目录

1 课程设计目的

1. 根据我们在软件工程这门课中所学的知识，我们组队设计相应课题的程序，以此来消化课堂所讲解的内容。

2. 通过调试课题程序来积累调试程序的经验。

3. 通过完成本次课题的项目，逐渐培养我们的 JAVA 编程能力和用计算机解决实际问题的能力。

4. 通过一起协同完善我们课题的程序，培养我们团队协作能力。

5. 运用 C/S 模型来实现主机端和被监控端的通信，加深我们对 CS 模型的理解提高我们对其的运用能力。

6. 通过利用 socket 来沟通 C/S，加深我们对 socket 的理解以及运用能力。

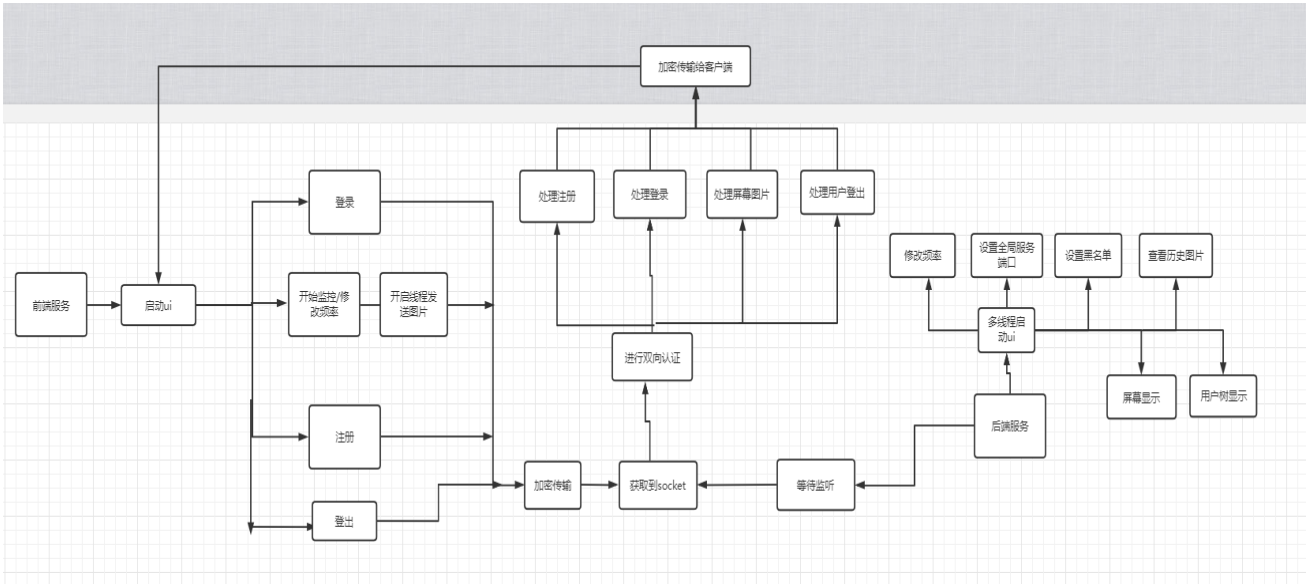
7. 在程序前端设计中，我们不断优化图形化界面，改善使用体验，提升我们 UI 设计的能力。

8. 在课题设计中加入数据库的运用，让我们对后端与数据库交互有了更深入的了解。

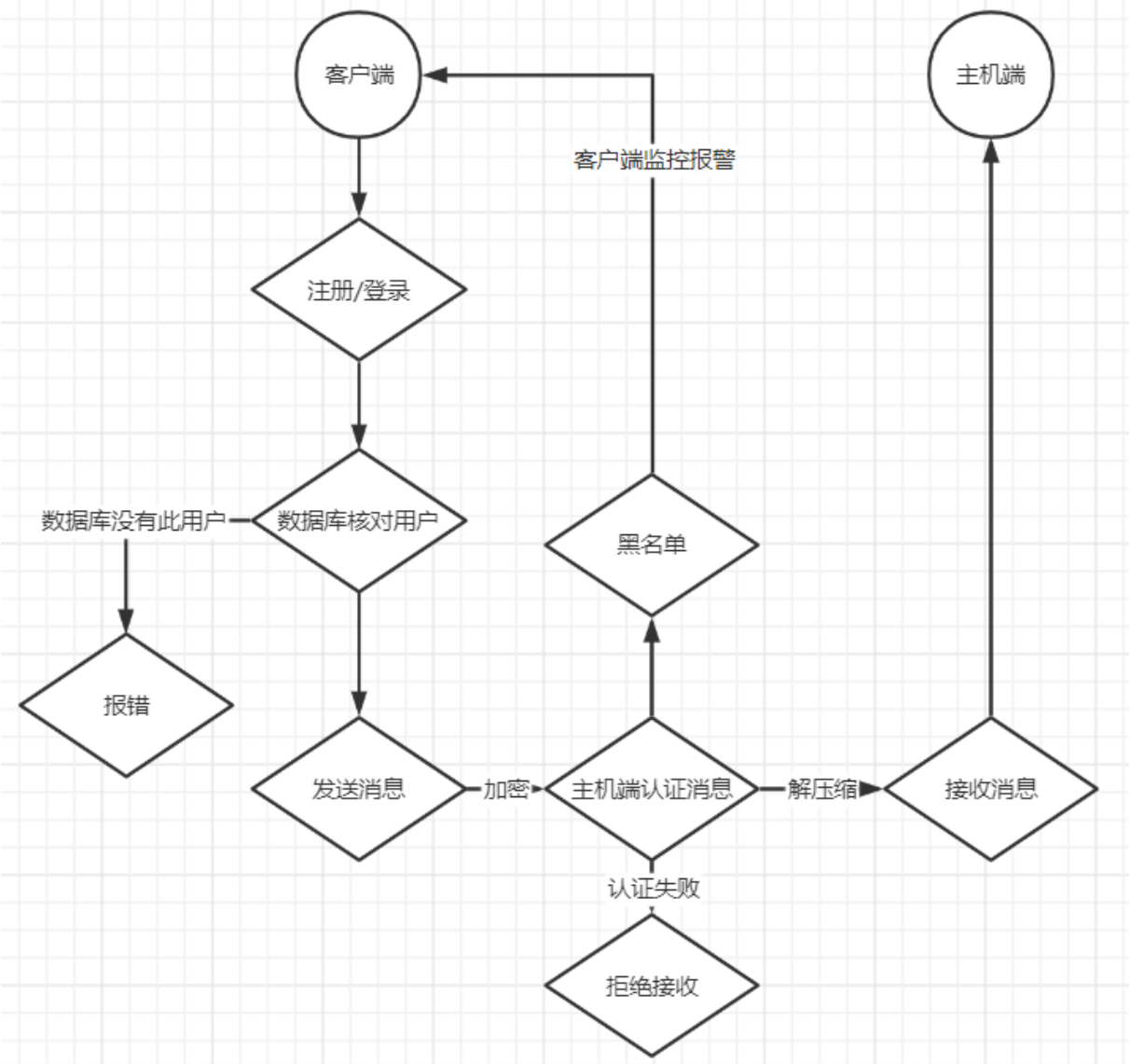
9. 本次课题设计中，通过对消息加密认证，让我们能直观感受加密认证在代码中的形式以及对加密认证有了更加深入的了解。

1.1 需求分析

整体框架：（前后端完全分离）


加密传输给客户端
处理注册
处理登录
处理屏幕图片
处理用户登出
登录
前端服务
启动ui
开始监控/修改频率
开启线程发送图片
修改频率
设置全局服务端口
设置黑名单
查看历史图片
多线程启动ui
进行双向认证
屏幕显示
用户列显示
后端服务
注册
加密传输
获取到socket
等待监听
登出

详细流程图：



1.2 功能需求：

客户端：

1. 客户端登录远程屏幕监控系统，在前端界面输入我们的用户名密码等信息，后台数据库搜索匹配用户信息，各项信息正确后登录成功，假如用户没有账户就可以先注册一个用户，自定义用户名、密码，主机 ip、主机 mac 码等信息自动获取，并且后台数据库录入信息，其中用户密码在前端转换为 MD5 后发送给服务器

