Design Document

Team: Soft_Engineers

Team Members:

1. Shiven Phogat (IMT2022050)
2. Samyak Jain (IMT2022071)
3. Shashwat Chaturvedi (IMT2022118)
4. Rishabh Dixit (IMT2022051)

Architectural Design

Client-Server Architecture:

The system is divided into two primary parts by client-server architecture: the client, which is responsible for the user interface, and the server, which is in charge of data management and business logic. A network is used to facilitate communication between the client and server.

Characteristics:

- Scalability: This design works well for large-scale applications since servers may be scaled independently to accommodate growing loads.
- Centralized Data Management: Since data is kept on the server, security and management can be done centrally.
- Thin Clients: Since most work occurs on the server, clients can be quite light.

Use Cases:

Web applications, email services, and online gaming platforms are just a few of the networked applications that rely on client-server architectures.

Reason for Using This Architecture:

1. Centralized Management : The centralized management simplifies data management and makes it easier to implement concurrent access of the resources

2. Scalability : The software can be scaled to accommodate as many library members as required

4. Ease of Maintenance: Changes, upgrades, or fixes on the server immediately reflect on all connected clients without requiring changes to each individual client.

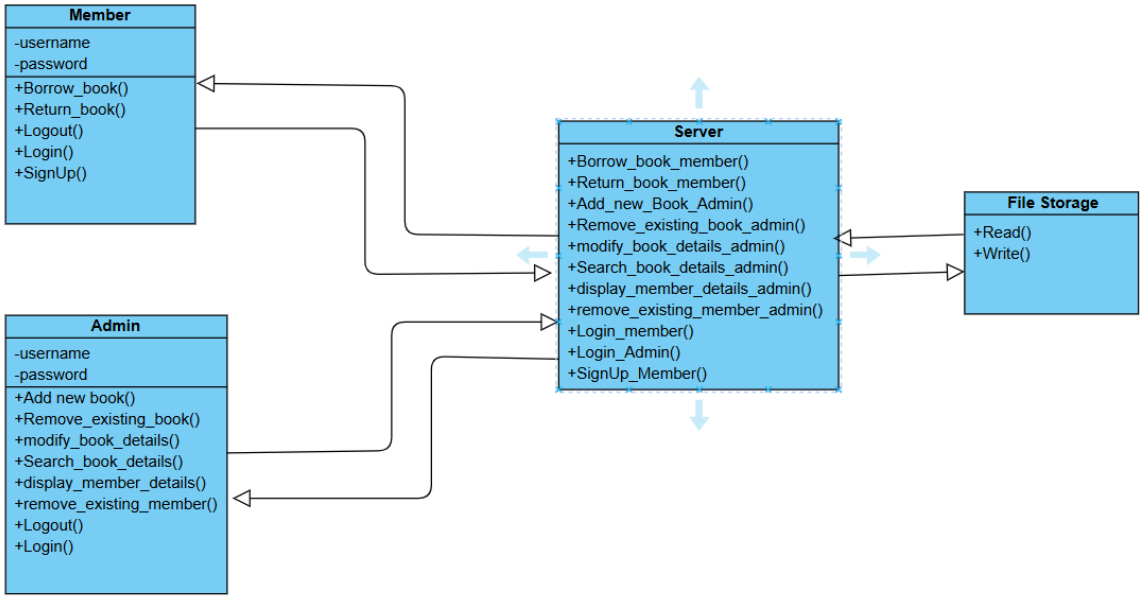
5. Enhanced Collaboration : Multiple members can view the availability of books at the same time . People can issue books without their requests conflicting from other members.

6. Reduced Data Redundancy : By centralizing data storage on the server, duplication of data across clients is minimized, ensuring consistency and saving storage space.

7. Fault Tolerance: Servers can incorporate robust fault-tolerant mechanisms (e.g. concurrency control , parallel processing), ensuring higher reliability compared to a peer-to-peer system.

8. Improved Performance : The server runs each client request on different threads. This ensures fairness for all users and better user response time for each user given the server device has enough processing power..

9. Support for Thin Clients: Lightweight clients can rely on the server for intensive processing and storage, making the client devices (i.e the member and the admin) require lesser processing power.



