Library Management System

Overview: The library management system is a client-server application designed for managing book records. Users log in through a GUI, which, upon successful authentication, launches the BookClient. This client allows users to add books and view the list of books by sending requests to the server. The server processes these requests, interacts with a MySQL database to store or retrieve book details, and responds to the client. Key features include user authentication, a user-friendly GUI, client-server architecture, and database integration, providing a simple and efficient way to manage a library's book inventory.

Archietcture:

Class Descriptions

1. **Login**: Handles user authentication with a GUI, validating the username and password.
2. **BookClient**: Provides a GUI for adding books and listing all books, communicating with the server.
3. **Server**: Listens for client connections, handles book addition and retrieval, and interacts with a database.
4. **Book**: Represents a book entity with properties like id, title, author, and availability.

Programflow:

1. **Login** Class Initialization:

A `**Login**` class is run that initializes a login GUI with fields for username and password, and a login button.

2. User Enters Credentials:

The user fills in the username and password in the corresponding fields and clicks the login button.

3. Credential Validation:

- The inputted credentials are matched with the predefined username and password, "admin" and "password".

- In case of a match, the success message is updated and the login window is closed.

- On unsuccessful login, an error message will be shown.

4. Starting **BookClient**:

- On successful login, the `**BookClient**` class is instantiated, which starts the Library Management System GUI.

5. **BookClient** GUI Initialisation:

The **BookClient** GUI encapsulates fields for the title and the author of the book and buttons to add a book and list all books.

• There is also an area on the GUI that will serve as a display area for the listing of books.

6. Adding a Book:

When the "Add Book" button is clicked, this triggers the execution of the *addBook* method.

- The entered title and author are retrieved, and these details are forwarded to the server over a socket connection.

- Server responds with an acknowledgement.

7. Listing Books:

- Clicking the "List Books" button shall trigger the *`listBooks()`* method.

- The method shall send a request to the server to obtain the list of all books.

- The server responds with the list of books that is set in the text area of the `**BookClient**` GUI.

8. Initialising the Server:

- An instance of the `**Server**` class is run, initializing a server socket. It starts waiting for new client *getConnection* on port number 12345.

9. New Client Connections:

- Upon the connection of a client—that is, `BookClient\"—, the server will accept it and generate a new thread for this very client.

10. Processing Client Requests:

- The server shall first read the command from the client, either "ADD_BOOK" or "LIST_BOOKS".

- For "ADD_BOOK":

- It reads the title and author from the client and calls `*addBook()*` to insert these details into the database.

- For "LIST_BOOKS":

  - The server makes a call to `*listBooks()*` to retrieve all book records from the database and send it to the client.

11. Database Interaction:

- The server connects to a MySQL database to both add new book entries and fetch the list of books.

  - `*addBook()*` - A new record for a book is inserted.

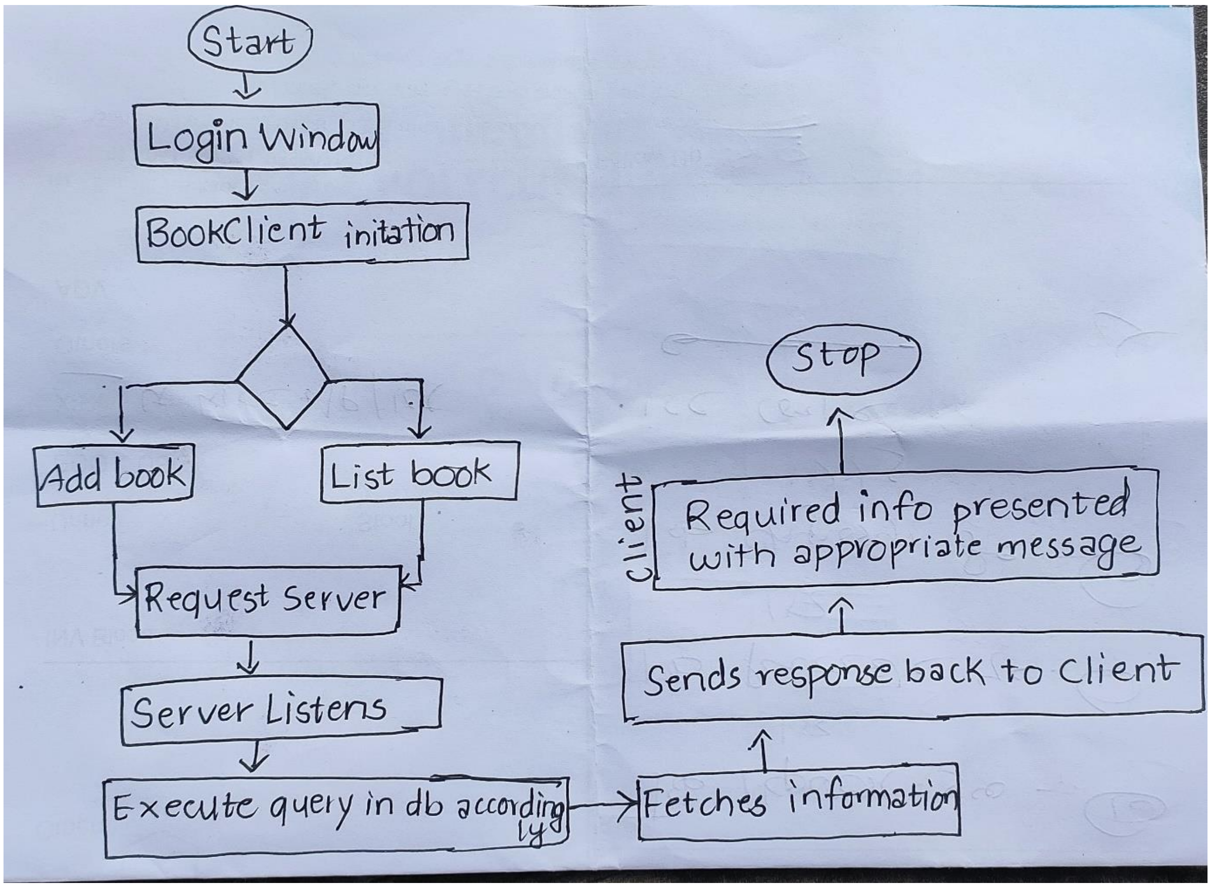
  - `*listBooks()*` returns all the records for the books and expresses them as a list of `**Book**` objects.

