

# گزارشکار پروژه نهایی

## درس اصول طراحی پایگاه داده ها

تهیه کننده : اشکان مقرب رازلیقی

شماره دانشجویی : ۹۸۲۳۰۸۱

نام استاد : دکتر حمیدرضا شهریار

### توجه :

این پروژه در قالب client server پیاده سازی شده است اما در گزارشکار به دلیل همپوشانی بخش server و client و وضوح بیشتر کد بخش server ، تنها این بخش توضیح داده شده است .

## Server Class

```
package Server;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.sql.*;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class Server {

    public static void main(String[] args) {
        ExecutorService pool = Executors.newCachedThreadPool();
        try (ServerSocket welcomingSocket = new ServerSocket( port: 7688)) {
            System.out.println("Server started");
            int count = 0;
            while (count != 20) {
                count++;
                Socket connectionSocket = welcomingSocket.accept();
                pool.execute(new ClientHandler(connectionSocket));
            }
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

توضیحات : در این کلاس برای هر کاربر (تا سقف ۲۰ کاربر) که به سرور وصل شود، یک Socket در نظر گرفته می شود و سپس این socket به کلاس Client Handler داده می شود .

توضیحات کلاس Client Handler پس از بیان توضیحات کلاس های Services و AuthenticationService گفته خواهد شد .

## AuthenticationService Class

```
private void act() {
    Label:
    while (true) {
        if (!ID.equals("")) {
            break;
        }
        String msg = "Welcome " + "\n" + "1:sign up" + '\n' + "2:Sign in" + '\n' + "3:recover password" + '\n' + "4:Quit";
        try {
            out = connectionSocket.getOutputStream();
            in = connectionSocket.getInputStream();
            usefullmethods.send_message(out, msg);
            msg = usefullmethods.read_message(in);
            switch (msg) {
                case "1":
                    signUp();
                    break;
                case "2":
                    signIn();
                    break;
                case "3":
                    recover_password();
                    break;
                default:
                    usefullmethods.send_message(out, string: "Goodbye,coming back soon");
                    Quit = true;
                    break Label:
            }
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

توضیحات : در Constructor این کلاس ، متد act وجود دارد که با هربار ساخت یک شی از این کلاس اجرا می شود.

در متد act ابتدا به کاربریک Menu نمایش داده می شود که گزینه های آن به ترتیب مربوط به ثبت نام ، واردشدن به حساب کاربری ، بازیابی رمز عبور و خروج از سامانه می باشد . سپس باتوجه به انتخاب کاربر، یا به یکی از مندهای مربوطه خواهیم رفت یا از سامانه خارج می شویم .

## signUp

```
usefulmethods.send_message(out, string: "what is your favorite color? ");
answer = usefulmethods.toHexString(usefulmethods.getSHA(usefulmethods.read_message(in)));
String st = "INSERT INTO user(ID, firstname, lastname, phone, email, password, answer) VALUES(?, ?, ?, ?, ?, ?, ?)";
PreparedStatement ps = connection.prepareStatement(st);
ps.setString( parameterIndex: 1, ID);
ps.setString( parameterIndex: 2, firstname);
ps.setString( parameterIndex: 3, lastname);
ps.setString( parameterIndex: 4, phone);
ps.setString( parameterIndex: 5, email);
ps.setString( parameterIndex: 6, password);
ps.setString( parameterIndex: 7, answer);
ps.execute();
usefulmethods.send_message(out, ("Hi " + ID + ". your signup successfully done"));
```

بخشی از متد signUp

توضیحات : در این متد نام، نام خانوادگی، شماره همراه(محدودیت شروع با ۰۹ ، ۱۱ رقمی و غیرتکراری بودن برآن اعمال شده)، ایمیل(محدودیت غیرتکراری بودن بر آن اعمال شده)، آیدی (محدودیت غیرتکراری بودن بر آن اعمال شده)، رمز عبور(محدودیت ترکیب حروف و عدد بودن برآن اعمال شده وبه صورت هش شده درمی آید) و سوال امنیتی(به صورت هش شده درمی آید) از کاربر گرفته می شود و در پایگاه داده ذخیره می شود .

