

ASSIGNMENT

Course Code	CSE302A
Course Name	Network Programming and Simulation
Programme	B.TECH
Department	CSE
Faculty	FET

Name of the Student	AISHWARYA
Reg. No	17ETCS002015
Semester/Year	6 TH SEM/ 3 RD YEAR
Course Leader/s	Dr. Rinki Sharma

Declaration Sheet			
Student Name	AISHWARYA		
Reg. No	17ETCS002015		
Programme	B.TECH	Semester/Year	6 TH SEM/ 3 RD YEAR
Course Code	CSE302A		
Course Title	Network Programming and Simulation		
Course Date		to	
Course Leader	Dr. Rinki Sharma		
<p>Declaration</p> <p>The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly.</p>			
Signature of the Student		Date	19/04/2020
Submission date stamp (by Examination & Assessment Section)			
Signature of the Course Leader and date		Signature of the Reviewer and date	

Faculty of Engineering and Technology							
Ramaiah University of Applied Sciences							
Department		Computer Science and Engineering		Programme	B. Tech. in CSE		
Semester/Batch		6 th Semester/2017					
Course Code		CSE302A		Course Title	Network Programming and Simulation		
Course Leaders		Dr. Rinki Sharma and Mr. Narasimha Murthy K. R.					
Assignment							
Reg. No.		17ETCS002015		Name of Student	AISHWARYA		
	Marking Scheme				Marks		
					Max Marks	First Examiner	Moderator
Question 1	1.1	Java Function Calls for Socket Programming			2		
	1.2	Difference between Socket Programming using C and Java			2		
	1.3	Conclusion			1		
		Question 1 Max Marks			5		
Question 2	2.1	Design Specifications, Functional and Non-functional Requirements			3		
	2.2	LMS Design with Flowcharts/Algorithms			4		
	2.3	Implementation			6		
	2.4	Test Results			2		
		Question 2 Max Marks			15		

Assignment Marks Tabulation				
Component-1 (C) Assignment	First Examiner	Remarks	Moderator	Remarks
Total Marks (out of 20)				
Total Marks (out of 10)				
<div>Signature of First Examiner</div> <div>Signature of Moderator</div>				

Question 1

Socket programming is two systems communicating with one another, generally TCP and UDP.

A socket is one end point of the two-way communication link between two programs running on the network. The socket is bound to a port number so that, the TCP layer can identify the application that data is destined to be sent.

Java Socket programming is used for communication between the applications running on different JRE. Java Socket programming can be connection-oriented or connection-less. `Socket` and `ServerSocket` classes are used for connection-oriented socket programming and `DatagramSocket` and `DatagramPacket` classes are used for connection-less socket programming.

`Java.net.Socket` and **`java.net.ServerSocket`** are java classes that implements socket and socket server.

Java provides a collection of classes and interfaces that take care of low-level communication details between the client and the server.

These are mostly contained in the *java.net* package

```
import java.net.*;
```

```
import java.io.*;
```

a server runs on a specific computer on the network and has a socket that is bound to a specific port number.

```
ServerSocket serverSocket = new ServerSocket(6666);
```

The server just waits, listening to the socket for a client to make a connection request. This happens in the next step:

```
Socket clientSocket = serverSocket.accept();
```

When the server code encounters the *accept* method, it blocks until a client makes a connection request to it.

the server *accepts* the connection. Upon acceptance, the server gets a new socket, ***clientSocket***, bound to the same local port

the new *Socket* object puts the server in direct connection with the client, the output and input streams to write and receive messages to and from the client respectively:

```
PrintWriter out = new PrintWriter(clientSocket.getOutputStream(),true);
```

```
BufferedReader in
```

```
= new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
```

From here, the server is capable of exchanging messages with the client endlessly until the socket is closed with its streams. For every new client, the server needs a new socket returned by the *accept* call. The *serverSocket* is used to continue to listen for connection requests while tending to the needs of the connected clients.

The client must know the hostname or IP of the machine on which the server is running and the port number on which the server is listening.

