SDL_Assignment_3_Code:

Server Code:

```
import java.net.*;
import java.io.*;
import java.util.*;
import java.sql.*;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

class Booking
{
    public synchronized String book(String dest_city, int no_of_pass,
HashMap<String, Integer> passenger_list, Connection conn, int user_id)  {
        try {
            PreparedStatement ps = null;
            ps = conn.prepareStatement("select * from cities where city_name =
?");
            ps.setString(1,dest_city);
            ResultSet rs = ps.executeQuery();
            if(!rs.next())
            {
                return "City not available";
            }
            String city_name=rs.getString(2);
            int city_id = rs.getInt(1);
            ps = conn.prepareStatement("select * from buses where city_id = ?");
            ps.setInt(1,city_id);
            rs = ps.executeQuery();
            if(!rs.next())
            {
                return "Bus not available";
            }
            int bus_id = rs.getInt(1);
            int seat_booked = rs.getInt(2);
            if(seat_booked==40) {
                return "Seats Not Available";
            }
            Random rand = new Random();
            int booking_id = rand.nextInt(999999);
            ps = conn.prepareStatement("select seat_no from passengers where
bus_id = ?");
            ps.setInt(1,bus_id);
            rs = ps.executeQuery();
            while(rs.next())
            {
                if(passenger_list.containsValue(rs.getInt(1)))
                {
                    return rs.getInt(1)+" seat is already booked!!\n Please see
seat available option!!";
                }
```

```
            }
            ps = conn.prepareStatement("insert into booking values(?,?,?,?,?)");
            ps.setInt(1, user_id);
            ps.setInt(2, booking_id);
            ps.setInt(3, city_id);
            ps.setInt(4, bus_id);
            ps.setInt(5, no_of_pass);
            ps.executeUpdate();
            ps = conn.prepareStatement("insert into passengers
values(?,?,?,?,?)");
            ps.setInt(2, booking_id);
            ps.setInt(3, bus_id);
            Set<String> key = passenger_list.keySet();
            Iterator<String> i = key.iterator();
            int j=1;
            String book_statement="Booking Id:" + booking_id + "\nBus Id:" +bus_id
+ "\nPassenger List:\n";
            while(i.hasNext())
            {
                ps.setInt(1,j++);
                String name = String.valueOf(i.next());
                int seat = passenger_list.get(name);
                ps.setString(4,name);
                ps.setInt(5, seat);
                ps.executeUpdate();
                book_statement += i + ". " + name + " " + seat + "\n";
            }
            ps = conn.prepareStatement("update buses set seat_booked = seat_booked
+ ? where bus_id = ?");
            ps.setInt(1,no_of_pass);
            ps.setInt(2,bus_id);
            ps.executeUpdate();
            return book_statement;
        }
        catch(Exception e)
        {
            return String.valueOf(e);
        }
    }

    public String check(int booking_id,Connection conn)
    {
        try{

            PreparedStatement ps = conn.prepareStatement(" select
city_name,bus_id,passenger_name,seat_no,username from booking natural join
passengers natural join cities natural join users where booking_id = ?");
            ps.setInt(1,booking_id);
            ResultSet rs = ps.executeQuery();
            if(!rs.next())
            {
                return "No Booking!!";
```

```
            }
            String check_statement="";
            String city = rs.getString(1) ;
            int bus_id = rs.getInt(2 );
            String name = rs.getString(3);
            int seat = rs.getInt(4);
            String username = rs.getString(5);
            check_statement += "Destination City: " + city + "\nBus id: " + bus_id
+ "\nPassenger List:\nName: "+name+" Seat Number: " + seat;
            while(rs.next())
            {
                name = rs.getString(3);
                seat = rs.getInt(4);
                check_statement += "\nName: "+name+" Seat Number: "+seat;
            }
            check_statement += "\nBooked by: " + username;
            return check_statement;
        }
        catch(Exception e)
        {
            return String.valueOf(e);
        }
    }
    public String cancel(int booking_id,int bus_id,Connection conn)
    {
        try
        {
            PreparedStatement ps = conn.prepareStatement("select * from booking
where booking_id = ? and bus_id = ?");
            ps.setInt(1,booking_id);
            ps.setInt(2,bus_id);
            ResultSet rs = ps.executeQuery();
            if(!rs.next())
                return "No bookings!!";
            int no_of_pass = rs.getInt(5);
            ps = conn.prepareStatement("delete from booking where booking_id = ?