## signIn

```
ID = usefulmethods.read_message(in);
if (!input_is_valid(ID, flag: 3) && !ID.toLowerCase().endsWith("deleted")) {
    flag = 1;
    usefulmethods.send_message(out, string: "true");
}
if (flag == 1) {
    String st = "select * from accesslimitation where accesslimited = ?";
    PreparedStatement ps = connection.prepareStatement(st);
    ps.setString( parameterIndex: 1, ID);
    ResultSet rs = ps.executeQuery();
    int flg = 0;
    int flg2 = 0;
    LocalDateTime endTime = LocalDateTime.now();
    if (!rs.next()) {
        flg = 1;
    } else {
        endTime = LocalDateTime.parse(rs.getString( columnLabel: "end_time"));
        if (endTime.isBefore(LocalDateTime.now())) {
            flg = 1;
            st = "delete from accesslimitation where accesslimited = ?";
            ps = connection.prepareStatement(st);
            ps.setString( parameterIndex: 1, ID);
            ps.execute();
        }
    }
}
```

توضیحات : در این بخش از این متد ، چک می شود که آیدی که کاربر برای ورود به حساب کاربری اش وارد کرده معتبر است یا نه ؟ و اگر بود ، دارای محدودیت دسترسی است یا نه ؟

```
st = "select * from log where logged = ? and type = 'login' and enable = 1";
ps = connection.prepareStatement(st);
ps.setString( parameterIndex 1, ID);
rs = ps.executeQuery();
if(!rs.next()){
    flg2 = 1;
}
```

توضیحات : در این بخش از این متد ، چک می شود که کاربر از جایی دیگر وارد حساب کاربری خود نشده باشد .

```
do {
    usefuleMethods.sendMessage(out, string: "password :");
    password = usefuleMethods.toHexString(usefuleMethods.getSHA(usefuleMethods.read_message(in)));
    if (check_sth_of_an_id(ID, password, flag 1)) {
        usefuleMethods.sendMessage(out, string: "true");
        usefuleMethods.sendMessage(out, string: "Hi " + ID + ". welcome to your account");
        this.ID = ID;
        st = "insert into 'messenger'.log ('logged', 'arrival_time', 'type') VALUES (?, ?, ?)";
        ps = connection.prepareStatement(st);
        ps.setString( parameterIndex 1, ID);
        ps.setString( parameterIndex 2, LocalDateTime.now().toString());
        ps.setString( parameterIndex 3, "login");
        ps.execute();
        break;
    } else {
        count++;
        usefuleMethods.sendMessage(out, string: "false");
        st = "insert into 'messenger'.log ('logged', 'arrival_time', 'type') VALUES (?, ?, ?)";
        ps = connection.prepareStatement(st);
        ps.setString( parameterIndex 1, ID);
        ps.setString( parameterIndex 2, LocalDateTime.now().toString());
        ps.setString( parameterIndex 3, "wrong password");
        ps.execute();
        usefuleMethods.sendMessage(out, string: "incorrect password!" + "\n" + "1:continue attempting" + "\n" + "2: back");
        if (usefuleMethods.read_message(in).equals("2")) {
            break;
        } else {
            if (count == 3) {
                st = "insert into 'messenger'.accesslimitation ('accesslimited', 'end_time') VALUES (?, ?)";
                String temp;
                ps = connection.prepareStatement(st);
                ps.setString( parameterIndex 1, ID);
                ps.setString( parameterIndex 2, temp = LocalDateTime.now().plusDays(1).toString());
                ps.execute();
            }
        }
    }
}
```

توضیحات : در این بخش از این متد ، پس از اینکه مطمئن شدیم کاربر محدودیت دسترسی ندارد و از جای دیگری وارد حساب کاربری اش نشده است ، از او رمز عبورش را می گیریم در صورتی که درست بود ، در جدول logged ورودش را درج می کنیم و در غیر این صورت تا ۳ بار به او اجازه وارد کردن مجدد رمز عبورش را می دهیم . (هر بار اشتباه وارد کردن رمز عبور در جدول logged درج می شود).

در صورتی که تا ۳ بار موفق به درست وارد کردن رمز عبورش نشد ، او را به مدت یک روز از دسترسی به حساب کاربری اش منع می کنیم و این را در جدول accesslimited درج می کنیم .

## recover password

```
while (flag == 0) {
    usefullmethods.send_message(out, string: "ID: ");
    ID = usefullmethods.read_message(in);
    if (!input_is_valid(ID, flag: 3) && !ID.ToLowerCase().endsWith("deleted")) {
        flag = 1;
        usefullmethods.send_message(out, string: "true");
    }
}

if (flag == 1) {
    String st = "select * from recoverpasswordlimitation where recoverpasswordlimited = ?";
    PreparedStatement ps = connection.prepareStatement(st);
    ps.setString(1, ID);
    ResultSet rs = ps.executeQuery();
    int flg = 0;
    LocalDateTime endTime = LocalDateTime.now();
    if (rs.next()) {
        flg = 1;
    } else {
        endTime = LocalDateTime.parse(rs.getString("end_time"));
        if (endTime.isBefore(LocalDateTime.now())) {
            flg = 1;
            st = "delete from recoverpasswordlimitation where recoverpasswordlimited = ?";
            ps = connection.prepareStatement(st);
            ps.setString(1, ID);
            ps.execute();
        }
    }
}

if (flg == 1) {
    usefullmethods.send_message(out, string: "true");
    int count = 0;
    do {
        usefullmethods.send_message(out, string: "security Question : what is your favorite color ?");
        String answer = usefullmethods.toHexString(usefullmethods.getSHA(usefullmethods.read_message(in)));
        if (check_sth_of_an_id(ID, answer, flag: 2)) {
            usefullmethods.send_message(out, string: "true");
        }
    } while (count < 3);
}
```