To make a connection request, the client tries to rendezvous with the server on the server's machine and port:

```
Socket clientSocket = new Socket("127.0.0.1", 6666);
```

The client also needs to identify itself to the server so it binds to a local port number, assigned by the system, that it will use during this connection. The above constructor only creates a new socket when the server has *accepted* the connection, then obtain input and output streams from it to communicate with the server:

```
PrintWriter out = new PrintWriter(clientSocket.getOutputStream(),true);
```

```
BufferedReader in
```

```
= new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
```

bind(SocketAddress bindpoint) : Binds the socket to a local address.

connect(SocketAddress endpoint) : Connects this socket to the server.

connect(SocketAddress endpoint, int timeout) : Connects this socket to the server with a specified timeout value.

getInetAddress() : Returns the address to which the socket is connected.

getReceiveBufferSize() : Gets the value of the SO_RCVBUF option for this Socket, that is the buffer size used by the platform for input on this Socket.

Connects this socket to the server.

public void connect(SocketAddress endpoint) throws IOException

Returns the remote port number to which this socket is connected.

public void bind(SocketAddress bindpoint) throws IOException

public InetAddress getInetAddress()

public int getPort()

Difference between Socket Programming using C and Java

The difference between the socket programming using C and Java is:

In C programming language at the server side, for creating socket the command is **int sockfd = socket(domain, type, protocol)**, which specifies the type of socket descriptor, whether the socket is connection less or connection oriented and bind function is used after the socket is created to bind the socket to the address and port number specified in the Sock_addr whereas in java for socket creation, the statement used is **public static int port = Integer_value;**
server = new ServerSocket(port);

In java the binding of the socket to the address is not required as it is done by the socket creation call.

In java listen function call is not present as compared to C as java internally binds and keeps listening for the client's request.

At the clients side, in C as well as in java the socket is created, but in C programming language to connect the socket to the file descriptor specified by the address, connect() function is used whereas in java no such function is used as when the socket is created in java it extracts the IP address of the server and also connects the client to the server.

CONCLUSION:

For client-server-based communication using socket programming in java programming language as compared to that of the C programming language is more effective as java programming language provides application of the program in an effective way, as Java itself affords a nice API for socket programming called Java, where network module can be easily written. As, java is an object-oriented programming language, it uses classes, here programs are designed using the sockets using the concepts of objects that interact with real world whereas C programming language is a procedural programming language where the concept is based upon calling of the procedure. Procedures known as routines, subroutines or functions simply consist of a series of computational steps to be carried out.

In socket programming using Java addition of new data and function is quite easy as compared to that of C programming language. Java provides overloading as well as here data is important than function whereas in the C programming language function is more important. The socket-programming through java programming language is for the real world as it can be easily accessible to the GUI for the used whereas C programming language is for virtual world. Hence, it is concluded that socket programming in java is more efficient than that of the socket programming in C programming language.

Question 2

1. Introduction

The problem given, a library management system is to be developed based on client server communication using socket-programming.

The LMS should be able to handle simultaneous request to issue books that is the system developed can be accessed and functioned by one or more clients at a time. The cases considered are: -

Multiple copies of same books available, reserving books not available, adding books details to the database which are not stored in database, check availability, no of maximum books borrowed by the client, date of issue and return of the books issued, a return window to return the books, to view all the details of all the books stored in the library database, and to access the book details of the books reserved. Here, client server-based communication is used using socket programming in java programming language. The GUI is to be developed for the client program using the java. Swing and connected to the server, the database used here to store the details used here is MySQL database.

The developed system is able to handle simultaneous request to issue books, that is books can be issued by one or more clients at the same time. There is also another option provided if the book is not available the books can be reserved. Once the book is reserved, the client will be informed about the availability of book as well as the date of issue and return. Here, the system developed for the library management where, in the portal various options will be provided such as to issue a book, check availability of the books, return as well as reserve books, show all the available books and the books issued as well as the portal is having a function to add new books. Here, the client will be able to issue book by entering book ID, student ID and student name, whereas for return the client needs to provide the book ID. The client in the portal developed is able to check the number of books available just by entering the book name. If the book needed by client is not available, then the client will be able to reserve the book using the student Id, name and book name. In the portal the client is able to check all the books stored in the database of the library, that is all the books available in the library. The client is able to receive information about all the available books in the library. In the GUI developed there is an option to add books to the library database.

2. Design specifications along with functional and non-functional requirements

The functional requirements for the Library Management System:

FR1: The system developed should be able to add books to the library database.

FR2: The library management system developed should be able to display all the books available in the library to issue to the user.

FR3: The library management system developed should be able to display all the books stored in the library database to the user.

FR4: The system developed should be able to issue book to the user, if the book user wants to issue is available in the library or be issued.

FR5: The system should be able to reserve books to the user, if the books to be issued by the user is not available in the library using the user credentials.