and bus_id = ?");
            ps.setInt(1,booking_id);
            ps.setInt(2,bus_id);
            ps.executeUpdate();
            ps = conn.prepareStatement("update buses set seat_booked = seat_booked
- ? where bus_id = ?");
            ps.setInt(1,no_of_pass);
            ps.setInt(2,bus_id);
            ps.executeUpdate();
            return "Ticket Cancelled!!";
        }
        catch(Exception e)
        {
            return String.valueOf(e);
        }
    }
```

```
    public String availability(String dest_city,Connection conn)
    {
        try
        {
            PreparedStatement ps = conn.prepareStatement("select city_id from
cities where city_name = ?");
            ps.setString(1,dest_city);
            ResultSet rs = ps.executeQuery();
            if(!rs.next())
            {
                return "City Not Available!!";
            }
            int city_id = rs.getInt(1);
            ps = conn.prepareStatement("select bus_id,seat_booked from buses where
city_id = ?");
            ps.setInt(1,city_id);
            rs = ps.executeQuery();
            String available_statement ="";
            while(rs.next()) {
                int bus_id = rs.getInt(1);
                available_statement += "Bus Id:" + bus_id + "\n";
                int seat_booked = rs.getInt(2);
                if (seat_booked == 0) {
                    for (int i = 1; i <= 40;) {
                        available_statement += (i++) + ".A  " + (i++) + ".A\t\t" +
(i++) + ".A  " + (i++) + ".A\n";
                    }
                } else {
                    ps = conn.prepareStatement("select seat_no from passengers
where bus_id = ?");
                    ps.setInt(1,bus_id);
                    ResultSet rs1 = ps.executeQuery();
                    ArrayList<Integer> seats = new ArrayList<>();
                    while(rs1.next())
                    {
                        seats.add(rs1.getInt(1));
                    }
                    for(int i=1;i<=40;i++)
                    {
                        if(seats.contains(i))
                        {
                            if(i%4==0)
                                available_statement += i + ".B\n";
                            else if(i%2==0)
                                available_statement += i+ ".B\t\t";
                            else
                                available_statement += i + ".B  ";
                        }
                        else
                        {
                            if(i%4==0)
                                available_statement += i + ".A\n";
```

```
                            else if(i%2==0)
                                available_statement += i+ ".A\t\t";
                            else
                                available_statement += i + ".A  ";

                        }
                    }
                }
            }
            return available_statement;
        }
        catch(Exception e)
        {
            return String.valueOf(e);
        }
    }
}

public class Server {
    public static void main(String[] args) throws Exception {
        ServerSocket ss = new ServerSocket(8081);
        ExecutorService executor = Executors.newFixedThreadPool(5);
        System.out.println("Starting Server...");
        while (true) {
            Socket s = null;
            try {
                s = ss.accept();
                System.out.println("New Client Connected:" + s);
                DataOutputStream outToClient = new
DataOutputStream(s.getOutputStream());
                DataInputStream inFromClient = new
DataInputStream(s.getInputStream());
                ObjectOutputStream os=new ObjectOutputStream(s.getOutputStream());
                ObjectInputStream is=new ObjectInputStream(s.getInputStream());
                System.out.println("Assigning Thread to Client");
                Thread t = new ClientHandler(s, inFromClient, outToClient,os,is);
                executor.execute(t);
            } catch (Exception e) {
                assert s != null;
                s.close();
                e.printStackTrace();
            }
        }
    }
}
class ClientHandler extends Thread {
    final DataInputStream inFromClient;
    final DataOutputStream outToClient;
    final Socket s;
    final ObjectInputStream is;
    final ObjectOutputStream os;
```

```
    public ClientHandler(Socket s, DataInputStream inFromClient, DataOutputStream
outToClient,ObjectOutputStream os,ObjectInputStream is) throws Exception {
        this.s = s;
        this.inFromClient = inFromClient;
        this.outToClient = outToClient;
        this.os = os;
        this.is = is;
    }

    public void run() {
        try {
            Scanner sc = new Scanner(System.in);
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/sdl_assignment_3?autoReco
nnect=true&useSSL=false", "root", "varunkarwa");
            Statement stmt = conn.createStatement();
            String choice,user_type="N";
            PreparedStatement ps=null;
            int user_id=0;
            do{
                boolean Authentication=false;
                while(!Authentication){
                    outToClient.writeUTF("1.Login\n2.Sign Up:");
                    choice = inFromClient.readUTF();
                    switch(choice)
                    {
                        case "1":{
                            outToClient.writeUTF("Enter Username:");
                            String username = inFromClient.readUTF();
                            outToClient.writeUTF("Enter Password:");
                            String password = inFromClient.readUTF();
                            ps = conn.prepareStatement("select
user_id,name,contact_no,user_type from users where username = ? and password =
?");
                            ps.setString(1, username);
                            ps.setString(2, password);
                            try{ResultSet rs = ps.executeQuery();
                                if (rs.next()) {
                                    outToClient.writeUTF("User_id:" + rs.getInt(1)
+ "\nName:" + rs.getString(2) + "\nContact: " + rs.getString(3));
                                    user_type=rs.getString(4);
                                    user_id = rs.getInt(1);
                                    Authentication = true;
                                    System.out.println("User logged in!\n" +
"User_id:" + rs.getInt(1) + "\nName:" + rs.getString(2) + "\nContact: " +
rs.getString(3));

                                } else {
                                    outToClient.writeUTF("Invalid Credentials!!");
                                }}
                            catch (Exception e)
```

```
                            {
                                System.out.println(e);
                                outToClient.writeUTF("Error!");}
                            outToClient.writeBoolean(Authentication);