توضیحات : در این بخش از این متد ، چک می شود که آیدی که کاربر برای بازیابی رمز عبورش وارد کرده معتبر است یا نه ؟ و اگر بود ، دارای محدودیت بازیابی رمز عبور است یا نه ؟

```

usefulmethods.send_message(out, string: "security Question : what is your favorite color ?");
String answer = usefulmethods.toHexString(usefulmethods.getSHA(usefulmethods.read_message(in)));
if (check_sth_of_an_id(ID, answer, flag: 2)) {
    usefulmethods.send_message(out, string: "true");
    while (true) {
        usefulmethods.send_message(out, string: "new password :");
        password = usefulmethods.read_message(in);
        if (check_password(password)) {
            usefulmethods.send_message(out, string: "true");
            password = usefulmethods.toHexString(usefulmethods.getSHA(password));
            st = "UPDATE 'messenger'. 'user' SET 'password' = ? WHERE ('ID' = ?)";
            ps = connection.prepareStatement(st);
            ps.setString( parameterIndex: 1, password);
            ps.setString( parameterIndex: 2, ID);
            ps.execute();
            st = "insert into 'messenger'. 'log' ('logged', 'arrival_time', 'type') VALUES (?, ?, ?)";
            ps = connection.prepareStatement(st);
            ps.setString( parameterIndex: 1, ID);
            ps.setString( parameterIndex: 2, LocalDateTime.now().toString());
            ps.setString( parameterIndex: 3, "change password");
            ps.execute();
            usefulmethods.send_message(out, string: "your password successfully changed");
            break;
        }
        usefulmethods.send_message(out, string: "false");
        usefulmethods.send_message(out, ("password should be complex of letters and numbers" + "\n" + "1:continue attempting"));
        if (usefulmethods.read_message(in).equals("2")) {
            break;
        }
    }
    break;
} else {
    count++;
    usefulmethods.send_message(out, string: "false");
    usefulmethods.send_message(out, string: "incorrect answer!" + "\n" + "1:continue attempting" + "\n" + "2: back");
    if (usefulmethods.read_message(in).equals("2")) {
        break;
    }
    if (count == 5) {
        st = "insert into 'messenger'. 'recoverpasswordlimited' ('recoverpasswordlimited', 'end_time') VALUES (?, ?)";
        ps = connection.prepareStatement(st);
        String temp;
        ps.setString( parameterIndex: 1, ID);
        ps.setString( parameterIndex: 2, temp = LocalDateTime.now().plusDays(1).toString());
        ps.execute();
        usefulmethods.send_message(out, string: "too many attempts!" + " " + "recover password for your account limited until " + temp);
        break;
    }
}
} while (true);

```

توضیحات : در این بخش از این متد ، سوال امنیتی پیش فرض از کاربر پرسیده می شود و در صورتی که درست جواب داد ، از او رمز عبور جدید گرفته می شود (با بررسی ترکیب حروف و عدد بودن آن، در جدول logged تغییر رمز عبورش را درج می کنیم) . در غیر این صورت تا ۵ بار به او اجازه داده می شود به سوال امنیتی به درستی جواب دهد و اگر نتوانست، او را به مدت ۱ روز از بازایی رمز عبورش منع می کنیم و این را در جدول recoverpasswordlimited درج می کنیم .