FR6: The system should be able to generate date of issue and date of return to the user, for the book(s) issued.

FR7: The system should be able to view all the books issued by the user with the validity of the book along with the book credentials.

FR8: The system developed should be able to allow the user to return the books issued.

NON-FUNCTIONAL REQUIREMENT:

The non-functional requirements used here are: -

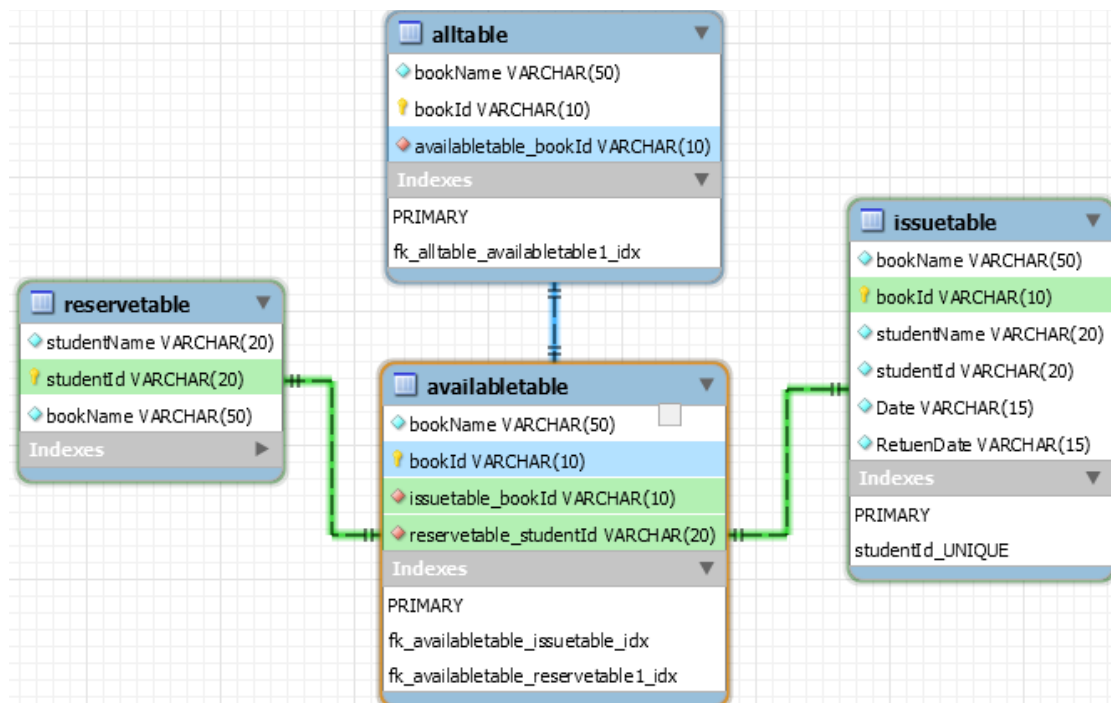
NFR1: the system developed must be able to handle multiple users.

NFR2: the system developed must be portable.

NFR3: the system developed must be scalable, reliable, serviceable, security must be provided.

3. LMS design, highlighting the messages between client and server along with the database structure

Database Structure



The above figure shows the structure of the database implemented in MySQL. All the entities in the diagram forms a table in database with the given attributes it with along with specification about the primary, unique and foreign key.

Client Server messages

The client is sending MySQL queries to the server in form of string. The server passing the query to the database. The database is replying the server with result. The server then transmits the result to the client then client to the calling function. This shows that the client and server do not have any separate message rather than the MySQL queries which for the client and server are just string and the MySQL queries are shown in the implementation part.

ALGORITHM for the Library Management System Designed:

CLIENT-SIDE:

Step1: socket is created by using object socket class, with host address and port-number.

```
Socket socket = new Socket(host.getHostName(), 4024);
```

Step2: create object "ois" ObjectOutputStream to send message from the client to the server.

Step3: to retrieve data from the database, the SQL queries from the client side is sent to the server.

Step4: read the data sent from the server to the client and store the data received and display.

Step5: close the objects "ois" and "oos" and then close the socket.

SERVER-SIDE:

Step1: connection between database and server is established, an object statement is created to execute queries.

```
Class.forName("com.mysql.cj.jdbc.Driver");  
Connection con = DriverManager.getConnection  
("database_name","user_name","password");  
Statement st = con.createStatement();
```

Step2: server socket is created using ServerSocket class.

```
ServerSocket server = new ServerSocket();
```

Step3: while loop is created, while true then endless loop. It consists of the statements which are required to be executed whenever a client program wants to communicate with server.