并存储，避免明文存储密码。

2. 客户端一定时间间隔便截取屏幕图像。

3. 客户端将截取的图像压缩后发送往主机。

4. 客户端在暂时不用的时候可以最小化托盘管理。

主机端：

1. 与客户端通信进行注册，检验对方是否为自己合法客户。

2. 检验成功后，连续接收客户端截取的屏幕图像并且解压缩。

3. 将接收到的屏幕图像显示在主机屏幕上。

4. 能够对客户端用户树进行及时的维护，对于在线的用户，显示在线，不在线的用户显示离线，按照用户名、ip 地址等分别存储历史屏幕图像，并可按照时间段顺序展示。

5. 下发指令变更监控频率。

高级功能：

1. 双向认证、加密传输：在传输前进行双向认证，只有 c/s 互相确认身份后才可以成功进行通信，并且消息传送过程中将其加密。

2. 黑名单程序告警：假如用户打开服务器端设置黑名单的程序，客户端监控持续弹窗报警，直到。

1.3 性能需求：

1. 响应时间<100ms

2. 消息传输时间<10ms

3. 数据库能容纳一定数量的用户信息

1.4 可靠性需求：

1. 一个月内不能出现三次及以上的故障。

2. 客户端能够短时间连续传输三张以上的图片。

3. 客户端能够对黑名单程序做出及时的报警。

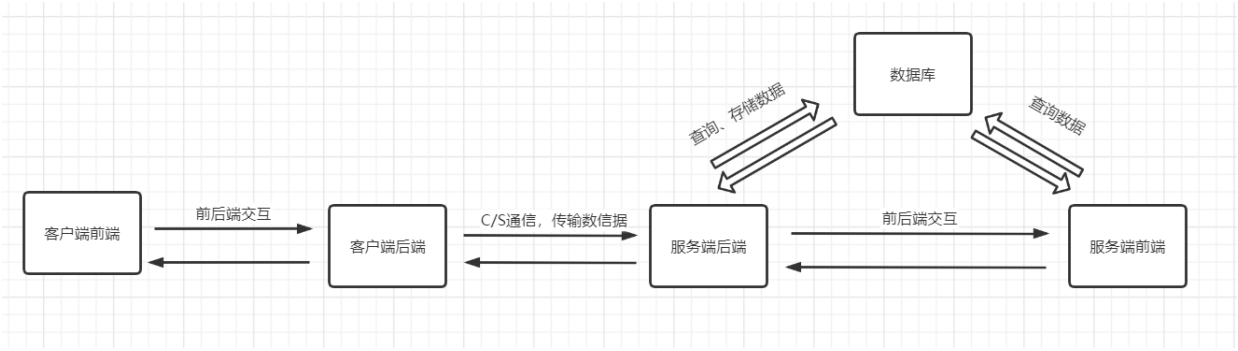
1.5 报错需求：

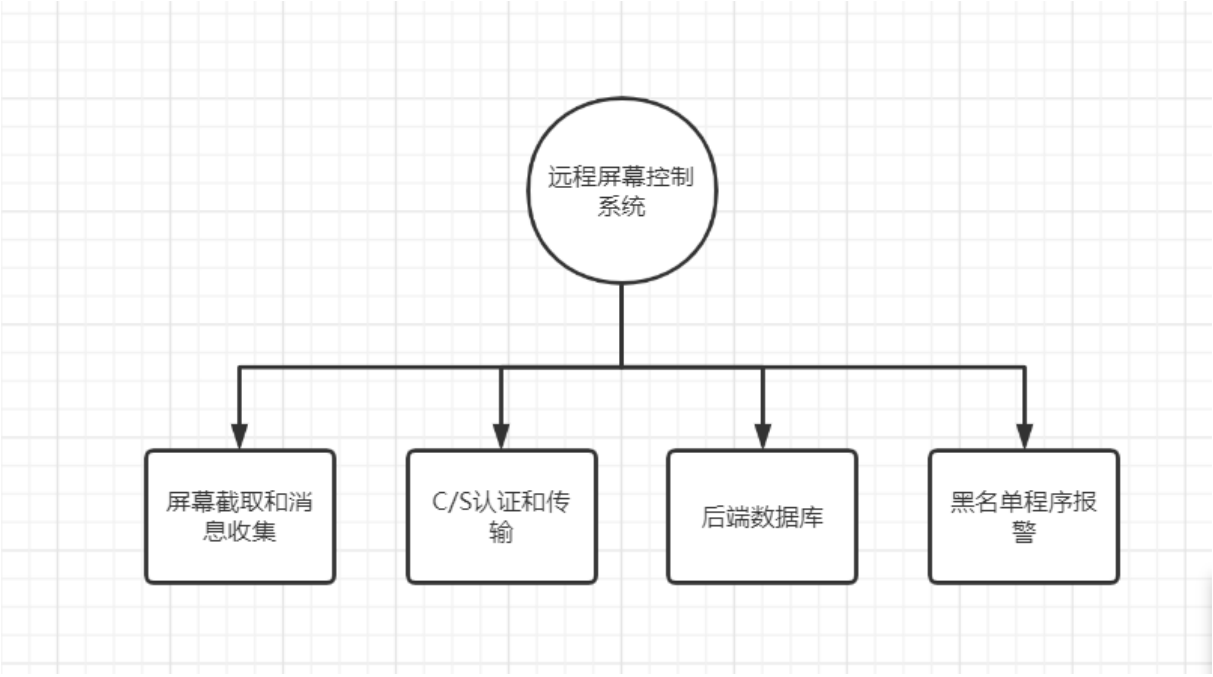
1.用户输入错误信息会报错

2.用户输入错误密码，ip 等会弹窗报错。

2 详细设计

2.1 系统结构设计：





远程屏幕控制系统前端设计各种 UI，后端便是数据库，用来管理用户数据。客户端登录后便又截取部分定时截取客户端屏幕图像，然后又传输部分将消息加密传输。

2.2 模块设计：

（1）客户端：

Client 模块：

客户端模块我们创建了与服务端的连接，然后实现屏幕的截取并且将截取的屏幕图像发送到服务端。将图像信息保存到 user 类并通过序列化方便数据传输。另外还有系统托盘的功能。

建立 SSLsocket 双向认证套接字

```java
/**
 * init
 */
public Client() throws NoSuchAlgorithmException {
    try{
        ctx1 = SSLContext.getInstance("SSL");
        KeyManagerFactory kmf = KeyManagerFactory.getInstance("SunX509");
        TrustManagerFactory tmf = TrustManagerFactory.getInstance("SunX509");

        KeyStore ks = KeyStore.getInstance("JKS");
        KeyStore tks = KeyStore.getInstance("JKS");

        ks.load(new FileInputStream( name: "../data/key/client.keystore"), CLIENT_KEY_STORE_PASSWORD.toCharArray());
        tks.load(new FileInputStream( name: "../data/key/client.keystore"), CLIENT_TRUST_KEY_STORE_PASSWORD.toCharArray());

        kmf.init(ks, CLIENT_KEY_STORE_PASSWORD.toCharArray());
        tmf.init(tks);

        ctx1.init(kmf.getKeyManagers(), tmf.getTrustManagers(), random: null);

        robot = new Robot();
```