## Services Class

```
try {
    out = connectionSocket.getOutputStream();
    in = connectionSocket.getInputStream();
    while (true) {
        usefulmethods.send_message(out,
            string: "Welcome to services" + "\n" + "1:block" + '\n' + "2:unlock" + '\n' +
            "3:request friendship" + '\n' + "4:requests to me" + '\n' +
            "5:accept or ignore requests" + '\n' + "6:show friends" + '\n' +
            "7:remove friend" + '\n' + "8:send message" + '\n' + "9:show received messages" +
            '\n' + "10:like message" + '\n' + "11:search ID of users" + '\n' + "12:delete account" +
            '\n' + "13:back");
        select = usefulmethods.read_message(in);
        if (select.equals("1") || select.equals("2")) {
            block_unlock();
        } else if (select.equals("3")) {
            request_friendship();
        } else if (select.equals("4")) {
            requests_to_me();
        } else if (select.equals("5")) {
            accept_or_ignore_requests();
        } else if (select.equals("6")) {
            show_friends();
        }
        else if (select.equals("7")) {
            remove_friend();
        }
        else if (select.equals("8")) {
            send_message();
        }
    }
}
```

توضیحات : در Constructor این کلاس ، متد act وجود دارد که با هربارساخت یک شی از این کلاس اجرا می شود.

در متد act ابتدا به کاربریک Menu نمایش داده می شود که گزینه های آن به ترتیب مربوط به مسدودکردن کاربران، رفع مسدودیت، درخواست دوستی، دریافت درخواست های داده شده به کاربربرای دوستی، موافقت یا عدم موافقت با یک درخواست، دریافت لیست دوستان، حذف از لیست دوستان، ارسال پیام، لیست پیام های دریافتی، لایک یک پیام، امکان پیداکردن افراد، حذف حساب کاربری و بازگشت به عقب می باشد. سپس باتوجه به انتخاب کاربر، یا به یکی از متدهای مربوطه خواهیم رفت یا به عقب بازمی گردیم .



## block unblock

```
usefulMethods.sendMessage(out, msg, "ID of user you want to block: ");
} else {
    usefulMethods.sendMessage(out, msg, "ID of user you want to unblock: ");
}
blocked_unblocked = usefulMethods.read_message(in);
if (input_exists(blocked_unblocked) && !blocked_unblocked.toLowerCase().endsWith("deleted")) {
    flag = 1;
    usefulMethods.sendMessage(out, msg, "true");
}
if (flag == 1) {
    String st = "select * from block where blocker = ? and blocked = ?";
    PreparedStatement ps = connection.prepareStatement(st);
    ps.setString(1, ID);
    ps.setString(2, blocked_unblocked);
    ResultSet rs = ps.executeQuery();
    if (rs.next()) {
        if (select.equals("1")) {
            usefulMethods.sendMessage(out, msg, "you blocked this user previously");
        } else {
            st = "delete from block where blocker = ? and blocked = ?";
            ps = connection.prepareStatement(st);
            ps.setString(1, ID);
            ps.setString(2, blocked_unblocked);
            ps.execute();
            usefulMethods.sendMessage(out, msg, "unlocking successfully done");
        }
    }
} else {
    if (select.equals("1")) {
        if (ID.equals(blocked_unblocked)) {
            usefulMethods.sendMessage(out, msg, "you can not block yourself!");
        } else {
            st = "insert into messenger_block (blocker, blocked) VALUES (?, ?)";
            ps = connection.prepareStatement(st);
            ps.setString(1, ID);
            ps.setString(2, blocked_unblocked);
            ps.execute();
            usefulMethods.sendMessage(out, msg, "blocking successfully done");
        }
    }
}
```

توضیحات : در این متد، ابتدا آیدی فردی که کاربر می خواهد او را مسدود یا رفع مسدود کند گرفته می شود . پس از بررسی موجود بودن چنین آیدی ، در جدول بلاک می گردیم که آیا کاربر، این فرد را قبلا مسدود کرده است یا خیر. در صورتی که قبلا مسدود کرده باشد و درخواست مسدود کردن مجدد بدهد، به او اطلاع می دهیم که قبلا این کاربر را مسدود کرده است و در صورتی که درخواست رفع مسدودیت بدهد، آیدی کاربر و فرد مسدود شده را از جدول بلاک حذف می کنیم .

در صورتی که قبلا مسدود نکرده باشد و درخواست مسدود کردن بدهد، آیدی کاربر و فرد مسدود شده را در جدول بلاک درج می کنیم و در صورتی که درخواست رفع مسدودیت بدهد، به او اطلاع می دهیم که قبلا این کاربر را مسدود نکرده است .