- 1 Accept the requests coming from the client by creating a Socket and accepting the requests using accept function.

```
Socket socket = server.accept();
```

- 2 Create an object of ObjectInputStream as "read" to read incoming messages from the client program.

```
ObjectInputStream ois = new ObjectInputStream(socket.getInputStream());
```

- 3 Read the incoming data from client and store it in a String variable.

```
String message = (String) read.readObject();
```

- 4 Create an object of ObjectOutputStream as "write" to write messages from the server to the client program.

```
ObjectOutputStream write = new ObjectOutputStream(socket.getOutputStream());
```

- 5 After execution of the query, if database returns any output store it in a String variable, send it to the client.

Step4: Close the server.

4. Implementation, documented using well-commented code snippets

Client-program:


```
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.InetAddress;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Vector;
import javax.swing.table.DefaultTableModel;

public class lib extends javax.swing.JFrame {
    static InetAddress host;
    static Socket socket = null;
    static ObjectOutputStream oos = null;
    static ObjectInputStream ois = null;
    static int port=4024;
    static String output="";
```

Fig.1 client program

Since in the LMS require to access multiple functions from the client side to the server side from different functions such as issue book, return book etc. Here for the basic implementation of object of classes like socket, ObjectOutputStream etc. are to be defined in each and every function. Hence, objects are defined with global scope so to be used without defining every time.

```
public static void main(String args[]) throws IOException {
    host = InetAddress.getLocalHost();
```

Fig.2 client program

Since object of any class can't be "initialised" in the global scope and the main function is called after the declaration of global variable in java. Swing after which the GUI functions. Hence, host is initialized in the main function.

```
public void sendCommandWithRecieveData(String data){
    output="";
    try {
        socket = new Socket(host.getHostName(), port);
        oos = new ObjectOutputStream(socket.getOutputStream());
        System.out.println("Sending request to Socket Server");
        oos.writeObject(data);
        ois = new ObjectInputStream(socket.getInputStream());
        String message = (String) ois.readObject();
        System.out.println("Message: " + message);
        output=message;
        ois.close();
        oos.close();
    } catch (Exception ex) {
        System.out.println("Error : " + ex);
    }
}
```

Fig.3 client program

Here object of socket is created with arguments InetAddress and port number where InetAddress is obtained by using getHostName function using InetAddress object named as host. This function is used to send request to the socket server and read the input from the object of objectInputStream. Here, "oos" is object of objectInputStream and "ois" is object of ObjectInputStream. Followed by writing input to the server using writeObject and reading the result send by the server using readObject and storing it in global variable named as "output". Then the objects of objectInputStream and objectInputStream are closed using "close" function

SERVER PROGRAM

```
package aishwaryaserver;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class AishwaryaServer {
    static ServerSocket server;
    static Statement stmt;
    static int port=4024;
    static String nextColumn="\t";
    static String nextRow="\n";
    static Connection con;

    static String data="";
    public static void main(String args[]) throws IOException, ClassNotFoundException, SQLException{
        try{
            Class.forName("com.mysql.cj.jdbc.Driver");
            con=DriverManager.getConnection("jdbc:mysql://localhost:3306/network","root","1234");
            stmt= con.createStatement();
        }
        catch(Exception e){
            System.out.println(e);
        }
        server = new ServerSocket(port);
```

Fig.4 server code

The above snippet shows the server socket and port are defined and initialized in globally. Necessary imports are also used such as ServerSocket, Socket, etc. In the main function ServerSocket is defined. In java, the bind function is present and shown below :

SocketAddress endpoint = new InetSocketAddress(port);

server.bind(endpoint);

But in java if port number is provided while creating object of ServerSocket class as an argument as used above then there is no need of bind function as ServerSocket already perform the functionality of the bind function.



```
while(true){
    System.out.println("Waiting for the client request");
    Socket socket = server.accept();
    System.out.println("accepted");
    ObjectInputStream read = new ObjectInputStream(socket.getInputStream());
    data = (String) read.readObject();
    ObjectOutputStream write = new ObjectOutputStream(socket.getOutputStream());
    System.out.println("Message Received: " + data);
    StringBuilder builder = new StringBuilder(data);
    if (data.charAt(0)=='-') {
        builder.deleteCharAt(0);
        stmt.executeUpdate(builder.toString());
    } else {
        data="";
        ResultSet rs=stmt.executeQuery(builder.toString());
    }
}
```

Fig.5 server code

The above figure shows accept function and some other statements are in loop and the loop is having condition which is running till infinity using “true” as condition. The importance of the function is that the accept function can accept one request at a time but the requirement of the LMS is to server should accept multiple requests where the loop is helping it to deliver the required functionality.