模块核心部分：

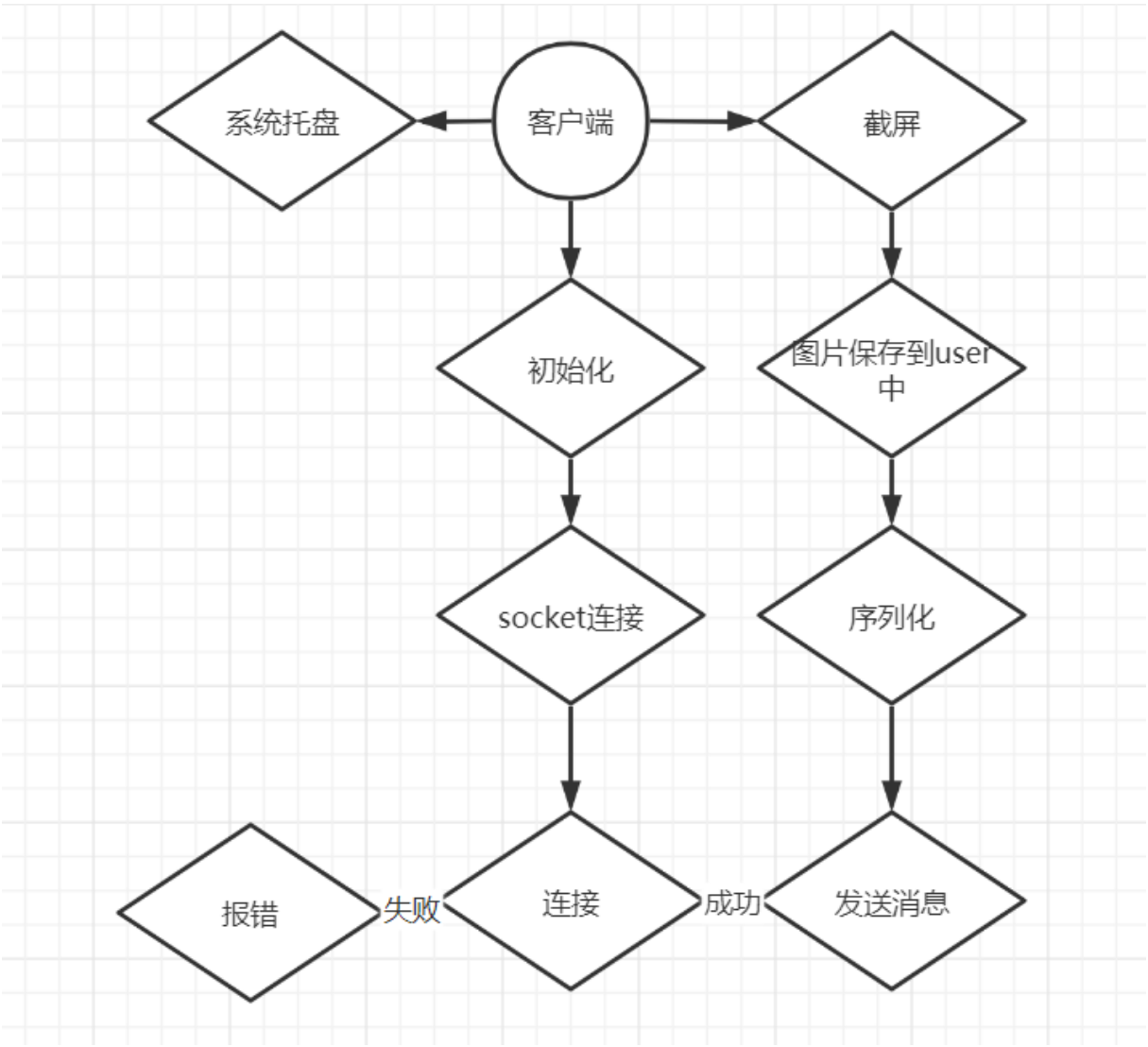
```java
/**
 * socket连接服务端
 * @param user
 * @return 返回map存放socket dos dis
 */
public static HashMap connect(User user) {
    HashMap con = new HashMap();
    try{
        socket = new Socket(user.ServerIP, Integer.parseInt(user.ServerPort));
        con.put("socket", socket);
        con.put("dos", new DataOutputStream(socket.getOutputStream()));
        con.put("dis", new DataInputStream(socket.getInputStream()));
        System.out.println("connected");
        return con;
    } catch (IOException e) {
        e.printStackTrace();
        System.out.println("Socket wrong");
        return null;
    }
}
```

这里是我们创建 SSLsocket 连接部分，会在前端 javafx 线程中调用。

```java
public BufferedImage getScreenShot(){
    return robot.createScreenCapture(new Rectangle( x: 0,  y: 0, screenSize.width, screenSize.height));
}
```