## request friendship

```
usefulMethods.sendMessage(out, msg, "don't need to request.");
requested = usefulMethods.read_message(in);
if (input_exists(requested) && !requested.toLowerCase().endsWith("deleted")) {
    flag = 1;
    usefulMethods.sendMessage(out, msg, "true");
}
if (flag == 1) {
    String st = "select * from friendship where accepted = ? and acceptor = ?";
    PreparedStatement ps = connection.prepareStatement(st);
    ps.setString(1, ID);
    ps.setString(2, requested);
    ResultSet rs = ps.executeQuery();
    if (rs.next()) {
        usefulMethods.sendMessage(out, msg, "you are a friend of this user,don't need to request friendship");
    } else {
        if (ID.equals(requested)) {
            usefulMethods.sendMessage(out, msg, "you can not request friendship to yourself!");
        } else {
            st = "select * from block where blocker = ? and blocked = ?";
            ps = connection.prepareStatement(st);
            ps.setString(1, requested);
            ps.setString(2, ID);
            rs = ps.executeQuery();
            if (rs.next()) {
                usefulMethods.sendMessage(out, msg, "you are blocked and can not request friendship to this user");
            } else {
                st = "select * from request where requestor = ? and requested = ?";
                ps = connection.prepareStatement(st);
                ps.setString(1, ID);
                ps.setString(2, requested);
                rs = ps.executeQuery();
                if (rs.next()) {
                    usefulMethods.sendMessage(out, msg, "you requested friendship to this user previously!");
                } else {
                    st = "insert into messenger.request (requestor, requested) VALUES (?, ?)";
                    ps = connection.prepareStatement(st);
                    ps.setString(1, ID);
                    ps.setString(2, requested);
                    ps.executeUpdate();
                    usefulMethods.sendMessage(out, msg, "request friendship successfully done");
                }
            }
        }
    }
}
```

توضیحات : در این متد، ابتدا آیدی فردی که کاربر می خواهد به او درخواست دوستی دهد گرفته می شود. پس از بررسی موجود بودن چنین آیدی، با استفاده از جدول friendship بررسی می کنیم که آیا این کاربر دوست آن فرد بوده است یا خیر. در صورتی که بوده باشد، به او اطلاع می دهیم که دوست آن فرد است و نیاز به درخواست دوستی نیست. در غیر این صورت، چک می کنیم که آن فرد، کاربر را مسدود نکرده باشد. در صورتی که مسدود کرده بود، به او اطلاع می دهیم که نمی تواند درخواست دوستی دهد و در غیر این صورت چک می کنیم که کاربر قبلاً درخواست دوستی دیگری را ثبت نکرده باشد. در صورت ثبت نشدن درخواست دیگری، درخواست کاربر در جدول request درج می شود.

## requests to me

```
private void requests_to_me() {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
    } catch (ClassNotFoundException cnfe) {
        System.out.println("Error loading driver");
    }

    try {
        Connection connection = DriverManager.getConnection(
            "jdbc:mysql://localhost",
            "user: \"ashkan1377\", password: \"a8c.123456\"");
        Statement statement = connection.createStatement();
        statement.execute("use messenger");
        String st = "select requestor from request where requested = ?";
        PreparedStatement ps = connection.prepareStatement(st);
        ps.setString(1, ID);
        ResultSet rs = ps.executeQuery();
        ArrayList<String> requestors = new ArrayList<>();
        while (rs.next()) {
            requestors.add(rs.getString("requestor"));
        }
        usefulethods.send_message(out, "string: " + requestors.size());
        for (String requestor : requestors) {
            usefulethods.send_message(out, requestor);
            Thread.sleep(1);
        }
    }
}
```

توضیحات: در این متد، با استفاده از جدول request آیدی افرادی که درخواست دوستی به کاربر را داده اند ، به او نمایش داده می شود.

## accept or ignore requests

```
usefulMethods.sendMessage(out, string: "ID of user you want to accept or ignore its request: ");
judged = usefulMethods.read_message(in);
if (input_exists(judged) && !judged.toLowerCase().endsWith("deleted")) {
    flag = 1;
    usefulMethods.sendMessage(out, string: "true");
}
if (flag == 1) {
    usefulMethods.sendMessage(out, string: "your choice?" + '\n' + "1:accept" + '\n' + "2:ignore");
    choice = usefulMethods.read_message(in);
    String st = "select requestor from request where requested = ? and requestor = ?";
    PreparedStatement ps = connection.prepareStatement(st);
    ps.setString( parameterIndex 1, ID);
    ps.setString( parameterIndex 2, judged);
    ResultSet rs = ps.executeQuery();
    if (!rs.next()) {
        usefulMethods.sendMessage(out, string: "this user didn't request to you any way");
    } else {
        st = "delete from request where requested = ? and requestor = ?";
        ps = connection.prepareStatement(st);
        ps.setString( parameterIndex 1, ID);
        ps.setString( parameterIndex 2, judged);
        ps.execute();
        if (choice.equals("1")) {
            st = "insert into friendship values (?,?)";
            ps = connection.prepareStatement(st);
            ps.setString( parameterIndex 1, judged);
            ps.setString( parameterIndex 2, ID);
            ps.execute();
            usefulMethods.sendMessage(out, string: "accept this request successfully done");
        }
        else {
            usefulMethods.sendMessage(out, string: "ignore this request successfully done");
        }
    }
}
```

توضیحات : در این متد، ابتدا آیدی فردی که کاربر می خواهد درخواست دوستی اش را رد یا قبول کند، گرفته می شود . پس از بررسی موجود بودن چنین آیدی، چک می شود که آیا همچنین فردی درخواستی به کاربر داده است یا خیر. در صورتی که درخواست داده باشد، از جدول request درخواست او حذف می شود و اگر کاربر با این درخواست موافقت کرده باشد، آیدی کاربر و آن فرد در جدول friendship درج می شود.