                            outToClient.writeUTF(user_type);
                            break;
                        }
                        case "2": {
                            outToClient.writeUTF("Enter Name:");
                            String name = inFromClient.readUTF();
                            outToClient.writeUTF("Enter username:");
                            String username = inFromClient.readUTF();
                            outToClient.writeUTF("Enter password:");
                            String password = inFromClient.readUTF();
                            outToClient.writeUTF("Enter Contact_no:");
                            int contact = inFromClient.readInt();
                            outToClient.writeUTF("User Type(Admin A/Normal N):");
                            String type = inFromClient.readUTF();
                            ps = conn.prepareStatement("insert into
users(username,password,name,contact_no,user_type) values(?,?,?,?,?)");
                            ps.setString(1,username);
                            ps.setString(2,password);
                            ps.setString(3,name);
                            ps.setInt(4,contact);
                            ps.setString(5,type);
                            try {
                                int rs = ps.executeUpdate();
                                outToClient.writeUTF("Account Created!!\nLogin to
continue!!");
                                System.out.println("Account Created!!\n Users
Details:\nUsername:" + username + "\nPassword:" + password + "\nName:" + name +
"\nContact No:" + contact);
                            }catch(Exception e)
                            {
                                outToClient.writeUTF(String.valueOf(e));
                            }
                            break;
                        }
                        default:
                            System.out.println("Incorrect Choice!!");
                            break;
                    }
                }
                Booking b = new Booking();
                if(user_type.equals("N"))
                {
                    do {
                        outToClient.writeUTF("------MENU------");
                        outToClient.writeUTF("1.Book Tickets\n2.Check
Reservation\n3.Cancel Tickets\n4.Seat Availability\n5.Service to
Cities\n6.Logout");
                        choice = inFromClient.readUTF();
```

```
                        switch (choice) {
                            case "1": {
                                outToClient.writeUTF("Enter destination city:");
                                String dest_city = inFromClient.readUTF();
                                outToClient.writeUTF("Enter no.of passengers");
                                int no_of_pass = inFromClient.readByte();
                                outToClient.writeUTF("Enter passenger's name and
seat_no");
                                HashMap<String, Integer> passenger_list =
(HashMap<String, Integer>) is.readObject();
                                String book_statement = b.book(dest_city,
no_of_pass, passenger_list, conn, user_id);
                                outToClient.writeUTF(book_statement);
                                break;
                            }
                            case "2":{
                                outToClient.writeUTF("Enter Booking ID:");
                                int booking_id = inFromClient.readInt();
                                String check_statement = b.check(booking_id,conn);
                                outToClient.writeUTF(check_statement);
                                break;
                            }
                            case "3":{
                                outToClient.writeUTF("Enter booking id:");
                                int booking_id = inFromClient.readInt();
                                outToClient.writeUTF("Enter bus_id:");
                                int bus_id = inFromClient.readInt();
                                String cancel_statement =
b.cancel(booking_id,bus_id,conn);
                                outToClient.writeUTF(cancel_statement);
                                break;
                            }
                            case "4":
                            {
                                outToClient.writeUTF("Enter destination city:");
                                String dest_city = inFromClient.readUTF();
                                String available_statement =
b.availability(dest_city,conn);
                                outToClient.writeUTF(available_statement);
                                break;
                            }
                            case "5":
                            {
                                ResultSet rs = stmt.executeQuery("select city_name
from cities");
                                String city_statement = "Cities:\n";
                                while(rs.next())
                                {
                                    city_statement += rs.getString(1) + "\n";
                                }
                                outToClient.writeUTF(city_statement);
                                break;
```

```
                            }
                        }
                    }while(!choice.equals("6"));
                }
                else
                {
                    do
                    {
                        outToClient.writeUTF("------MENU-----");
                        outToClient.writeUTF("1.Add City\n2.Add Bus\n3.Delete
Bus\n4.Delete City\n5.Passenger List\n");
                        choice = inFromClient.readUTF();
                        switch(choice)
                        {
                            case "1":
                            {
                                outToClient.writeUTF("Enter City Name");
                                String dest_city = inFromClient.readUTF();
                                ps = conn.prepareStatement("insert into
cities(city_name) values(?)");
                                ps.setString(1,dest_city);
                                try
                                {
                                    ps.executeUpdate();
                                    outToClient.writeUTF("Added Successfully");
                                }
                                catch(Exception e)
                                {
                                    outToClient.writeUTF(String.valueOf(e));
                                }
                                break;
                            }
                            case "2":{
                                outToClient.writeUTF("Enter City Name");
                                String dest_city = inFromClient.readUTF();
                                ps = conn.prepareStatement("select city_id from