Resources: https://www.javatpoint.com/architectural-styles-in-software-engineering

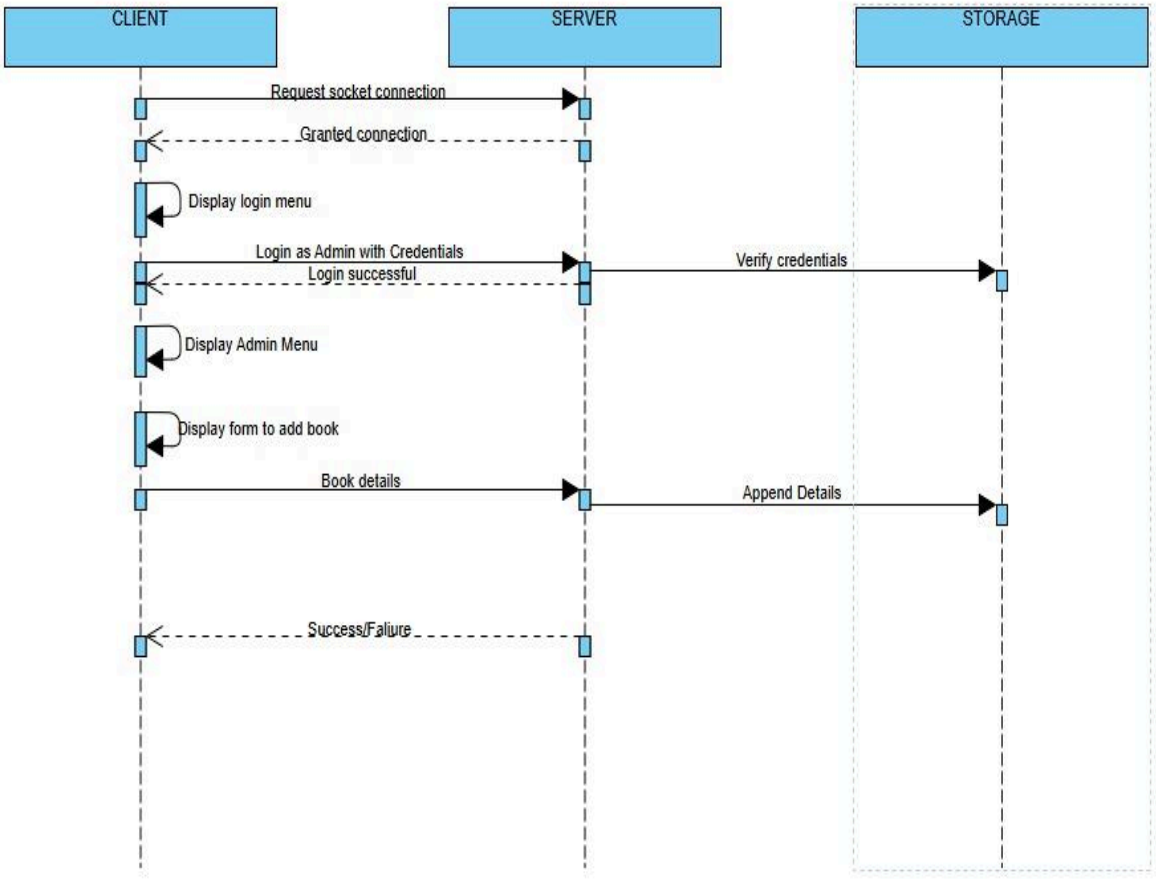
Sequence diagrams:

The following sequence diagrams can be used to visualize the interaction between the actors in a sequential order. There are 3 main actors involved for each functionality:

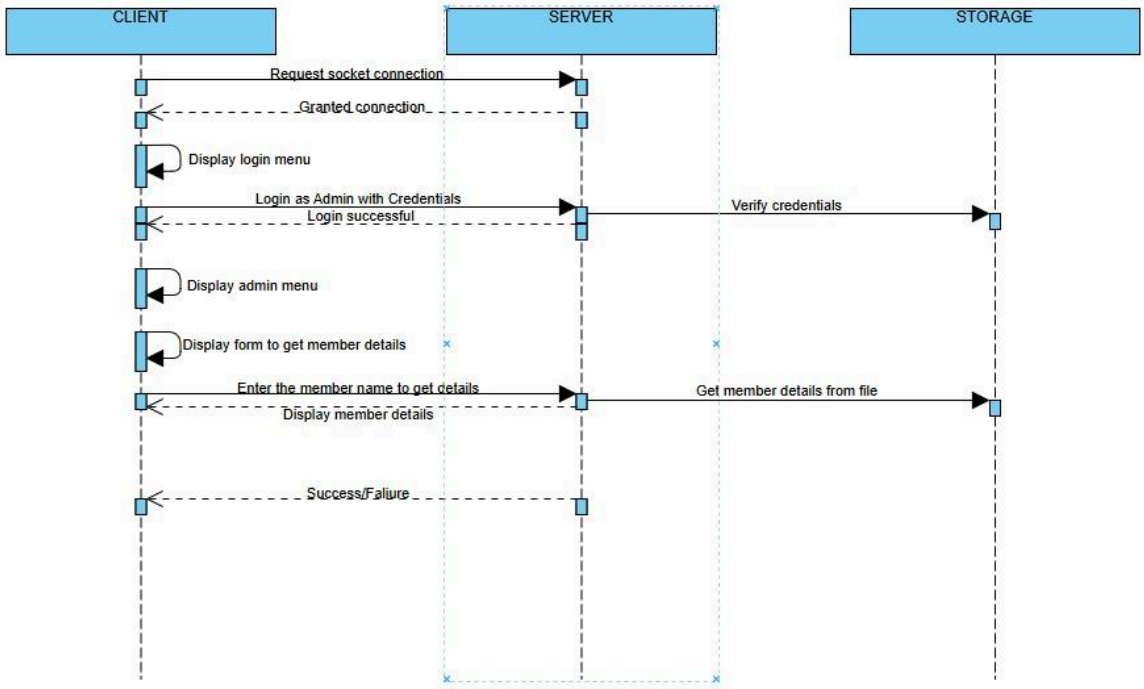
a) The library administrator / member
b) The server
c) File storage

All the menu methods can broadly be classified into two types - admin methods and member methods, with separate menus for each type.

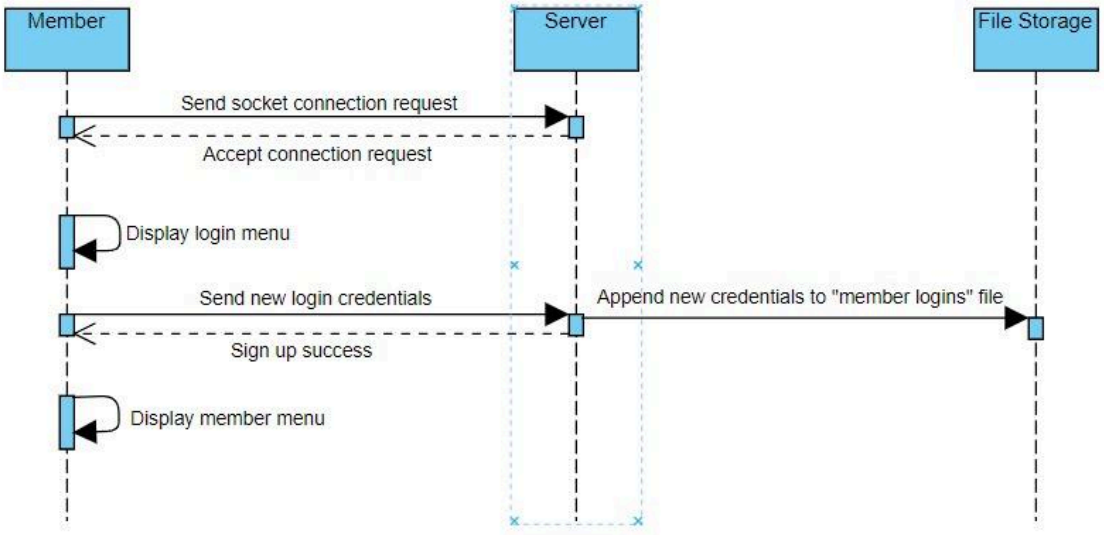
1. Adding a new book (through admin login):