这一部分就是我们截取屏幕图像的代码部分。

然后通过将消息存到 user 中再发给服务端。



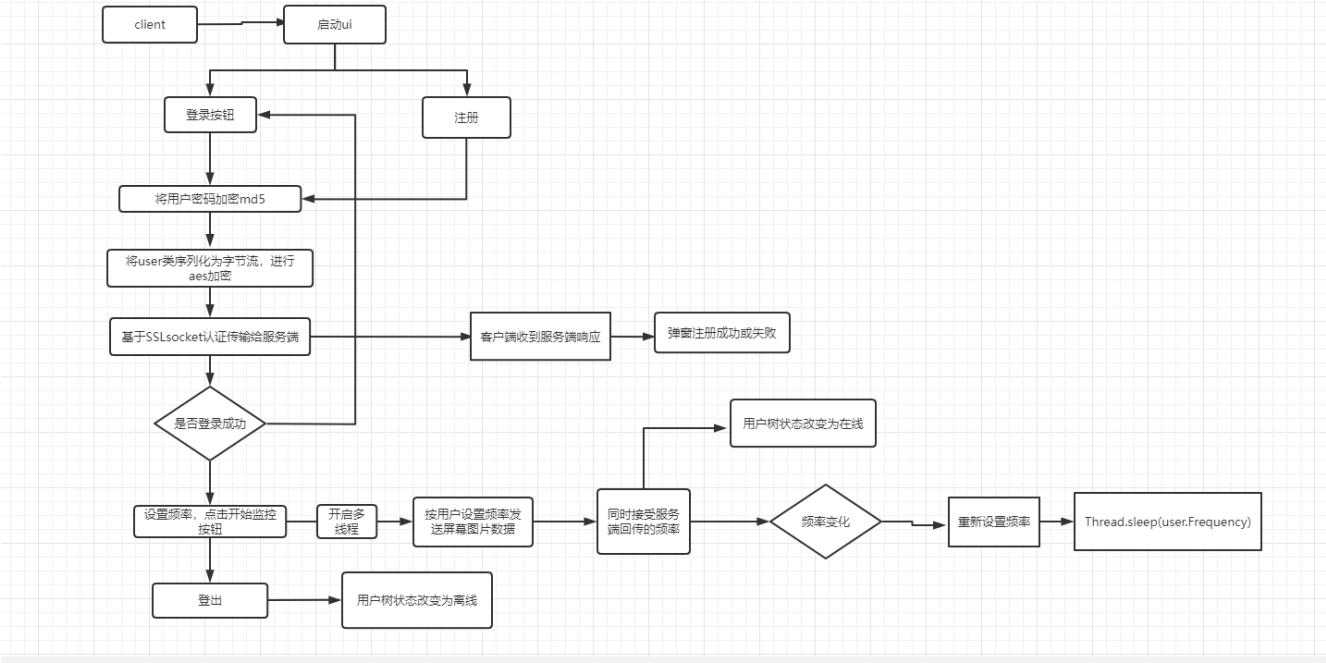
ClientView 模块：

这个类里面定义了我们程序的图形化界面，设置了一系列按钮和标签。以及按钮的一系列点击事件。

功能包括：用户登录、用户登出、用户注册、客户端最小化至托盘、弹出程序黑名单警告、以及相关提示。

我们看到我们客户端界面的图形化设计都定义在这个类中，例如账户、用户名等属性。还有我们界面的一些按钮，点击会发生什么，达成什么样的效果都是定义在这个类中。

客户端核心功能总体设计



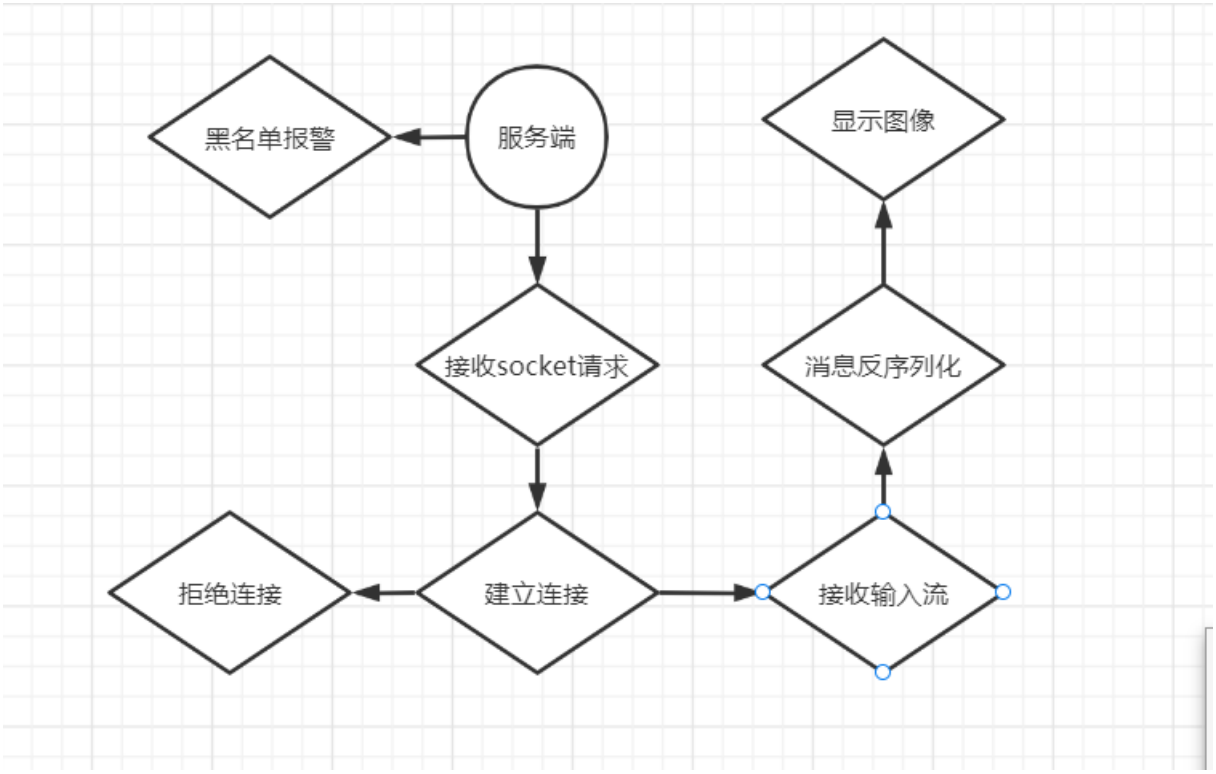
（2）服务端：

服务端中我们主要是接收来自客户端的消息，然后处理消息，最后将其显示在我们的屏幕上。服务器端程序运行后，创建 SSLServerSocket，然后不断的接受连接上的 SSLSocket，每当客户端连接上，就将 SSLSocket 交到一个线程手里，由该线程负责该客户端的所有交互行为，服务器保存客户端 IP 与 SSLSocket 的对应关系

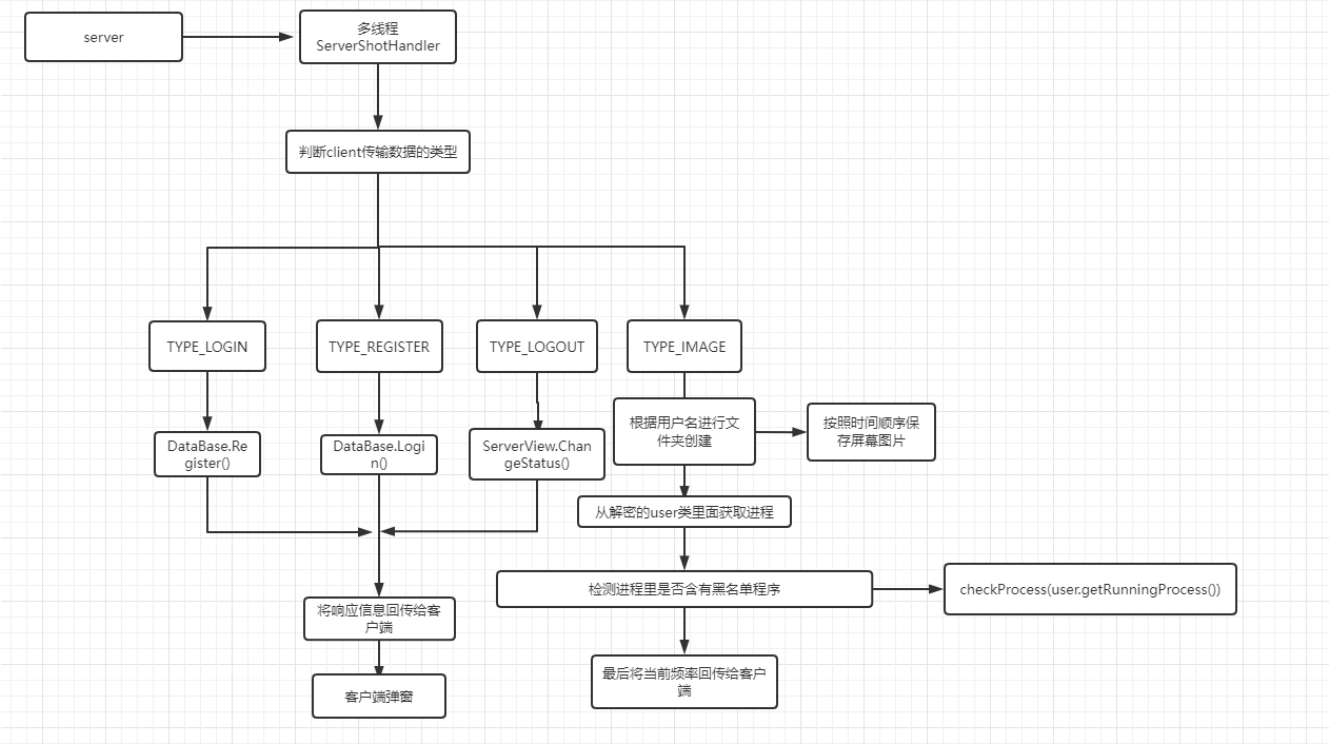
这部分核心部分就是显示我们的图像信息：

```
    //Creating a tree table view
    TreeTableView<String> treeTableView = new TreeTableView<>(root);
    treeTableView.getColumns().add(column);
    treeTableView.setPrefWidth(202);
    treeTableView.setPrefHeight(700);
    treeTableView.setShowRoot(true);
//      sceneRoot.getChildren().add(treeTableView);



    GridPane grid = new GridPane();
    grid.setHgap(10);
    grid.setVgap(10);
    grid.setPadding(new Insets( top: 0,  right: 10,  bottom: 0,  left: 5));
    grid.add(treeTableView,  columnIndex: 0,  rowIndex: 0);
```

基本流程：



服务端核心功能总体设计



dbcon 类：

Database 模块：

这一部分里面写了有关后台数据库的处理，以及数据库中核对我们用户登录的数据是否正确。

后台数据库中包含了我们用户的大量数据，我们也是基于此数据库来完成登录、注册、图像信息保存和传输等。

下面是此部分核心部分，这里我们写了当我们创建用户时在数据库中建立新的表来存放有关此用户的各种信息。

```java
public static void CreateTable(Connection c)/*创建USER表*/
{
    Statement stmt=null;
    try{
        stmt = c.createStatement();
        String sql = "CREATE TABLE USER " +
                "(ID integer PRIMARY KEY autoincrement," +
                " USERNAME           TEXT    NOT NULL, " +
                " PASSWORD           TEXT    NOT NULL, " +
                " IP                 TEXT    NOT NULL, " +
                " MAC                TEXT    NOT NULL)";
        stmt.executeUpdate(sql);
        stmt.close();
    }catch (Exception e){
        System.out.println("Create Table Wrong");
        System.err.println(e);
    }
}
```

还有我们登录时检索我们数据库中的数据，只有当数据库中有我们用户的信息，且输入密码的 MD5 与数据库中存储的 MD5 相匹配时才能够登录成功。

当我们注册输入用户名的时候，下面部分代码也能够检验我们输入的用户名是否重复以及其他属性输入是否有不妥当的地方。

当登录成功时候返回 1，不然返回 0。

```
    Statement stmt=null;
    /*校验Username是否重复*/
      System.out.println("database register");
    System.out.println(Username + Password + Ip + Mac);
    stmt = c.createStatement();
    ResultSet rs = stmt.executeQuery( sql: "SELECT * FROM USER WHERE USERNAME=\""+Username+"\";" );
    int rowCount = 0;
    while(rs.next()) {
        rowCount++;
    }
    if (rowCount>0)
        return 2;

    /*传参*/
    try {
        c.setAutoCommit(false);
        System.out.println("Opened database successfully");

        stmt = c.createStatement();
        String sql = "INSERT INTO USER (USERNAME,PASSWORD,IP,MAC) " +
                "VALUES ("+"\""+Username+"\",\""+Password+"\",\""+Ip+"\",\""+Mac+ "\");";
        System.out.println(sql);
        stmt.executeUpdate(sql);
        c.commit();
        return 1;
    } catch ( Exception e ) {
        System.err.println(e);
        return 0;
    }
```

User 部分：

这部分我们定义了 user 对象所包含的属性，我们将对照我们的定义对每一个 user 对象进行赋值，在后续调用中我们只需要明确我们需要 user 的哪一个变量就可以灵活地调出此信息。

```java
package dbcon;

import ...

public class User implements Serializable {
    public String Username;
    public String Password;
    public String RePassword;
    public String ClientIP;
    public String ClientMac;
    public String ServerIP;
    public String ServerPort;
    public int Frequency;
    public byte[] imageData;
    public String RunningProcess;
```