cities where city_name = ?");
                                ps.setString(1,dest_city);
                                ResultSet rs = ps.executeQuery();
                                if(!rs.next())
                                {
                                    outToClient.writeUTF("Please first add this
city!!");
                                    break;
                                }
                                int city_id = rs.getInt(1);
                                ps = conn.prepareStatement("insert into
buses(seat_booked,city_id) values(0,?)");
                                ps.setInt(1,city_id);
                                ps.executeUpdate();
                                outToClient.writeUTF("Added Successfully!");
                                break;
```

```
                            }
                            case "3":{
                                outToClient.writeUTF("Enter bus id:");
                                int bus_id = inFromClient.readInt();
                                ps = conn.prepareStatement("delete from buses
where bus_id = ?");
                                ps.setInt(1,bus_id);
                                if(ps.executeUpdate() == 0)
                                    outToClient.writeUTF("Bus already
cancelled!!");
                                else
                                    outToClient.writeUTF("Bus Cancelled!!");
                                break;
                            }
                            case "4":{
                                outToClient.writeUTF("Enter city_name:");
                                String dest_city = inFromClient.readUTF();
                                ps = conn.prepareStatement("delete from cities
where city_name = ?");
                                ps.setString(1,dest_city);
                                if(ps.executeUpdate() == 0)
                                    outToClient.writeUTF("First add the city
please!!");
                                else
                                    outToClient.writeUTF("City deleted
Successfully!!");
                                break;
                            }
                            case "5":
                            {
                                String passenger_statement = "Today's
Bookings:\n";
                                ResultSet rs =stmt.executeQuery("select * from
passengers order by bus_id,seat_no");
                                if(rs.next()) {
                                    passenger_statement += "Passenger Id:" +
rs.getInt(1) + "\tBooking Id:" + rs.getInt(2) + "\tBus Id:" + rs.getInt(3) +
"\tName:" + rs.getString(4) + "\tSeat:" + rs.getInt(5) + "\n";
                                    while (rs.next()) {
                                        passenger_statement += "Passenger Id:" +
rs.getInt(1) + "\tBooking Id:" + rs.getInt(2) + "\tBus Id:" + rs.getInt(3) +
"\tName:" + rs.getString(4) + "\tSeat:" + rs.getInt(5) + "\n";
                                    }
                                    outToClient.writeUTF(passenger_statement);
                                }
                                else
                                    outToClient.writeUTF("No Bookings!!");
                                break;
                            }
                        }
                    }while(!choice.equals("6"));
                }
```

```
                outToClient.writeUTF("1.Exit\n2.Continue");
                choice = inFromClient.readUTF();
            }while(!choice.equals("1"));
        }
        catch (Exception e) {
            System.out.println(e);
        }
        try{
            this.inFromClient.close();
            this.outToClient.close();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

Client Code:

```
import java.net.*;
import java.io.*;
import java.util.HashMap;
import java.util.Scanner;

public class Client {
    public static void main(String[] args) throws IOException
    {
        String serveraddress = "127.0.0.1";
        int port =8081;
        String user_type = "N";
        Scanner sc=new Scanner(System.in);
        String choice;
        Socket client = new Socket(serveraddress,port);
        System.out.println("Connected to: "+client.getRemoteSocketAddress());
        DataOutputStream outToServer = new
DataOutputStream(client.getOutputStream());
        DataInputStream inFromServer = new
DataInputStream(client.getInputStream());
        ObjectOutputStream os=new ObjectOutputStream(client.getOutputStream());
        ObjectInputStream is=new ObjectInputStream(client.getInputStream());
        do{
            boolean Authentication=false;
            do {
                System.out.println(inFromServer.readUTF());
                choice = sc.next();
                outToServer.writeUTF(choice);
                switch (choice) {
                    case "1": {
                        System.out.println(inFromServer.readUTF());
                        String username = sc.next();
```

```
                    outToServer.writeUTF(username);
                    System.out.println(inFromServer.readUTF());
                    String password = sc.next();
                    outToServer.writeUTF(password);
                    System.out.println(inFromServer.readUTF());
                    Authentication = inFromServer.readBoolean();
                    user_type = inFromServer.readUTF();
                    break;
                }
                case "2": {
                    System.out.println(inFromServer.readUTF());
                    String name = sc.next();
                    outToServer.writeUTF(name);
                    System.out.println(inFromServer.readUTF());
                    String username = sc.next();
                    outToServer.writeUTF(username);
                    System.out.println(inFromServer.readUTF());
                    String password = sc.next();
                    outToServer.writeUTF(password);
                    System.out.println(inFromServer.readUTF());
                    int contact = sc.nextInt();
                    outToServer.writeInt(contact);
                    System.out.println(inFromServer.readUTF());
                    String type = sc.next();
                    outToServer.writeUTF(type);
                    System.out.println(inFromServer.readUTF());
                    break;
                }
                default:
                    System.out.println("Incorrect Choice!!");
                    break;
            }
        }while(!Authentication);
        if(user_type.equals("N"))
        {
            do {
                System.out.println(inFromServer.readUTF());
                System.out.println(inFromServer.readUTF());
                choice = sc.next();
                outToServer.writeUTF(choice);
                switch(choice)
                {
                    case "1":
                    {
