to have a smooth and fast experience on different devices

9. Ease of development: Developers can easily develop cross-platform applications using client server

10. It is characterized by making the system performance better

17. OCL(Object Constraint Language )

**Borrower class**

Context Borrower: Student \_ Borrower () post:

Validate ID \_Guest\_Database ()

Context Borrower: staff \_ Borrower () post:

Validate ID \_ Personnel\_Database. ()

Context Borrower::bring\_book(book)post:

Provide\_idCard()

Context Borrower::request\_chked(book)pre:

unavailable\_book()

Context Borrower::return\_book()pre:

bring\_book(book)

**library class**

Context library::send\_remainder\_email(email)pre:

send\_email(email)

Context library::notify\_ Borrower()pre:

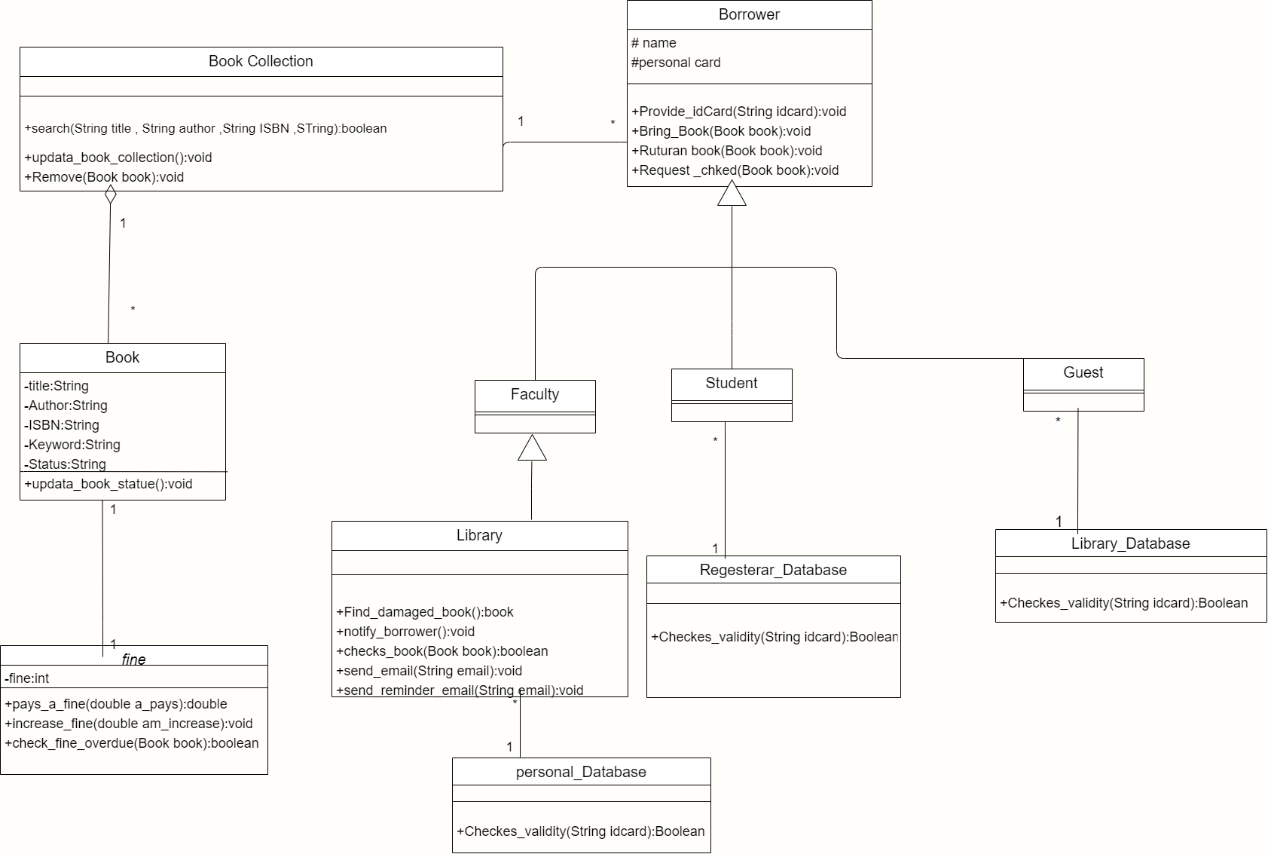
available\_book()

**Fine class**

Context **Fine** inv::self. Increase\_ fine(amount increase)>

Self. Pays\_a\_fine()

18. Desidn Class Diagram:



<<postcondition>>

provid\_idcard()

<<precondition>>

Bring\_book()