The point to address is that there is no listen function used. It is not present because of the feasibility of the language java as the accept function also performs the functionality of the listen function.

After this, connection to the server is established followed by the creation of objects for ObjectInputStream and ObjectOutputStream as read and write for reading input coming from the client and writing result to the client as output. For reading, function readObject is used using read object and stored in variable names “data” with string as datatype.

```
while(rs.next()){
    for (int i = 1; i <= rs.getMetaData().getColumnCount(); i++){
        data+=rs.getString(i)+nextColumn;
        data+=nextRow;
    }
    System.out.println("---"+data);
    write.writeObject(data);
}
con.close();
read.close();
write.close();
socket.close();
if(data.equalsIgnoreCase("exit")){
    break;
}
}
server.close();
}
```

Fig.6 server code

The received input in the variable “data” is divided into two categories on the bases of the first character using if-else statement. If the first character is “-”, it represents that the sql query received is of update type and does not require any output, so there is no need to use writeObject function, using it will be wastage of the bandwidth.

If the first character is not “-”, then the received sql query from the client shows that the client is wanting an output from the server and is waiting for it.

After that, the connection to the server, object of ObjectInputStream and object of ObjectOutputStream is closed followed by closing of socket but the server is still not closed. It will be closed when any client is requesting it to close by using “exit” string in the input.

MySQL queries used to retrieve data are: -

1. Check availability of books

```
private void CheckAvailabilityButtonActionPerformed(java.awt.event.ActionEvent evt) {
    String data="select count(*) from availbletable where BookName='"+TextBookNameCheck.getText()+"'";
    sendCommandWithRecieveData(data);
}
```

2. Show all books available

```
private void AllBookButtonActionPerformed(java.awt.event.ActionEvent evt) {
    String Data ="select * from alltable";
}
```

3. Issue books

```
private void IssueButtonActionPerformed(java.awt.event.ActionEvent evt) {
    String Data="-insert into issuetable values('"+TextBookNameIssue.getText()+"','"+TextBookIdIssue.getText()+"','"+
        +TextStudentNameIssue.getText()+"','"+TextStudentIdIssue.getText()+"','"+TextDateIssue.getText()+"')";
}
```

4. Return book

```
private void returnButtonActionPerformed(java.awt.event.ActionEvent evt) {
    String Data="-insert into availbletable values('"+TextBookNameReturn.getText()+"','"+TextBookIdReturn.getText()+"')";
    sendCommandWithNoRecieveData(Data);
    Data="-delete from issuetable where BookId='"+TextBookIdReturn.getText()+"'";
}
```

5. Reserve book

```
private void reserveButtonActionPerformed(java.awt.event.ActionEvent evt) {
    String Data="-insert into reservetable values('"+TextStudentNameReserve.getText()+"','"+TextStudentIdReserve.getText()+"','"+TextBookNameReserve.getText()+"')";
}
```

6. Issued books

```
private void IssueButtonActionPerformed(java.awt.event.ActionEvent evt) {
    String Data="-insert into issuetable values('"+TextBookNameIssue.getText()+"','"+TextBookIdIssue.getText()+"','"+
        +TextStudentNameIssue.getText()+"','"+TextStudentIdIssue.getText()+"','"+TextDateIssue.getText()+"')";
}
```