                        System.out.println(inFromServer.readUTF());
                        String dest_city = sc.next();
                        outToServer.writeUTF(dest_city);
                        System.out.println(inFromServer.readUTF());
                        int no_of_pass = sc.nextInt();
                        outToServer.writeByte(no_of_pass);
                        System.out.println(inFromServer.readUTF());
                        HashMap<String,Integer> passenger_list = new
```

```
HashMap<>();
                        for(int i=0;i<no_of_pass;i++)
                        {
                            passenger_list.put(sc.next(),sc.nextInt());
                        }
                        os.writeObject(passenger_list);
                        System.out.println(inFromServer.readUTF());
                        break;
                    }
                    case "2":{
                        System.out.println(inFromServer.readUTF());
                        int booking_id = sc.nextInt();
                        outToServer.writeInt(booking_id);
                        System.out.println(inFromServer.readUTF());
                        break;
                    }
                    case "3":{
                        System.out.println(inFromServer.readUTF());
                        int booking_id = sc.nextInt();
                        outToServer.writeInt(booking_id);
                        System.out.println(inFromServer.readUTF());
                        int bus_id = sc.nextInt();
                        outToServer.writeInt(bus_id);
                        System.out.println(inFromServer.readUTF());
                        break;
                    }
                    case "4":{
                        System.out.println(inFromServer.readUTF());
                        String dest_city = sc.next();
                        outToServer.writeUTF(dest_city);
                        System.out.println(inFromServer.readUTF());
                        break;
                    }
                    case "5":{
                        System.out.println(inFromServer.readUTF());
                        break;
                    }
                }
            }while(!choice.equals("6"));
        }
        else
        {
            do
            {
                System.out.println(inFromServer.readUTF());
                System.out.println(inFromServer.readUTF());
                choice = sc.next();
                outToServer.writeUTF(choice);
                switch(choice)
                {
                    case "1":{
                        System.out.println(inFromServer.readUTF());
```

```
                            String dest_city = sc.next();
                            outToServer.writeUTF(dest_city);
                            System.out.println(inFromServer.readUTF());
                            break;
                        }
                        case "2":{
                            System.out.println(inFromServer.readUTF());
                            String dest_city = sc.next();
                            outToServer.writeUTF(dest_city);
                            System.out.println(inFromServer.readUTF());
                            break;
                        }
                        case "3":{
                            System.out.println(inFromServer.readUTF());
                            int bus_id = sc.nextInt();
                            outToServer.writeInt(bus_id);
                            System.out.println(inFromServer.readUTF());
                            break;
                        }
                        case "4":{
                            System.out.println(inFromServer.readUTF());
                            String dest_city = sc.next();
                            outToServer.writeUTF(dest_city);
                            System.out.println(inFromServer.readUTF());
                            break;
                        }
                        case "5":System.out.println(inFromServer.readUTF());
                        break;
                    }
                }while(!choice.equals("6"));
            }
            System.out.println(inFromServer.readUTF());
            choice = sc.next();
            outToServer.writeUTF(choice);
        }while(!choice.equals("1"));
    }
}
```

Client_1 Output(Normal user):

Connected to: /127.0.0.1:8081
1.Login
2.Sign Up:
1
Enter Username:
varunkarwa
Enter Password:
vk123456
User_id:1
Name:Varun Karwa
Contact: 220479
------MENU------
1.Book Tickets
2.Check Reservation
3.Cancel Tickets
4.Seat Availability
5.Service to Cities
6.Logout
1
Enter destination city:
Ahemdabad
Enter no.of passengers
1
Enter passenger's name and seat_no
Stuti
15
Booking Id:824623
Bus Id:100
Passenger List:
java.util.HashMap$KeyIterator@51f89e76. Stuti 15

------MENU------
1.Book Tickets
2.Check Reservation
3.Cancel Tickets
4.Seat Availability
5.Service to Cities
6.Logout
2
Enter Booking ID:
824623
Destination City: Ahemdabad
Bus id: 100
Passenger List:
Name: Stuti Seat Number: 15
Booked by: varunkarwa
------MENU------
1.Book Tickets
2.Check Reservation
3.Cancel Tickets
4.Seat Availability
5.Service to Cities
6.Logout
4
Enter destination city:
Nagpur
Bus Id:104

1.A 2.A	3.A 4.A
5.A 6.A	7.A 8.A
9.B 10.B	11.A 12.A
13.A 14.A	15.A 16.A

17.A 18.A 19.A 20.A
21.A 22.A 23.A 24.A
25.A 26.A 27.A 28.A
29.A 30.A 31.A 32.A
33.A 34.A 35.A 36.A
37.A 38.A 39.A 40.A

------MENU------
1.Book Tickets
2.Check Reservation
3.Cancel Tickets
4.Seat Availability
5.Service to Cities
6.Logout
5
Cities:
Ahemdabad
Jalgaon
Kolhapur
Mumbai
Nagpur
Nashik
Solapur

------MENU------
1.Book Tickets
2.Check Reservation
3.Cancel Tickets
4.Seat Availability
5.Service to Cities
6.Logout

3
Enter booking id:
844447
Enter bus_id:
102
Ticket Cancelled!!
------MENU------
1.Book Tickets
2.Check Reservation
3.Cancel Tickets
4.Seat Availability
5.Service to Cities
6.Logout

Client_2 Output(Admin):

Connected to: /127.0.0.1:8081
1.Login
2.Sign Up:
1
Enter Username:
tejasd12
Enter Password:
tdj1302
Invalid Credentials!!
1.Login
2.Sign Up:
1
Enter Username:
tejas12
Enter Password:

tdj1302
User_id:2
Name:Tejas Dahad
Contact: 246468
------MENU-----
1.Add City
2.Add Bus
3.Delete Bus
4.Delete City
5.Passenger List

1
Enter City Name
Kolhapur //City Already added!
java.sql.SQLIntegrityConstraintViolationException: Duplicate entry 'Kolhapur' for key 'cities.city_name'
------MENU-----
1.Add City
2.Add Bus
3.Delete Bus
4.Delete City
5.Passenger List

1
Enter City Name
Amravati
Added Successfully
------MENU-----
1.Add City
2.Add Bus
3.Delete Bus

4.Delete City
5.Passenger List

2
Enter City Name
Nashik
Added Successfully!
------MENU-----
1.Add City
2.Add Bus
3.Delete Bus
4.Delete City
5.Passenger List

3
Enter bus id:
102
Bus Cancelled!!
------MENU-----
1.Add City
2.Add Bus
3.Delete Bus
4.Delete City
5.Passenger List

4
Enter city_name:
Solapur
City deleted Successfully!!