## show friends

```
private void show_friends() {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
    } catch (ClassNotFoundException cnfe) {
        System.out.println("Error loading driver");
    }
    try {
        Connection connection = DriverManager.getConnection(
            url, "jdbc:mysql://localhost",
            user, "ashkan1377", password: "aBc.123456");
        Statement statement = connection.createStatement();
        statement.execute("use messenger");
        String st = "select accepted from friendship where acceptor=?";
        PreparedStatement ps = connection.prepareStatement(st);
        ps.setString(1, ID);
        ResultSet rs = ps.executeQuery();
        ArrayList<String> friends = new ArrayList<>();
        while(rs.next()){
            friends.add(rs.getString(1));
        }
        usefullMethods.send_message(out, " " + friends.size());
        for(String friend : friends){
            usefullMethods.send_message(out, friend);
            Thread.sleep(1000);
        }
    }
    catch (SQLException | InterruptedException e1) {
        e1.printStackTrace();
    }
}
```

توضیحات : در این متد، با استفاده از جدول friendship آیدی افرادی که دوست کاربر هستند، به او نمایش داده می شود.

## Remove friend

```
usefullMethods.send_message(out, "ID of friend you want to remove: ");
removed= usefullMethods.read_message(in);
if (input_exists(removed) && !removed.toLowerCase().endsWith("deleted")) {
    flag = 1;
    usefullMethods.send_message(out, "true");
}
if (flag == 1) {
    String st = "select accepted from friendship where accepted = ? and acceptor = ?";
    PreparedStatement ps = connection.prepareStatement(st);
    ps.setString(1, removed);
    ps.setString(2, ID);
    ResultSet rs = ps.executeQuery();
    if (!rs.next()) {
        usefullMethods.send_message(out, "this user isn't your friend");
    } else {
        st = "delete from friendship where accepted = ? and acceptor = ?";
        ps = connection.prepareStatement(st);
        ps.setString(1, removed);
        ps.setString(2, ID);
        ps.execute();
        usefullMethods.send_message(out, "this user removed from your friends successfully");
    }
} else {
    usefullMethods.send_message(out, "false");
    usefullMethods.send_message(out, "There is no user with this username" + "\n" + "1: continue attempting" + "\n" + "2: back");
    if (usefullMethods.read_message(in).equals("2")) {
        break;
    }
}
```

توضیحات: در این متد، ابتدا آیدی دوستی که کاربر می خواهد حذف کند، گرفته می شود.

پس از بررسی موجود بودن چنین آیدی، با استفاده از جدول friendship چک می شود که آیا این فرد دوست کاربر بوده است یا خیر. در صورتی که بوده باشد از جدول friendship تاپلی که مربوط به آیدی کاربر و آن فرد است حذف می شود.

## send\_message

```
while (flag == 0) {
    usefulMethods.send_message(out, msg, "ID of friend you want to send message: ");
    receiver = usefulMethods.read_message(in);
    if (input_exists(receiver) && !receiver.toLowerCase().endsWith("deleted")) {
        flag = 1;
        usefulMethods.send_message(out, msg, "true");
    }
}

if (flag == 1) {
    usefulMethods.send_message(out, msg, "text:");
    text = usefulMethods.read_message(in);
    String st = "select * from friendship where accepted = ? and acceptor = ?";
    PreparedStatement ps = connection.prepareStatement(st);
    ps.setString(1, ID);
    ps.setString(2, receiver);
    ResultSet rs = ps.executeQuery();
    if (!rs.next()) {
        usefulMethods.send_message(out, msg, "you are not a friend of this user and can not send message");
    } else {
        if (ID.equals(receiver)) {
            usefulMethods.send_message(out, msg, "you can not send message to yourself!");
        } else {
            st = "select * from block where blocker = ? and blocked = ?";
            ps = connection.prepareStatement(st);
            ps.setString(1, receiver);
            ps.setString(2, ID);
            rs = ps.executeQuery();
            if (rs.next()) {
                usefulMethods.send_message(out, msg, "you are blocked and can not send message to this user");
            } else {
                st = "INSERT INTO `messenger`.`message` (`text`, `sender`, `receiver`, `send_time`, `status`) VALUES (?, ?, ?, ?, ?)";
                ps = connection.prepareStatement(st);
                ps.setString(1, text);
                ps.setString(2, ID);
                ps.setString(3, receiver);
                ps.setString(4, LocalDateTime.now().toString());
                ps.setString(5, "0");
                ps.execute();
                usefulMethods.send_message(out, msg, "send message successfully done");
            }
        }
    }
}
```