7. Add new books

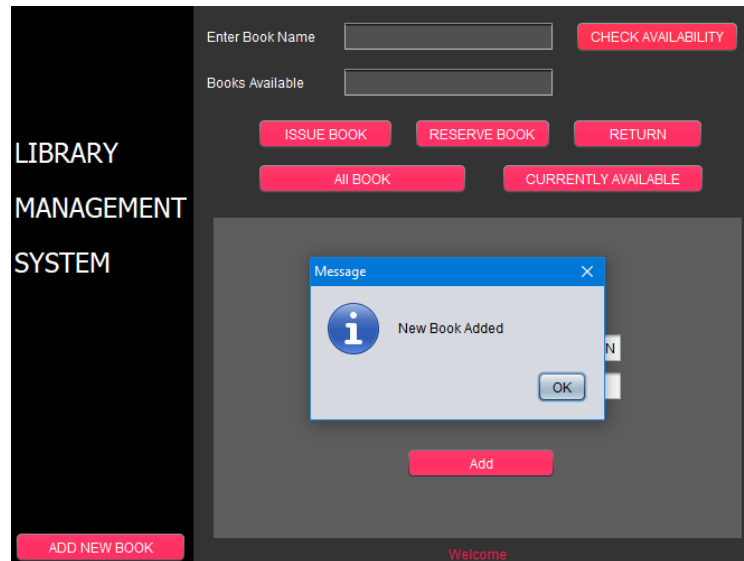
```
private void AddButtonActionPerformed(java.awt.event.ActionEvent evt) {
    String Data="-insert into alltable values('"+TextBookNameAdd.getText()+"','"+TextBookIdAdd.getText()+"')";
    sendCommandWithNoRecieveData(Data);
    Data="";
    Data="-insert into availbletable values('"+TextBookNameAdd.getText()+"','"+TextBookIdAdd.getText()+"')";
}
```

8. Currently available

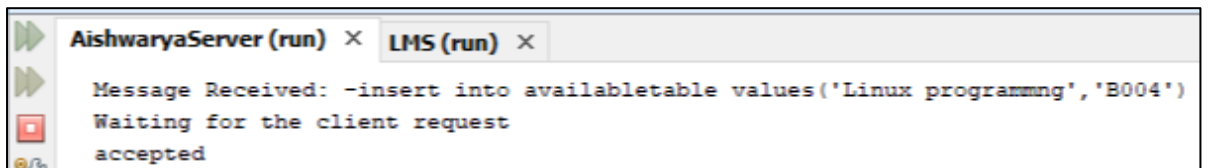
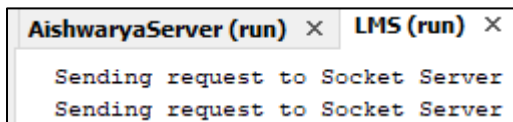
```
private void CurrentyAvailableButtonActionPerformed(java.awt.event.ActionEvent evt) {
    String Data ="select * from availbletable";
}
```

5. Test results, using an adequate set of test cases and screenshots

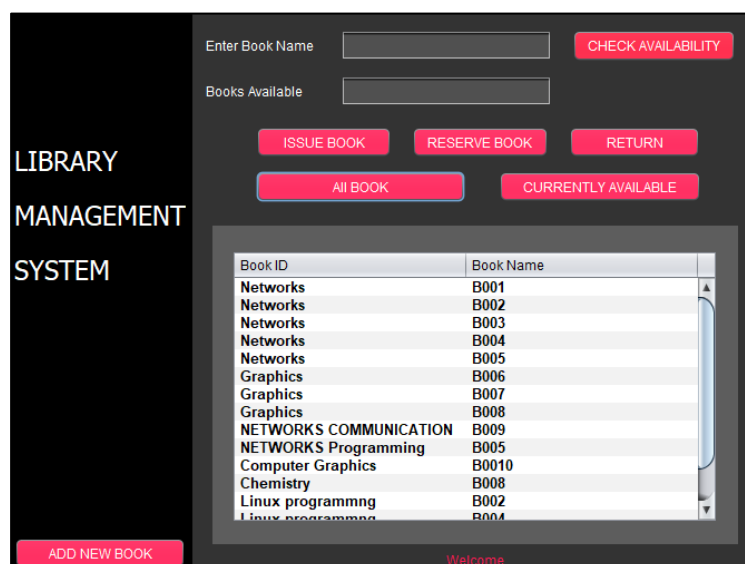
ADD NEW BOOK



In this test case, the software is able to add books, here multiple books of same name and different IDs can be added. The request sent by the client (here, LMS) to the server to add data as shown below and the response redirected by the server displaying the request accepted.