序列化设计

```java
/**
 * 反序列化
 * @param bytes
 * @return
 * @throws IOException
 * @throws ClassNotFoundException
 */
public static User DeserializeData(byte[] bytes) throws IOException, ClassNotFoundException {
    ByteArrayInputStream bin = new ByteArrayInputStream(bytes);
    ObjectInputStream oin = new ObjectInputStream(bin);
    User user = (User)oin.readObject();
    oin.close();
    bin.close();
    return user;
}
```

```java
/**
 * 序列化
 * @param user
 * @return
 * @throws IOException
 */
public static byte[] SerializeData(User user) throws IOException {
    ByteArrayOutputStream bos = new ByteArrayOutputStream();
    ObjectOutputStream out = new ObjectOutputStream(bos);
    out.writeObject(user);
    byte[] bytes = bos.toByteArray();
    out.flush();
    return bytes;
}
```

传输 aes 加密设计

```java
/**
 * AES 对称加密（RSA非对称加密）
 * CBC 有向量 （ECB 无向量）
 * PKCS5Padding 填充模式（NoPadding 无填充）
 */
private static final String ALG_AES_CBC_PKCS5 = "AES/CBC/PKCS5Padding";
private static final String ALGORITHM = "AES";
private static final Charset UTF8 = StandardCharsets.UTF_8;
private String aesKey = "12e476beac1a4g20";  // 指定好的秘钥，非Base64和16进制
private String aesIv = "2e119e58a526bc64";   // 偏移量
private SecretKeySpec skeySpec;
private IvParameterSpec iv;
```

自定义协议模块：

这一模块定义了我们消息传输时候的格式以及传输的是什么内容，以及状态的一些标志。例如 TYPE_LOGIN 指明本次消息发送是传送的 login 需要的信息，TYPE_IMAGE 指明本次消息发送是传送监控图像

```java
package protocol;

import ...


public class Protocol {
    public static int TYPE_REGISTER=1; // 表示客户端注册
    public static int TYPE_LOGIN=2;    // 表示客户端登录
    public static int TYPE_LOGOUT=3;   // 表示登出
    public static int TYPE_IMAGE=4;    // 表示发送的是屏幕图片数据
```

程序函数清单：

client 类：

1. Client()：实例化函数，在其中加载用于 ssl 传输的证书及相关初始化工作
2. getMD5(byte[] source)：返回 md5 加密结果
3. isCorrectIp(String ipString)：判断参数是否是合法 IP 地址
4. isCorrectPort(String Port)：判断参数是否是合法端口号
5. getRunningProcess()：获取当前客户端正在运行的所有进程名称
6. connect(User user)：SSLSocket 建立连接，绑定输入输出流
7. login(User user, DataOutputStream dos)：发送登录消息
8. register(User user, DataOutputStream dos)：发送注册消息
9. logout(User user, DataOutputStream dos, DataInputStream dis)：发送登出消息，关闭 socket 连接和输入输出流
10. getAccuracy(long size)：获取消息精度用于压缩消息
11. getLocalMac(InetAddress inetAddress)：获取客户端 MAC 地址

12. getScreenShot()：截取屏幕图像

13. sendUser(int type, User user)：传输数据，Type 决定我们传输的是什么类型的数据，user 是一个 User 类的对象，包含 User 的所有属性。

14. saveImage(BufferedImage buff,long desFileSize)：保存图像数据，Buff 是用来装图像数据的，desFikeSize 是数据的长度。

15. getMsg()：获取 server 端返回消息

16. sendImage()：发送图片

17. run()：与前端双线程并行执行时被调用

18. stop()：线程停止时被调用

19. main()：函数入口用来启动客户端

ClientVuew 类：

1. start(Stage Client)函数：创建一个 client 对象，里面还有我们登录界面以及其他图形化界面的各种 ui 设计。

2. ClientView()：ui 入口，其中 launch 启动 ui

3. clientAlert()：在 client 类中调用这里实现黑名单程序弹窗

4. handle(WindowEvent arg0)函数：

5. enableTray(final Stage stage)函数：显示、最小化、退出等的图标设置

6. actionPerformed(java.awt.event.ActionEvent e)函数：显示、最小化、退出等按钮的点击事件设置

7. run()函数：运行函数

8. mouseReleased(MouseEvent e)函数：按下鼠标并释放的对应事件

9. mousePressed(MouseEvent e)函数：按下鼠标对应的事件

10. mouseExited(MouseEvent e)函数：鼠标移动到子组件时发生的事件

11. mouseEntered(MouseEvent e)函数：鼠标经过的事件

12. mouseClicked(MouseEvent e)函数：鼠标点击响应的事件

13. init()函数：初始化

14. stop()函数：停止运行

ServerView 类：

1. start(Stage Server)函数：开始并创建一个 sever 对象

2. run()：打开 ui

3. setImg()：显示图像

4. ChangeStatus()：修改用户状态

5. start()：ui 实现

6. getName()：获取点击的用户名

7. setName(String fName)：设置用户树对应的用户名

8. getStatus()：获取用户状态

9. setStatus(String fName)：更改用户状态

10. enableTray(final Stage stage)：实现最小化托盘的右键菜单

11. init()：初始化

12. stop()：停止 UI

SeverShotHandler 类：

1. ServerShotHandler(Socket socket)：一个接口，在 Server 类中循环多线程监听，没监听到请求消息就调用该接口，获取请求来源的输入流

2. run()函数：运行函数，多线程监听中运行，用于处理来自客户端的请求消息

database 类：

1. Register(String Username, String Password, String Ip, String Mac)函数：注册一个用户以及检索用户是否重复。Username 就是用户名，Password 就是密码，Ip 就是用户 ip 地址，Mac 就是用户 mac 码。

2. DatabaseInit()：数据库初始化以及初始化失败的报错

3. IsTableExist(Connection c)：检索我们的数据库中是否有相应对象的数据，c 就是与我们连接的对象客户

4. CreateTable(Connection c)：为我们的客户对象在数据库中创建相应的表单

5. Login(String Username,String Password)函数：用户登录，两个参数为用户名和密码。

User 类：

1. setStatus(String status)：标志用户状态

2. getStatus()：获取用户状态

3. setClientMonitor(String clientMonitor)：设置一个管程

4. getClientMonitor()：获取一个管程

5. getClientIP()：获取客户 ip 地址

6. setClientIP(String clientIP)：设置客户 ip 地址

7. getServerIP()：获取服务端 ip 地址

8. getClientMac()：获取客户 mac 码

9. getPassword()：获取密码

10. getServerPort()：获取服务端端口

11. getUsername()：获取用户名

12. setClientMac(String clientMac)：设置客户端 mac 码

13. setPassword(String password)：设置密码

14. CheckRePassword(String rePassword)：检查密码，参数为待检查密码

15. setServerIP(String severIP)：设置服务端 ip 地址

16. setServerPort(String severPort)：设置服务端端口

17. setUsername(String username)：设置用户名

Protocol 类：

1. send(int type, DataOutputStream dos,byte[] data)：规定消息传送的格式，type 标志传送消息类型，dos 为消息传输流，data 装我们要传输的消息

2. getResult(DataInputStream dis)：获取结果，dis 为输入流

3. DeserializeData(byte[] bytes)：反序列化，用来获取传输的对象信息

4. SerializeData(User user)：序列化，用来序列化需要传输的对象信息

Result 类：

1. Result(int type, int totalLen, byte[] data)：规定消息结果的类型，type 就是消息的类型，data 装我们的消息
2. getType()：获取消息类型标志
3. getData()：获取数据

AesEnc 类：

1. decrypt(byte[] cipherStr)：密文解密，base64 解码得到明文
2. encrypt(byte[] m)：明文加密，得到 base64 编码的密文

3 系统设计难点与亮点

3.1 屏幕图像在服务端显示

本次程序的主要功能就是将客户端的屏幕图像传送给服务端并且在服务端显示出来，那么我们选择这样处理这个问题。首先我们在 client 类里面设置了 senduser 函数，有两个参数，type 和 user，type 是告诉 senduser 函数我们将要传送什么样的消息，而 user 则是储存我们将要传输的消息。服务端根据 type 得知接收的是什么消息，如果是图像信息，则进行消息处理以及反序列化，然后将处理好的图像信息显示在我们的屏幕上。