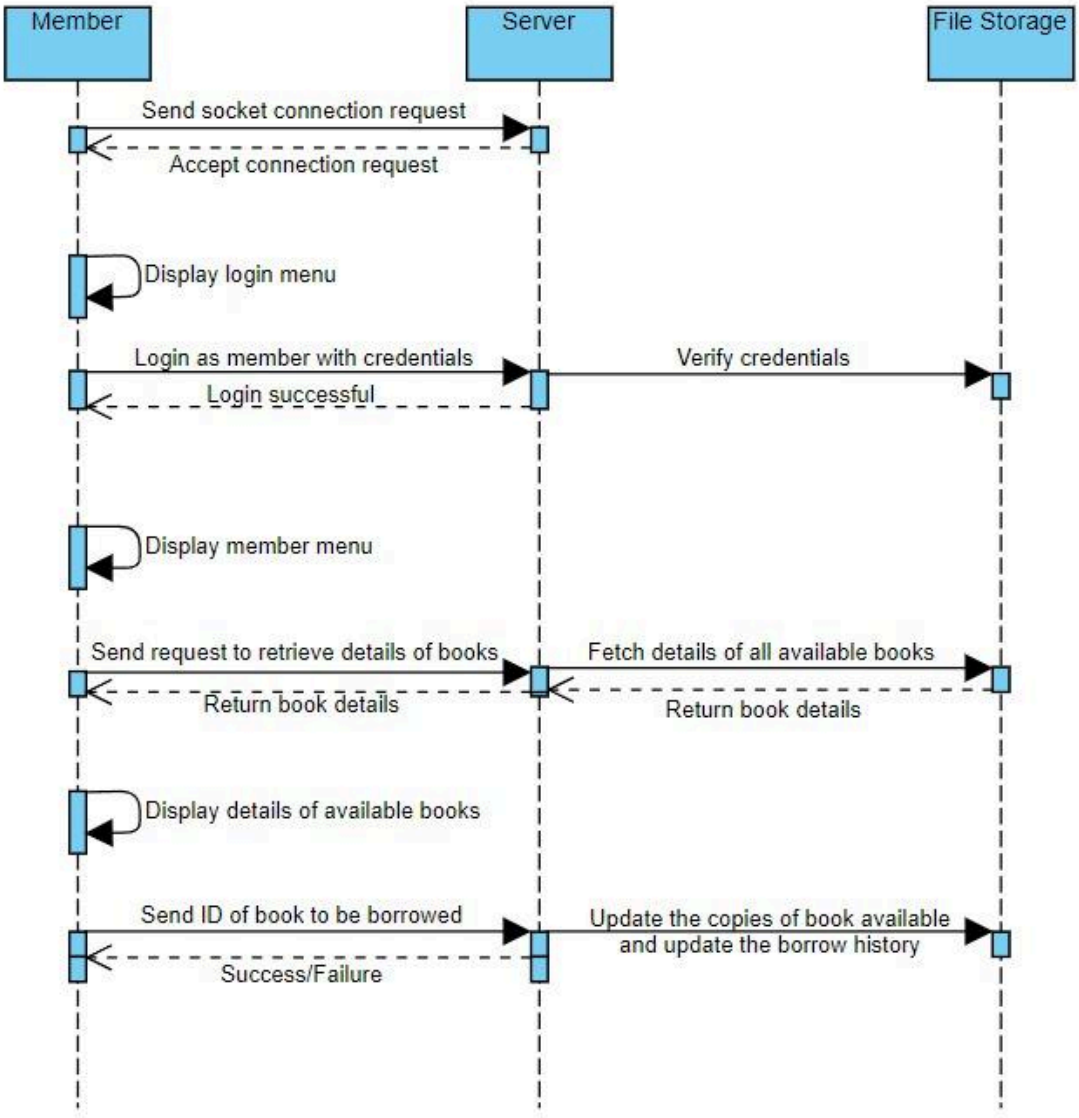
2. Fetching member details (through admin login):



3. Sign up as a new member:



4. Borrowing book (though member login)



Below is a list of all menu functionalities:

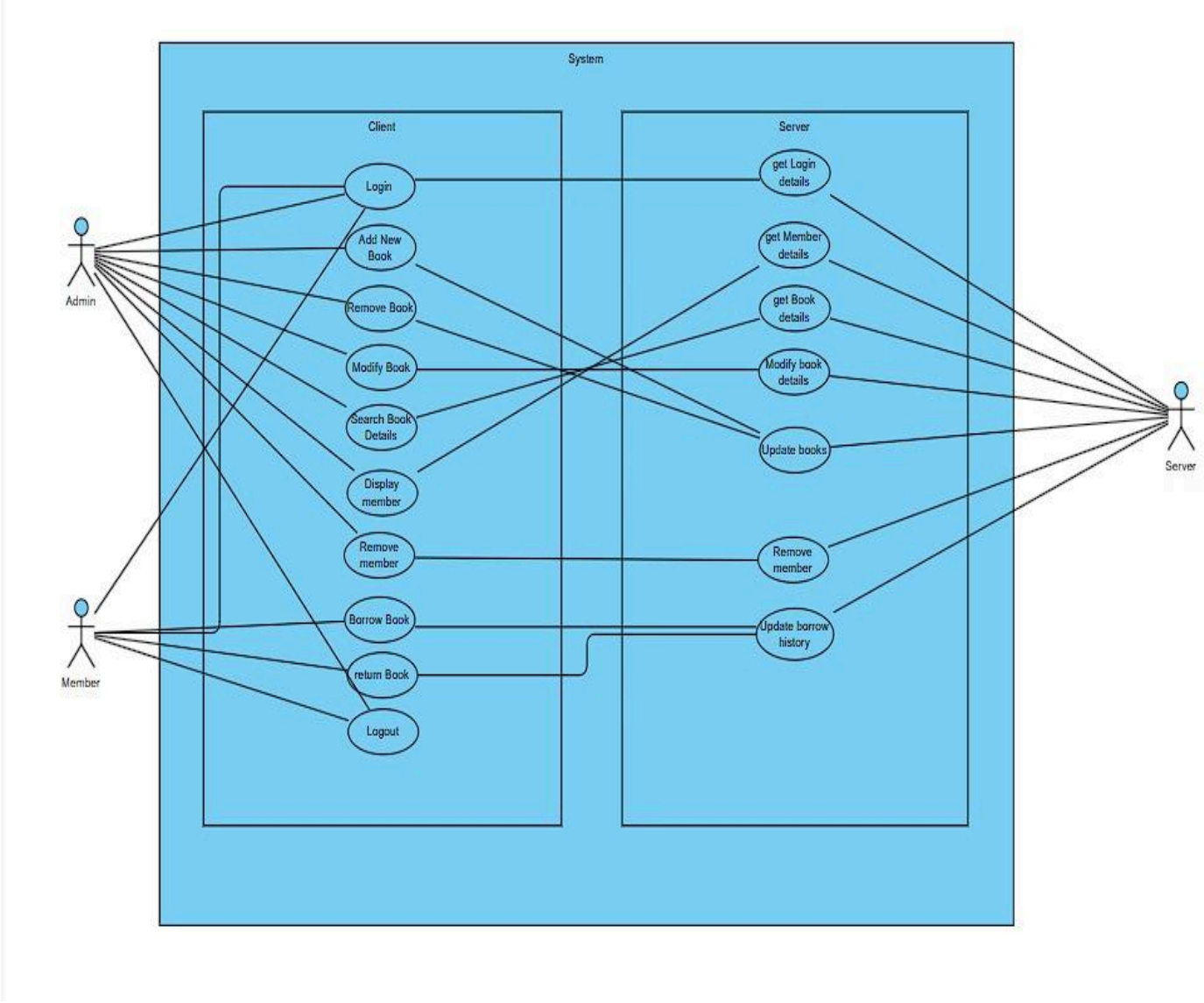
1. Main menu:
    a) Login as admin
    b) Login as existing member
    c) Sign up as new member

2. Admin menu:
    a) Add new book.
    b) Remove existing book.
    c) Modify book details.
    d) Search for book details.