DISPLAY ALL BOOKS AVAILABLE IN THE LIBRARY



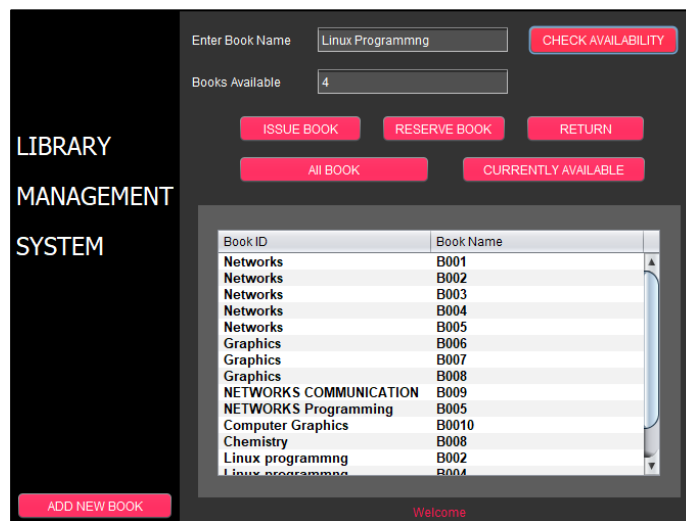
In the above test case, the output is shown for all the books in the library. Here, the client requests the server to retrieve data from the database and display the request made by the client. As, shown below the client sends the request and the server responds to it.

Message Received: select * from availabletable	
---Networks	B004
Networks	B005
Graphics	B006
Graphics	B007
Graphics	B008
NETWORKS COMMUNICATION	B009
NETWORKS Programming	B005
Computer Graphics	B0010

Sending request to Socket Server	
Message: Networks	B001
Networks	B002
Networks	B003
Networks	B004
Networks	B005
Graphics	B006
Graphics	B007
Graphics	B008
NETWORKS COMMUNICATION	B009
NETWORKS Programming	B005
Computer Graphics	B0010

Here, client sent the request to the server and the server responds to the client queries.

CURRENTLY AVAILABLE BOOK



LIBRARY MANAGEMENT SYSTEM

Enter Book Name: Linux Programming CHECK AVAILABILITY

Books Available: 4

ISSUE BOOK RESERVE BOOK RETURN

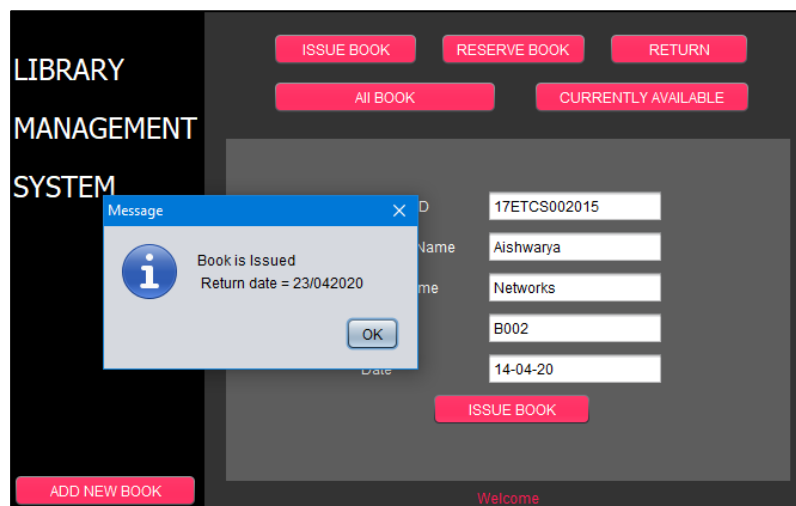
ALL BOOK CURRENTLY AVAILABLE

Book ID	Book Name
Networks	B001
Networks	B002
Networks	B003
Networks	B004
Networks	B005
Graphics	B006
Graphics	B007
Graphics	B008
NETWORKS COMMUNICATION	B009
NETWORKS Programming	B005
Computer Graphics	B0010
Chemistry	B008
Linux programming	B002
Linux programming	B004

ADD NEW BOOK Welcome

The above snippet, query is made for the availability of a particular book, the client sends the query request to the server, server accepts the request and sends the query made by the server.

ISSUE Book



LIBRARY MANAGEMENT SYSTEM

ISSUE BOOK RESERVE BOOK RETURN

ALL BOOK CURRENTLY AVAILABLE

Message: Book is Issued
Return date = 23/04/2020

OK

ADD NEW BOOK Welcome

```
Sending request to Socket Server
Sending request to Socket Server
```