要将服务端后端收到的图片显示到 UI 界面中也是一个难点，我们在 ServerView 中建立一个方法 setImage（），设置一个 public Image img 用来存放图像，由于我们 UI 是用 JavaFX 实现的，用到了 JavaFX 的 ImageView 来显示 Image 中记录的图像信息，ImageView 有个性质，若其中显示的 Image 发生改变，它可以自动更新，我们利用了这一点来实现动态的图片显示。

3.2 服务端 UI 和客户端 UI 关闭，服务仍在继续

在关闭 UI 是未能停止服务，为了保证 UI 关闭服务也要停止，程序完全停止，我们在 Server 和 Client 中设置了 stop（）方法，来关闭服务，在 ServerView 和 ClientView 中调用相应的 stop（）函数，来停止服务。

1. 服务端 UI 的启动和服务端 socket 通信的进行

在进行服务端的前后端对接时发现，若先启动 UI，服务端 socket 通信无法进行，不能开启监听，若先开启监听，UI 则无法启动，我们最后给 UI 界面单独开设了一个线程，这样不影响后续代码的执行。

3.3 用户树的点击事件和动态显示

用户树我们采用的是 JavaFX 的 TreeTableView，在一些教程中，没能发现有实现 TreeTableView 的点击事件的，在查阅各种资料后，结合 TreeView 和 TableView 的特征，最终用户树的点击事件代码相关实现如下：

```
treeTableView.addEventFilter(MouseEvent.MOUSE_CLICKED, new
EventHandler<MouseEvent>()
{
    public void handle(MouseEvent event)
    {
        Node node = event.getPickResult().getIntersectedNode();
        if (node instanceof Text || (node instanceof TreeTableCell &&
((TreeTableCell) node).getText() != null)) {
            String
name=treeTableView.getSelectionModel().getSelectedItem().getValue().getName();
            v = dbcon.DataBase.UserList(DatabaseInit());
            int i=0;
            while(i<v.size()){
                if(v.get(i).getUsername().equals(name)){
                    System.out.println(v.get(i).getClientIP());
                    System.out.println(v.get(i).getClientMac());
                    Username=v.get(i).getUsername();
                    text1.setText("  用户名："+v.get(i).getUsername()+"  IP:
"+v.get(i).getClientIP()+"  Mac："+v.get(i).getClientMac());
                    break;
                }
                i++;
            }
            System.out.println(name);
        }
    }
});
```

有了用户树的点击功能，我们就能通过点击用户树中的用户来显示对应图片和用户信息。

TreeTableView 不具备自动更新节点的性质，为了能动态修改用户树的用户状态，在每次修改前清空节点，重新写入，这样实现了用户状态的动态显示。

3.4 黑名单检测告警

黑名单检测的难点不在检测本身，在于如何前端弹窗显示，这和

我们前端使用的是 javafx 有关。因为 javafx 的原理是在其主线程中执行 ui 并根据事件响应操作，而黑名单告警消息的得到不在 javafx 类的处理中，javafx 前端程序和后端消息响应属于双线程并发执行的，所以需要从外部类调用 javafx 中的弹窗方法，如果直接调用是会出现异常的。最后通过查找资料找到了一种方法是利用 Platform.runlater，重写其中的 run 方法，从而实现弹窗方法在外部类调用。弹窗的实现我们做到的效果是持续出现，直到用户运行的进程中没有黑名单程序。

调用出现弹窗的方法相关代码如下

```
public static void clientAlert() {
        Platform.runLater(new Runnable() {
            @Override
            public void run() {
                alert.setTitle("异常报警");
                alert.setHeaderText("发现下列黑名单程序，请立刻关闭");
                alert.setContentText(IllegalProcess);
                if(!alert.isShowing())
                    alert.showAndWait();
            }
        });
    }
```

3.5 C/S 双向认证，加密传输

关于加密传输很容易，将序列化好的数据进行 AES 加密就可以了，双向认证主要在于客户端，服务端需要协商一个两者共同认可并且验证的协议或者加解密方式，最初的想法是客户端服务端都有各自自定义的私钥，同时两者共有一个公钥，然后协定一个类似于 RSA 双方都能加解密的方法。但是实现过程中觉得客户端的私钥并不易保证安全，而且实现过程的封装性和安全性都没有保证。最终采用了基于 TLS 协议的方式进行双向认证。

实现技术：是 Sun 为了解决在 Internet 上的安全通讯而推出的解决方案。它实现了 SSL 和 TLS（传输层安全）协议。在 JSSE 中包含了数据加密，服务器验证，消息完整性和客户端验证等技术。通过使用 JSSE，开发人员可以在客户机和服务器之间通过 TCP/IP 协议安全地传输数据。

Server 需要：

（1）KeyStore: 其中保存服务端的私钥

（2）sKeyStore:其中保存客户端的授权证书

Client 需要：

1）KeyStore：其中保存客户端的私钥

2）Trust KeyStore：其中保存服务端的授权证书

在这里我使用 Java 自带的 keytool 命令，去生成这样信息文件。当然目前非常流行的开源的生成 SSL 证书的还有 OpenSSL。OpenSSL 用 C 语言编写，跨系统。但是我们可能在以后的过程中用 java 程序生成证书的方便性考虑，还是用 JDK 自带的 keytool。

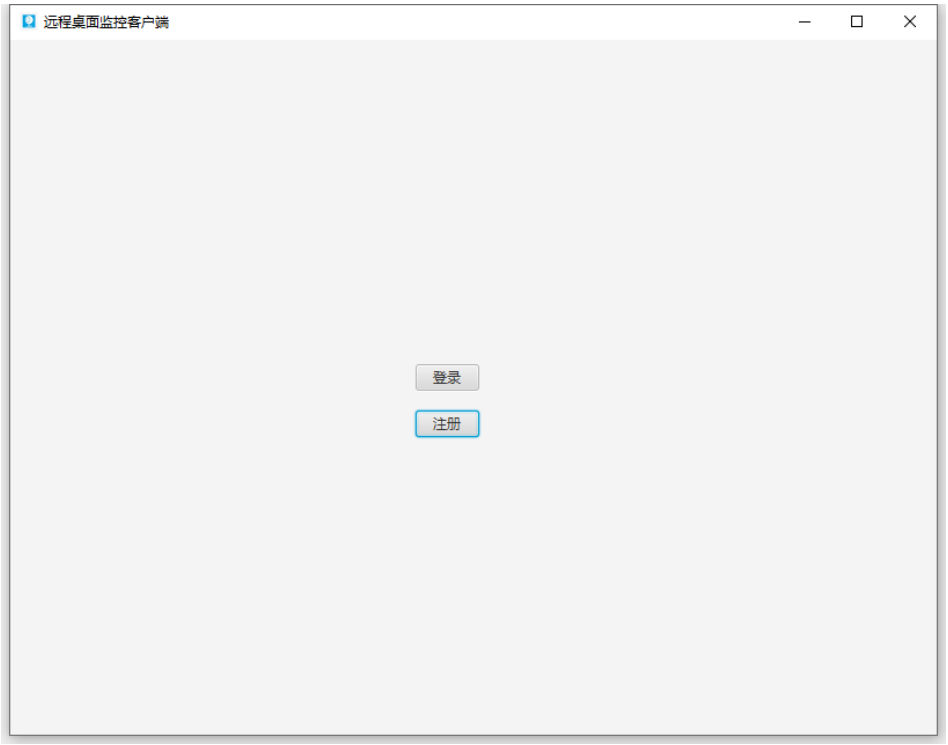
3.7 生成 jar 包转 exe 直接执行

这一步实现了把代码变成一个点击即可运行的 exe 文件，从而真正意义上完成了程序设计。这一步其实也并不顺畅，因为我们使用相对路径进行相关文件例如证书、图标等的读取，在生成 jar 包试运行的时候发现一些路径存在异常，而用 IDEA 执行却是正常的。最后发现原因出在 IDEA 运行代码时，相对路径的根目录是项目根目录，而生成 jar 包运行或者运行 exe 时其相对路径的根目录就是其文件的所属目录，所以在代码上需要调整所有路径相