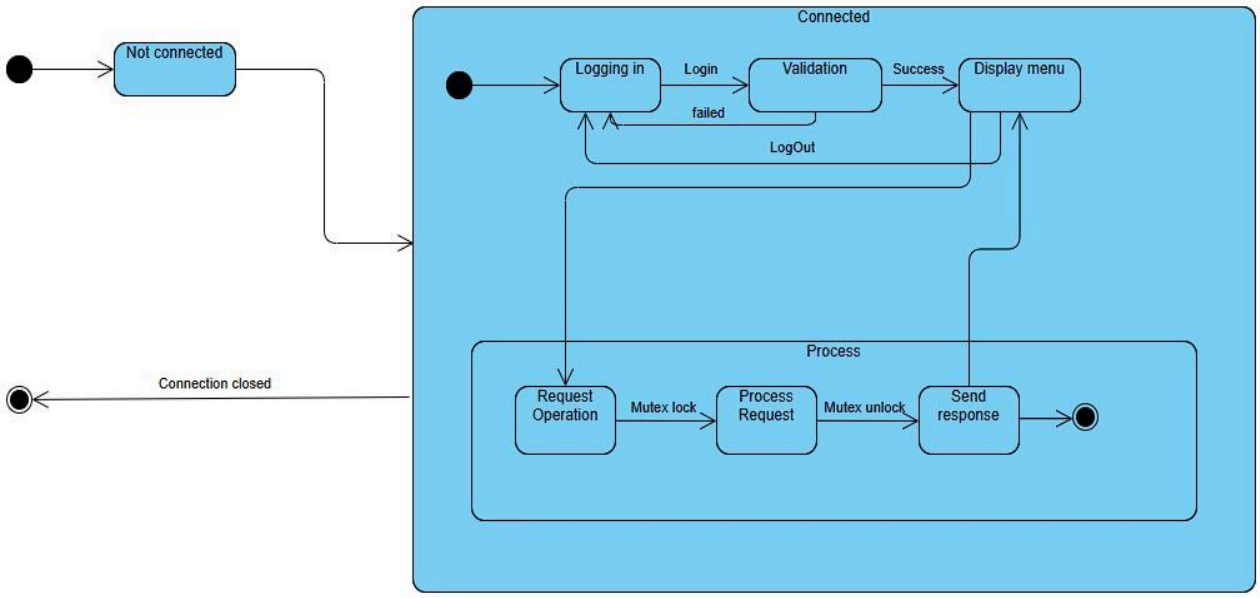
e) Display member details.
f) Remove existing member.
g) Logout.

3. Member menu:
    a) Borrow book
    b) Return book
    c) Logout

Use Case Diagram:



State Diagram:



Low level details:

Strategies used:

1. **Socket Programming:** All interactions between client and server (inter-process communication) happen through socket connections. The server, whenever active, keeps waiting for connection requests, enabling multiple client connections at a time.
2. **Multithreading:** Once the server accepts a new connection, a new thread is created and all the functionalities are carried out through a handler function. This ensures that several client requests can be handled at a time.
3. **File locking:** Mutex-based file locking mechanism ensures concurrency control. No two server operations can have access to the same resource at a time.

Major components:

1. **File storage:**

All data has been stored in text files. The server has access to all the data files, but only one thread can have control over a file at a time.
There are mainly 4 types of files:

a) logins.txt: This stores credentials of all existing library members.
b) admin_login.txt: This stores the credentials of the library administrator.
c) books.txt: This stores all the details of the books, such as book id, name, author, count etc.
d) borrowed_books.txt: This stores the history of all the borrowings. Whenever a book is returned the entry is removed.

2. Server and client files:

a) library_server.c: consists of all the functionalities to be performed by the server end. It receives all the details of any operation to be performed (including the operation id and other parameters, if needed), and calls the corresponding handler functions to process the request. Before accessing any text file, a mutex lock is applied and released once the file operations are over.

b) library_client.c: displays all the menu features available, and allows the user to send requests to the server for performing various tasks. Depending on the type of CRUD operation, the server either sends details of books/members or just a simple success/failure message.

   This client-server architecture is capable of handling all kinds of requests mentioned alongside the sequence diagrams.