توضیحات: در این متد، ابتدا آیدی دوستی که کاربر می خواهد به او پیام دهد، گرفته می شود. پس از بررسی موجود بودن چنین آیدی، با استفاده از جدول friendship چک می شود که آیا کاربر جزو دوستان آن فرد است یا خیر. در صورتی که باشد، با استفاده از جدول block، چک می شود که آیا آن فرد، کاربر را مسدود کرده است یا خیر. در صورتی که نکرده باشد، متن پیام از کاربر گرفته می شود و در جدول message درج می شود.

## show received messages

```
String st = "select * from message where receiver = ? order by send_time desc";
PreparedStatement ps = connection.prepareStatement(st);
ps.setString(1, ID);
ResultSet rs = ps.executeQuery();
ArrayList<String>messages = new ArrayList<>();
while(rs.next()){
    int flag = 1;
    if(rs.getString("status").equals("0")){
        String st2 = "update message set status = 1 where receiver = ? and ID = ?";
        PreparedStatement ps2 = connection.prepareStatement(st2);
        ps2.setString(1, ID);
        ps2.setString(2, rs.getString("ID"));
        ps2.execute();
        flag = 0;
    }
    if(flag == 0){
        messages.add("ID: " + rs.getString("ID") + "\n" + rs.getString("send_time") + "\n" + rs.getString("text") + " (unread)");
    }
    else{
        messages.add("ID: " + rs.getString("ID") + "\n" + rs.getString("send_time") + "\n" + rs.getString("text"));
    }
}
usefulMethods.sendMessage(out, msg, "" + messages.size());
for(String message : messages){
    usefulMethods.sendMessage(out, message);
    Thread.sleep(1000);
}
```

توضیحات: در این متد، با استفاده از جدول message پیام هایی که دوستان کاربر برای او فرستاده اند به ترتیب نزولی برحسب زمان ارسال ، به او نمایش داده می شود.

باتوجه به اینکه پیامی که برای کاربر نمایش داده می شود قبلا خوانده شده یا خیر برچسب unread برای آن پیام نمایش داده می شود .

## like message

```
usefulMethods.sendMessage(out, msg, "ID of message you want to Like: ");
message_ID = usefulMethods.readMessage(in);

String st = "select * from message where receiver = ? and ID = ?";
PreparedStatement ps = connection.prepareStatement(st);
ps.setString(1, ID);
ps.setString(2, message_ID);
ResultSet rs = ps.executeQuery();
if(rs.next()){
    usefulMethods.sendMessage(out, msg, "you can only like received message!");
}
else{
    st = "select * from liked where user_ID = ? and message_ID = ?";
    ps = connection.prepareStatement(st);
    ps.setString(1, ID);
    ps.setString(2, message_ID);
    rs = ps.executeQuery();
    if(rs.next()){
        usefulMethods.sendMessage(out, msg, "you liked this message previously");
    }
    else{
        st = "insert into liked values (?, ?)";
        ps = connection.prepareStatement(st);
        ps.setString(1, ID);
        ps.setString(2, message_ID);
        ps.execute();
        usefulMethods.sendMessage(out, msg, "Like this message successfully done");
    }
}
```

توضیحات: در این متد، ابتدا آیدی پیامی که کاربر می خواهد لایک کند گرفته می شود .



سپس با استفاده از جدول message، چک می شود که آیا این پیام جزو پیام های دریافتی کاربر هست یا خیر. در صورتی که باشد، با استفاده از جدول liked چک می کنیم که قبلا کاربر این پیام را لایک نکرده باشد. در صورتی که لایک نکرده بود، این لایک را در جدول liked درج می کنیم.