------MENU-----
1.Add City

2.Add Bus
3.Delete Bus
4.Delete City
5.Passenger List

5
Today's Bookings:
Passenger Id:1 Booking Id:556859 Bus Id:100 Name:Miti Seat:11
Passenger Id:2 Booking Id:556859 Bus Id:100 Name:Siddhi Seat:12
Passenger Id:1 Booking Id:824623 Bus Id:100 Name:Stuti Seat:15
Passenger Id:2 Booking Id:935143 Bus Id:104 Name:VarunSeat:9
Passenger Id:1 Booking Id:935143 Bus Id:104 Name:Prachi Seat:10

------MENU-----
1.Add City
2.Add Bus
3.Delete Bus
4.Delete City
5.Passenger List

Database Tables:

```
mysql> select * from users;
+---------+------------+----------+-------------+------------+-----------+
| user_id | username   | password | name        | contact_no | user_type |
+---------+------------+----------+-------------+------------+-----------+
|       1 | varunkarwa | vk123456 | Varun Karwa |     220479 | N         |
|       2 | tejas12    | tdj1302  | Tejas Dahad |     246468 | A         |
|       3 | mehta      | mpm1712  | Miti        |     262347 | N         |
+---------+------------+----------+-------------+------------+-----------+
3 rows in set (0.01 sec)

mysql> select * from booking;
+---------+------------+---------+--------+------------------+
| user_id | booking_id | city_id | bus_id | no_of_passengers |
```

```
+---------+------------+---------+--------+------------------+
|      1 |    556859 |      1 |   100 |            2 |
|      1 |    824623 |      1 |   100 |            1 |
|      1 |    935143 |      5 |   104 |            2 |
+---------+------------+---------+--------+------------------+
3 rows in set (0.01 sec)

mysql> select * from cities;
+---------+-----------+
| city_id | city_name |
+---------+-----------+
|      1 | Ahemdabad |
|     12 | Amravati  |
|      4 | Jalgaon   |
|     10 | Kolhapur  |
|      2 | Mumbai    |
|      5 | Nagpur    |
|      3 | Nashik    |
+---------+-----------+
7 rows in set (0.00 sec)

mysql> select * from buses;
+--------+-------------+---------+
| bus_id | seat_booked | city_id |
+--------+-------------+---------+
|   100 |         3 |      1 |
|   101 |         0 |      2 |
|   103 |         0 |      4 |
|   104 |         2 |      5 |
|   107 |         0 |      1 |
|   108 |         0 |      3 |
+--------+-------------+---------+
6 rows in set (0.00 sec)

mysql> select * from passengers;
+--------------+------------+--------+----------------+---------+
| passenger_id | booking_id | bus_id | passenger_name | seat_no |
+--------------+------------+--------+----------------+---------+
|          1 |    556859 |    100 | Miti         |     11 |
|          2 |    556859 |    100 | Siddhi       |     12 |
|          1 |    935143 |    104 | Prachi       |     10 |
|          2 |    935143 |    104 | Varun        |      9 |
|          1 |    824623 |    100 | Stuti        |     15 |
+--------------+------------+--------+----------------+---------
```

Page No.
Date

Assignment - 03

* Title: JDBC, Multithreading, ThreadPool

* Problem Statement: Develop an application by using JDBC, Multithreading, concurrency, synchronous and asynchronous callbacks, ThreadPools using ExecutorService.

* Objective: 1. To learn database connectivity
  2. To learn concurrency

* Outcome: students should be able to implement
  1. JDBC drivers
  2. Concurrency in their application

* S/W AND H/W Requirements:
  IntelliJ IDE, JDBC connectivity, Windows 10 (64-bit), i5 processor

* PREREQUISITES: Multithreading in Java, Object Oriented Programming, SQL.

* THEORY:

• JDBC

Java JDBC is a Java API to connect and execute query with the database, JDBC API uses jdbc drivers to connect with the database.

Teacher's Sign.: _____

Following steps needto be followed:

i) import the package:
ex java.sql.* needs to be imported

ii) Load & register the drivers:
Load: The jdbc drivers used for connection should be available in your system. If you are using Eclipse IDE, then you have to provide link in properties while if you are running your jdbc code on terminal, then you have to provide classpath in bashrc file.

Register: In jdbc code, we need to register a drivers for use. A method forName() is provided for same purpose.
ex Class.forName("com.mysql.jdbc.Driver");

iii) Establish a Connection to the database:
Create a connection object. Provide URL, username and password
ex:
```
Connection conn = DriverManager.getConnection
("jdbc:mysql://localhost:3306/dbname", "username",
"password");
```

iv) Create a statement object from the connection
Statement object is used for executing queries on database.
ex Statement stmt = conn.createStatement();

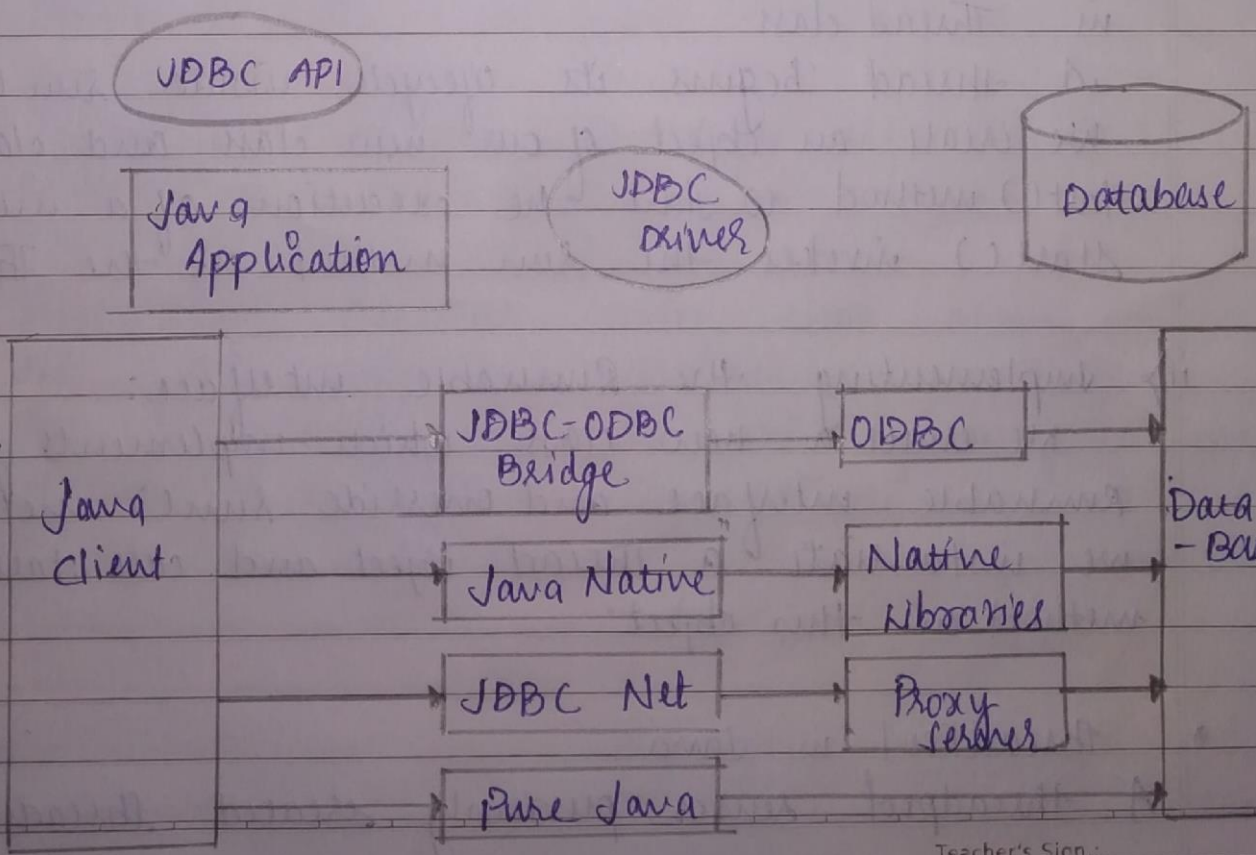
Teacher's Sign.:

v) Use the statement object to execute query. If we are fetching data from database, then we need to define Resultset object.