Request sent to the socket server by the client

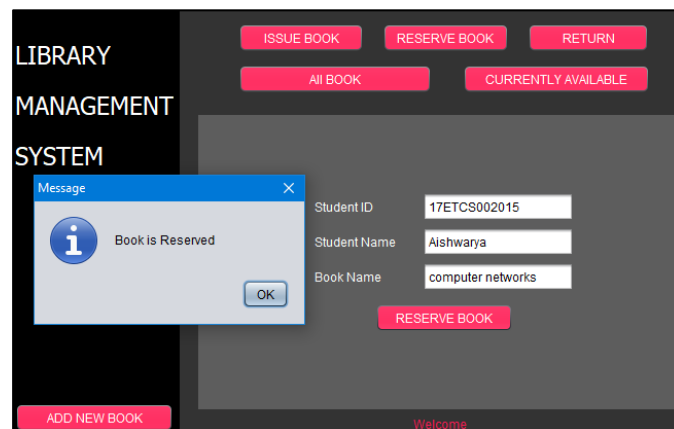
```
Waiting for the client request
accepted
Message Received: select * from issuetable where StudentId='17ETCS002015'
---Networks      B001      Aishwarya      17ETCS002015      12/04/2020
Networks          B002      Aishwarya      17ETCS002015      12/04/2020
Networks          B003      Aishwarya      17ETCS002015      12/04/2020

Waiting for the client request
accepted
```

Reply by the server to the client's query

In the above snippet, the software allows the client to issue book by entering the details such as ID, name, Book name, book ID, the system takes the date of the book issued and can eventually provide the return date to the user. As shown above date of issue is 14/04/20, hence the date of return, as shown in the pop-up window is 23/04/20.

RESERVE BOOK



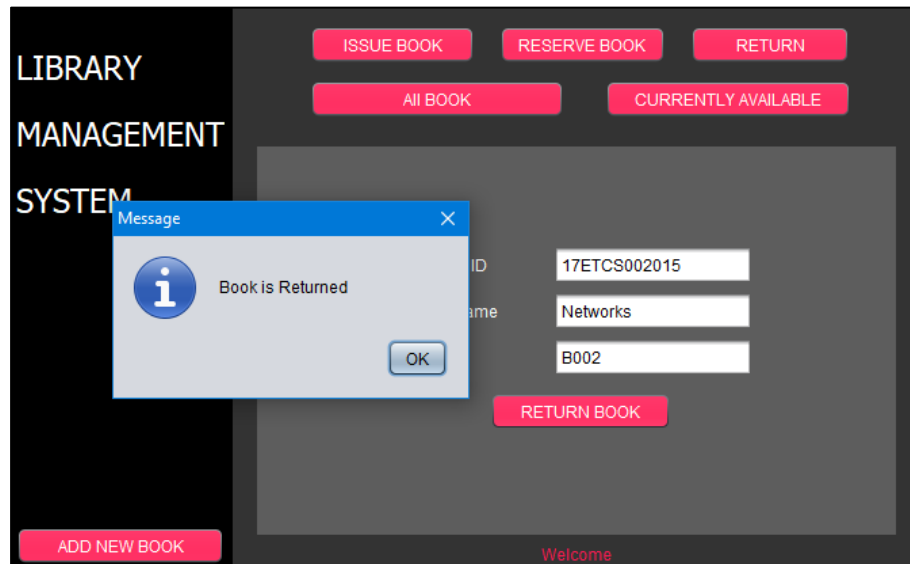
```
Sending request to Socket Server
Sending request to Socket Server
```

```
Waiting for the client request
accepted
Message Received: -delete from issuetable where BookId='B001'
Waiting for the client request
accepted
```

Request sent to socket server by the client Reply by the server to the client's query

The above snippet, the software allows the user reserve a book which is not available to be issued. Hence It can be reserved by student name, ID and book name. a pop-up window appears revealing, book reserved.

RETURN BOOK



```
Sending request to Socket Server
Sending request to Socket Server
```

Request sent to the socket server by the client

```
Waiting for the client request
accepted
Message Received: -insert into availabletable values('Networks','B002')
Waiting for the client request
accepted
```

Reply by the server to the client's query

The above snippet, the software allows the client to return the book by entering credentials such as student ID, book name and book ID, the pop-up window shows the book returned.

6. Conclusion

A library management system is developed based on the client-server-based communication, using socket programming. Client- server architecture is a way of building computer systems. It consists in splitting the system into two categories of programs: clients and servers.

A client is a program or a process which connects to another program (the server) and lets it carry out a specific task. In particular it might require supplying some data from the server, as can be seen in the LMS developed above, client connects to the server to retrieve data from the database and answer the queries.

A server is a program or a process which provides services for clients. For example, it might supply some data or a result of processing data to clients.

The library management system developed, using java socket programming language and a GUI is developed using java swing connected to the MySQL database.