## search

```
private void search() {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
    } catch (ClassNotFoundException cnfe) {
        System.out.println("Error loading driver");
    }
    try {
        Connection connection = DriverManager.getConnection(
            url: "jdbc:mysql://localhost",
            user: "ashkan1377", password: "aBc.123456");
        Statement statement = connection.createStatement();
        statement.execute( sql: "use messenger");
        usefuleMethods.sendMessage(out, string: "word: ");
        String word = usefuleMethods.read_message(in);
        ResultSet rs = statement.executeQuery( sql: "select ID from user");
        ArrayList<String> nears = new ArrayList<>();
        while(rs.next()){
            String st = rs.getString( columnLabel: "ID");
            if(st.startsWith(word) && !st.endsWith("deleted")){
                nears.add(st);
            }
        }
        usefuleMethods.sendMessage(out, string: "" +nears.size());
        for(String near : nears){
            usefuleMethods.sendMessage(out,near);
            Thread.sleep( millis: 1);
        }
    }
    catch (SQLException | InterruptedException e1) {
        e1.printStackTrace();
    }
}
```

توضیحات: در این متد، با استفاده از جدول user آیدی همه کاربرانی که با کلمه وارد شده از سمت کاربر شروع می شوند، نمایش داده می شود.



## delete\_account

```
String st = "Delete from block where blocker = ? or blocked = ?";
PreparedStatement ps = connection.prepareStatement(st);
ps.setString(1, ID);
ps.setString(2, ID);
ps.execute();
st = "Delete from friendship where accepted = ? or acceptor = ?";
ps = connection.prepareStatement(st);
ps.setString(1, ID);
ps.setString(2, ID);
ps.execute();
st = "Delete from recoverpasswordlimitation where recoverpasswordlimited = ?";
ps = connection.prepareStatement(st);
ps.setString(1, ID);
ps.execute();
st = "Delete from request where requestor = ? or requested = ?";
ps = connection.prepareStatement(st);
ps.setString(1, ID);
ps.setString(2, ID);
ps.execute();
st = "UPDATE messenger, user SET phone = ? WHERE ('ID' = ?)";
ps = connection.prepareStatement(st);
ps.setString(1, " " + "deleted");
ps.setString(2, ID);
ps.execute();
st = "UPDATE messenger, user SET email = ? WHERE ('ID' = ?)";
ps = connection.prepareStatement(st);
ps.setString(1, " " + "deleted");
ps.setString(2, ID);
ps.execute();
st = "UPDATE messenger, user SET ID = ? WHERE ('ID' = ?)";
ps = connection.prepareStatement(st);
ps.setString(1, " " + ID + "deleted");
ps.setString(2, ID);
ps.execute();
st = "update log set enable = 0 where logged = ? and type = 'login' and enable = 1";
ps = connection.prepareStatement(st);
ps.setString(1, " " + ID + "deleted");
ps.execute();
enable = 0;
```

توضیحات: در این متد، تاپل هایی از جداول block، friendship، request و recoverpasswordlimitation که آیدی کاربر در آن ها است، پاک می شوند. در جدول user، شماره همراه، ایمیل و آیدی به ترتیب به deleted، deleted و deleted + نام کاربری کاربر تغییر می یابند.

## ClientHandler Class

```
AuthenticationService authenticationService = new AuthenticationService(connectionSocket);
ID = authenticationService.getID();
if(authenticationService.isQuit()){
    connectionSocket.close();
    break;
}
} else {
    select = usefulMethods.read_message(in);
    if (select.equals("1")) {
        Services services = new Services(ID, connectionSocket);
        if (services.getEnable() == 0) {
            usefulMethods.send_message(out, "delete account successfully done:");
            AuthenticationService authenticationService = new AuthenticationService(connectionSocket);
            ID = authenticationService.getID();
            if(authenticationService.isQuit()){
                connectionSocket.close();
                break;
            }
        }
    }
} else {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
    } catch (ClassNotFoundException cnfe) {
        System.out.println("Error loading driver");
    }
    try {
        Connection connection = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/your_database",
            "root", "password");
        Statement statement = connection.createStatement();
        statement.execute("use messenger");
        String st = "update log set enable = 0 where logged = ? and type = 'login' and enable = 1";
        PreparedStatement ps = connection.prepareStatement(st);
        ps.setString(1, ID);
        ps.execute();
        AuthenticationService authenticationService = new AuthenticationService(connectionSocket);
        ID = authenticationService.getID();
        if (authenticationService.isQuit()) {
            connectionSocket.close();
        }
    }
}
```

توضیحات: این کلاس یک متد run دارد که در ابتدا هرکاربر را به یک شی از کلاس AuthenticationService ارجاع می دهد. وقتی کاربر رفت و ثبت نام کرد و وارد حساب کاربری اش شد، او را به یک شی از کلاس services ارجاع می دهد. و در صورتی که یا حساب کاربری اش را حذف کند یا از حساب کاربری اش خارج شود، به یک شی از کلاس AuthenticationService ارجاع می دهد.

## شمای پایگاه داده پروژه