eg: ResultSet rs = stmt.executeQuery("select * from users");
or stmt.executeUpdate("delete from User");

vi) Process Result:
If we are fetching data, we can get it in Resultset object. We can process this data.

vii) Close/terminate the objects:
ex.
rs.close();
stmt.close();
conn.close();

- JDBC Drivers:



Teacher's Sign.: _____

Page No.
Date

* Multithreading in Java:

Different phases in Thread life cycle:

- Newborn: New thread is created
- Running: Thread is running on processor core
- Runnable: Thread is waiting for access of core
- Blocked: Thread is suspended
- Dead: Execution of thread is stopped.

Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part is called a thread.

Threads can be created by using two mechanisms:

1. Extending the Thread class:
   - We create a class that extends the java.lang.Thread class. This class overrides the run() method available in Thread class.
   - A thread begins its lifecycle inside run() method
   - We create an object of our new class and call start() method to start the execution of a method. start() invokes the run method on the Thread object.

ii) Implementing the Runnable interface:
   - We create a new class which implements java.lang.Runnable interface and override run() method. Then we instantiate a Thread object and call start() method on this object.

* ThreadPool in Java:

A threadpool reuses previously created threads to execute

Teacher's Sign.: _____

current tasks & offers a solution to the problem of thread cycle overhead and resource thrashing. Since the thread is already existing when the request access arrives, the delay introduced by thread creation is eliminated, making the application more responsive.

* ExecutorService :

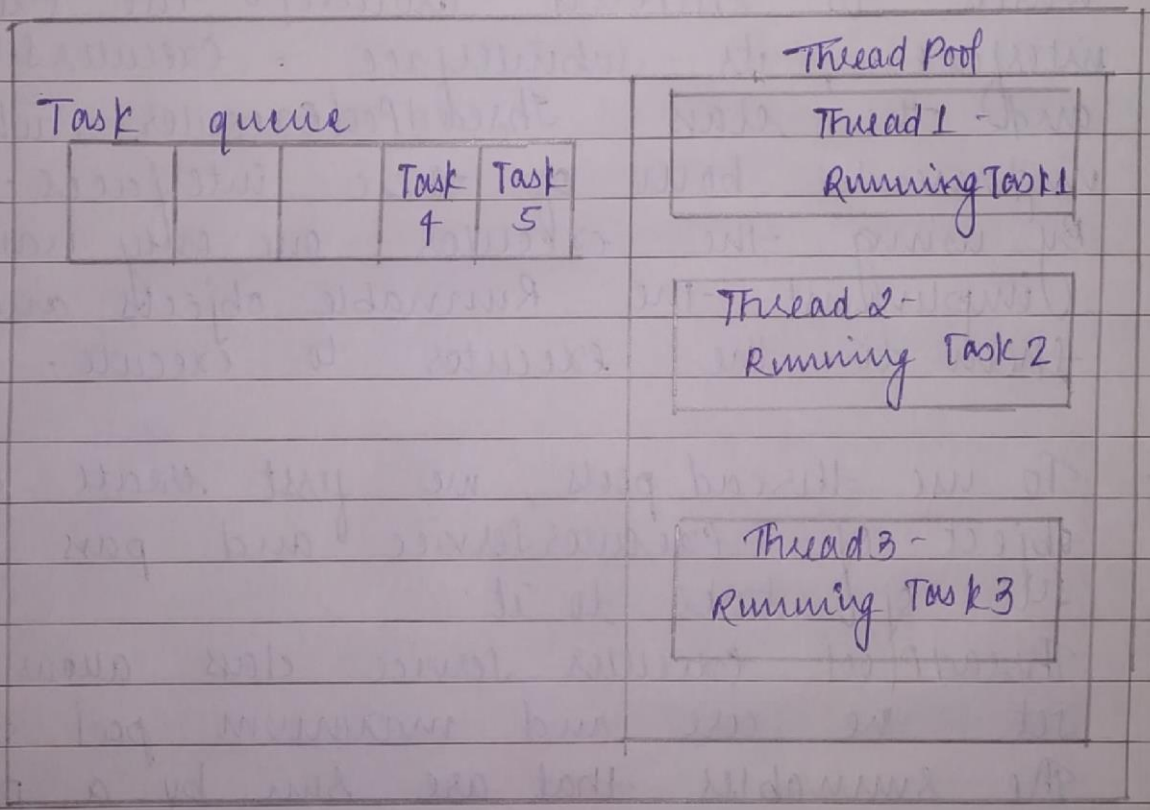
- Java provides the Executor framework which is centered around the Executor interface, its subinterface - ExecutorService and the class ThreadPoolExecutor, which implements both of these interfaces. By using the executor, one only has to implement the Runnable objects and send them to the executor to execute.