12. **Book** Class:

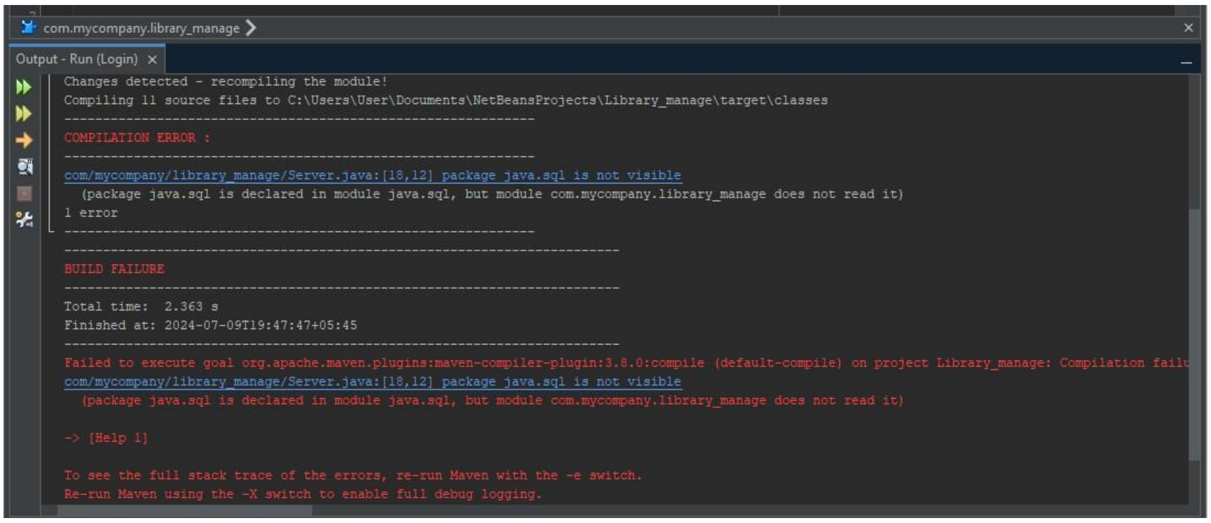
- The class `**Book**` defines the structure of the book entity with fields for id, title, author, availability.

  - It has methods to get and set these fields and a `*toString()*` for a readable form of a book.

Diagram:



Result:



```
Changes detected - recompiling the module!
Compiling 11 source files to C:\Users\User\Documents\NetBeansProjects\Library_manage\target\classes
-------------------------------------------------------------
COMPILATION ERROR :
-------------------------------------------------------------
com/mycompany/library_manage/Server.java:[18,12] package java.sql is not visible
  (package java.sql is declared in module java.sql, but module com.mycompany.library_manage does not read it)
1 error
-------------------------------------------------------------
------------------------------------------------------------------------
BUILD FAILURE
------------------------------------------------------------------------
Total time:  2.363 s
Finished at: 2024-07-09T19:47:47+05:45
------------------------------------------------------------------------
Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.8.0:compile (default-compile) on project Library_manage: Compilation failu
com/mycompany/library_manage/Server.java:[18,12] package java.sql is not visible
  (package java.sql is declared in module java.sql, but module com.mycompany.library_manage does not read it)

-> [Help 1]

To see the full stack trace of the errors, re-run Maven with the -e switch.
Re-run Maven using the -X switch to enable full debug logging.
```

Explanation:

I ran the program multiple of times **but package visible problem** set me behind . Despite searching hours for the solution it does not fix.

I have now understood how a server executes the commands sent by client and got familiar with the basic outline of an application program. I learnt to organize project files and add libraries. I also understood how server handles requests and how it executes query In database.