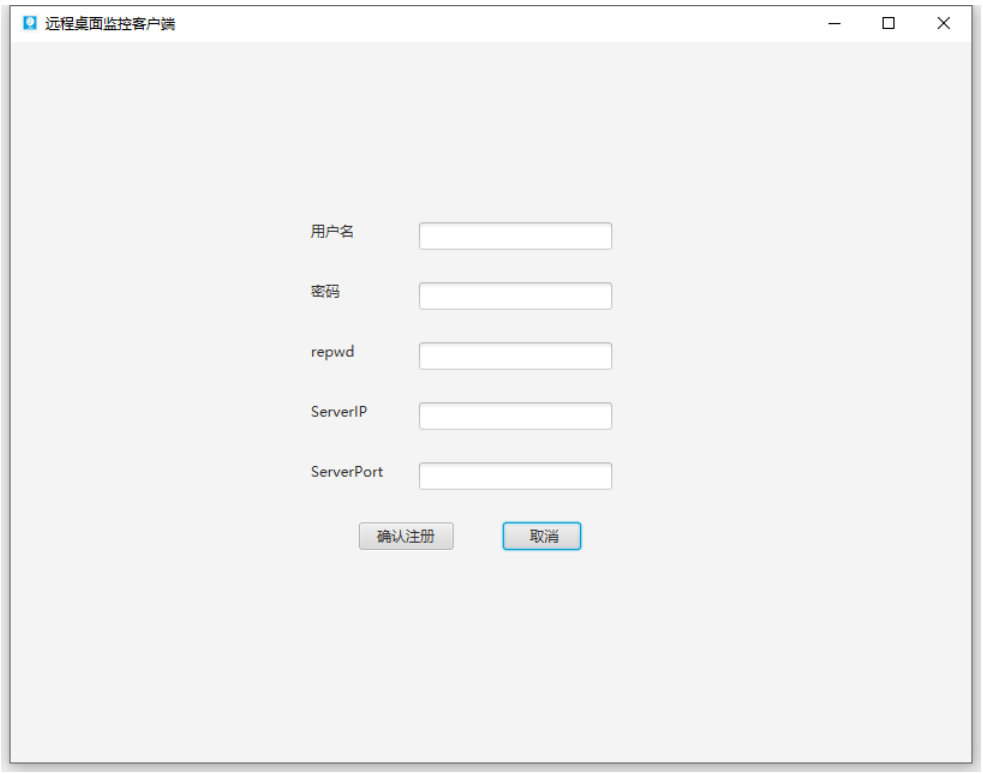
关的内容。最后也是通过 java 生成 jar 包，jar 包用 exe4j 生成 exe 从而如愿以偿得到了可以直接执行的 exe 文件。