- To use thread pools, we first create a object of ExecutorService and pass a set of tasks to it.
ThreadPool executor service class allows to set the core and maximum pool size.
The Runnables that are run by a particular thread are executed sequentially.

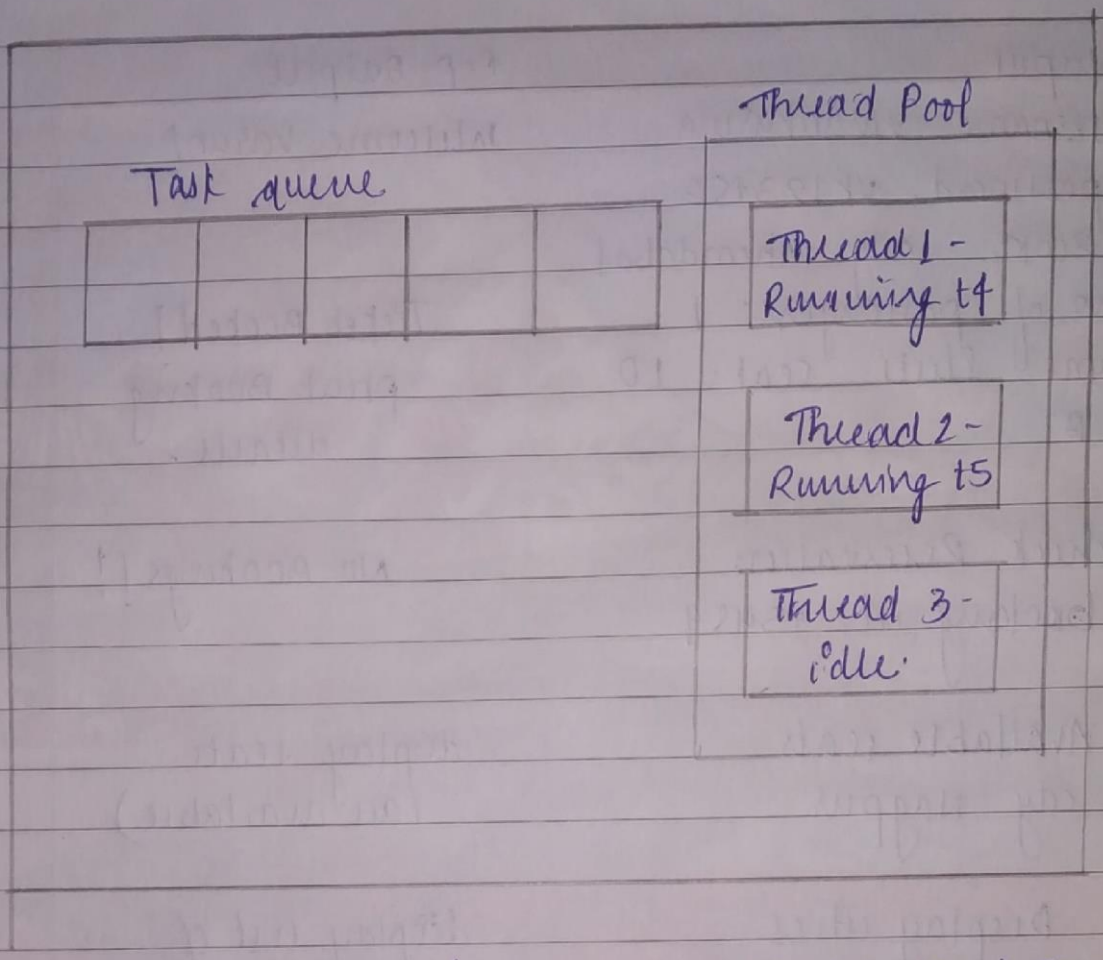
Teacher's Sign.: _____

Executor Thread Pool Methods

- new FixedThreadPool(int) - fixed size thread Pool.
- new CachedThreadPool() - creates new threads as needed but will resume reuse previous ones too.
- new SingleThreadExecutor() - creates a single thread



Thread Pool executing first three tasks.



Thread Pool executing Task 4 and Task 5.

- synchronized ( ) keyword can be used to make sure two threads do not access the same block of code at the same time in order to avoid concurrency issues like deadlock.

Teacher's Sign.: _____

Page No.
Date

* TEST CASES:

	Exp. Output	Result
1. Input: username: varunkarwa password: VK123456	Welcome Varun!	Success
i) Book: city: Ahemdabad no of passengers: 1. Name: Stuti, seat = 10	Ticket Booked! print Booking details.	Success
ii) check Reservation booking id: 95459	No Bookings!!	Success
iii) Available seats city Nagpur	display seats (all available)	Success
iv) Display cities	display list of cities	Success.
2) input: username: tejas012 password: tej1302	Welcome tejas! (Admin)	Success.
i) Add city: Kolhapur	Added!	Success
ii) Delete city: Solapur	Deleted!	Success
iii) Add Bus city: Ahemdabad	Added!	Success
iv) Delete Bus city: Nashik Bus_id: 104	Deleted!	Success.

* CONCLUSION: Thus we implemented database connectivity using JDBC, multithreading & threadpools using executor service in our application (Bus Reservation System).

Teacher's Sign.: _____