4 设计成果

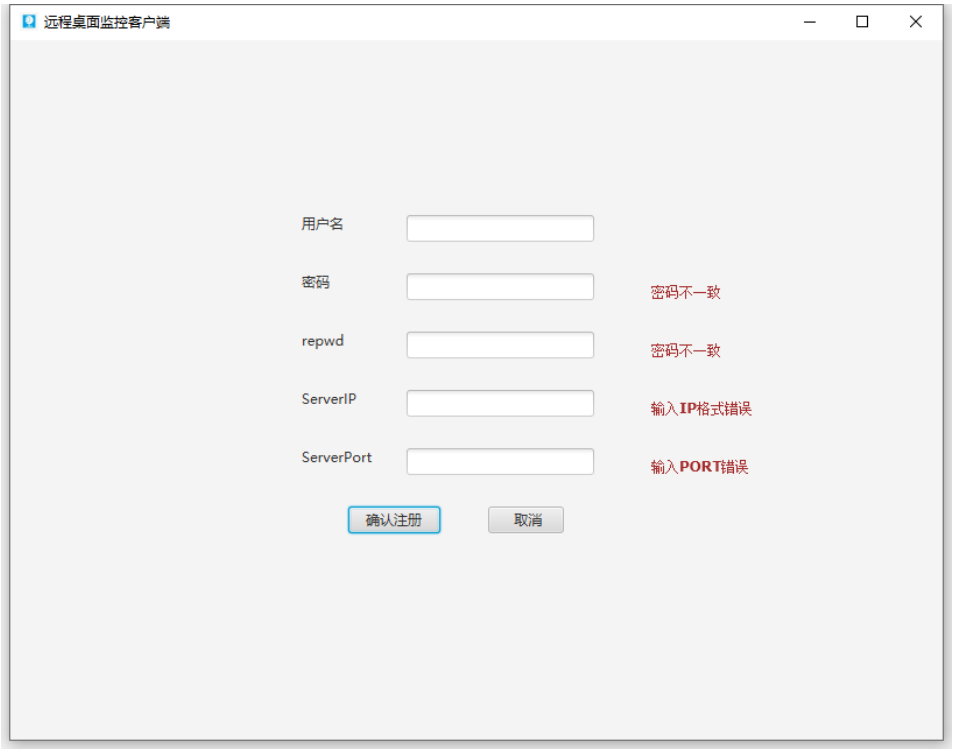
我们打开我们的程序，进入登录界面：



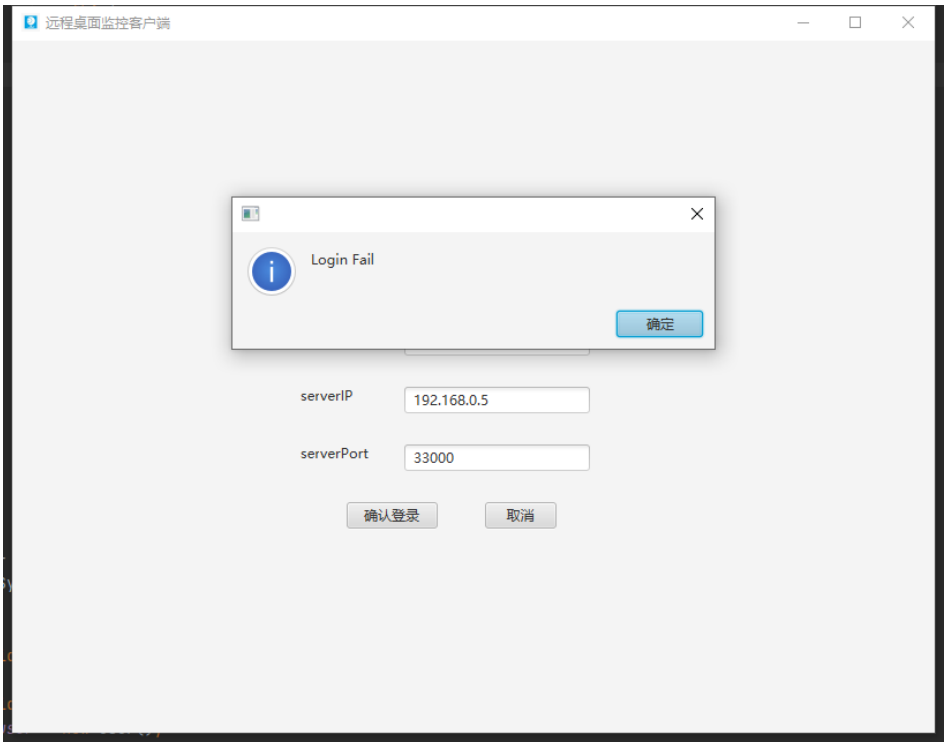
然后假如我们没有账户的话就需要注册一个账户：



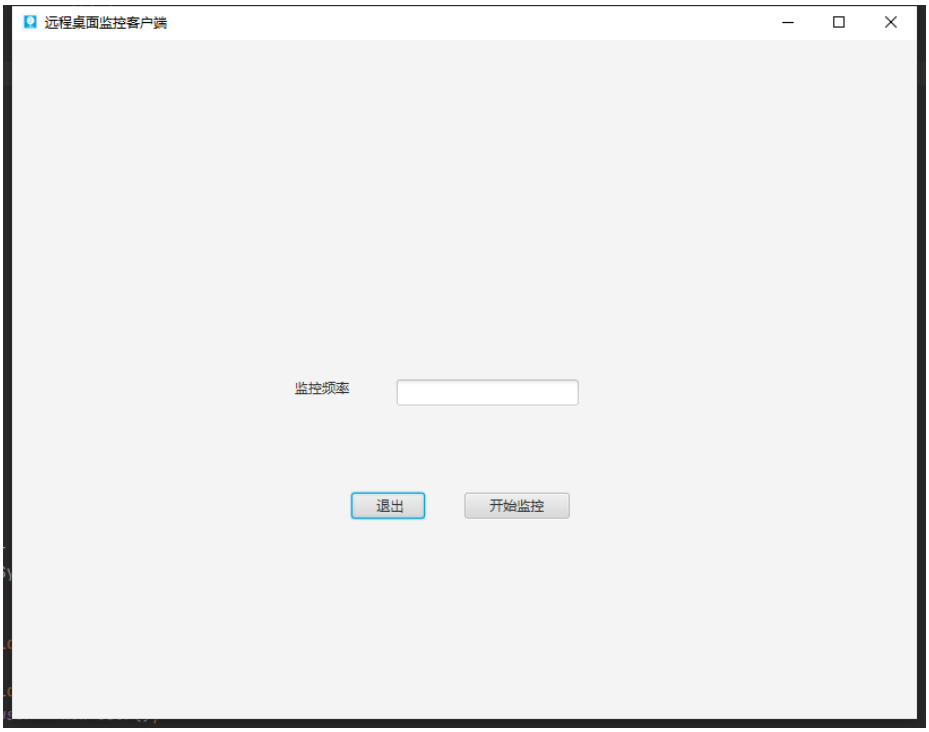
假如我们两次密码不一致或者 ip 格式有问题则会报错：



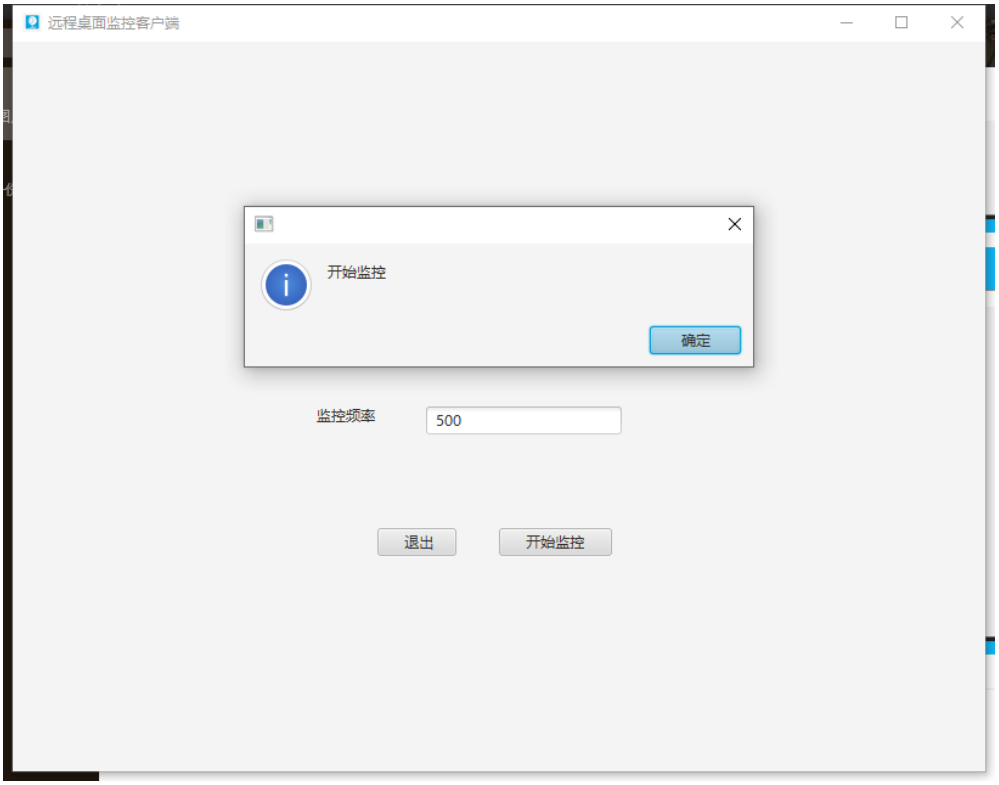
登录输入密码错误：



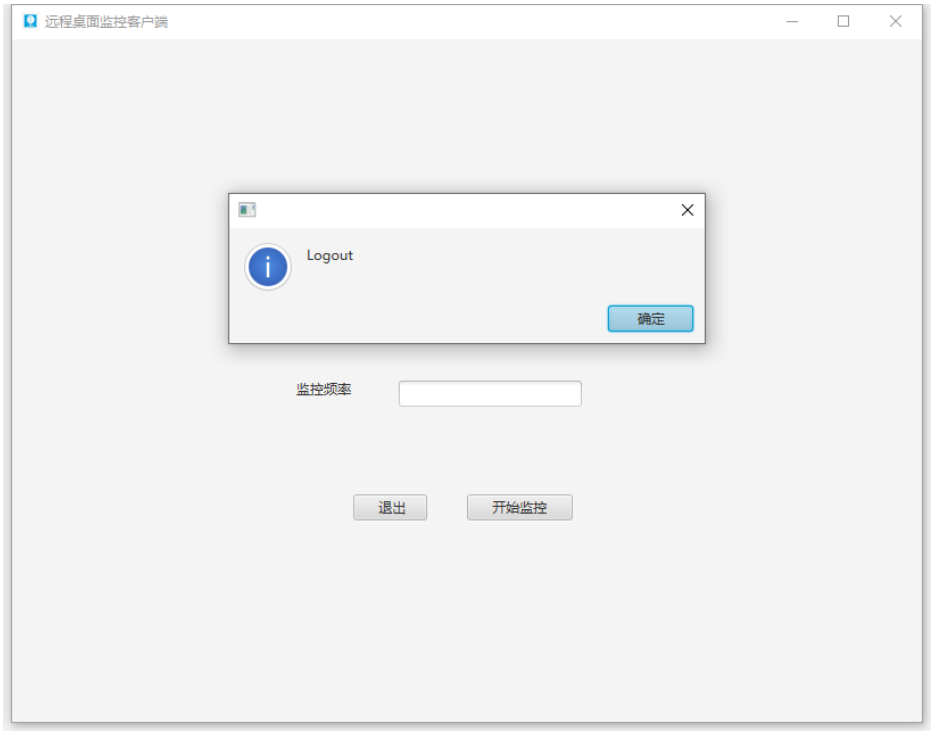
登录成功，我们可以输入我们监控的频率：



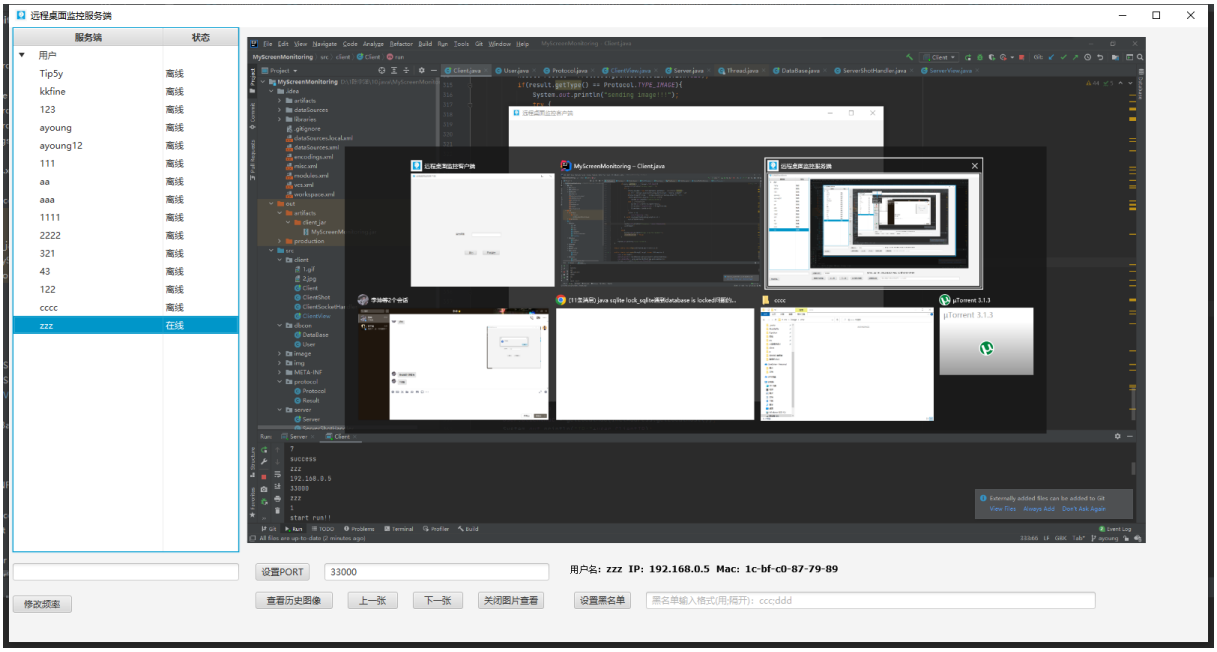
开始监控：


远程桌面监控客户端
开始监控
确定
监控频率
500
退出
开始监控

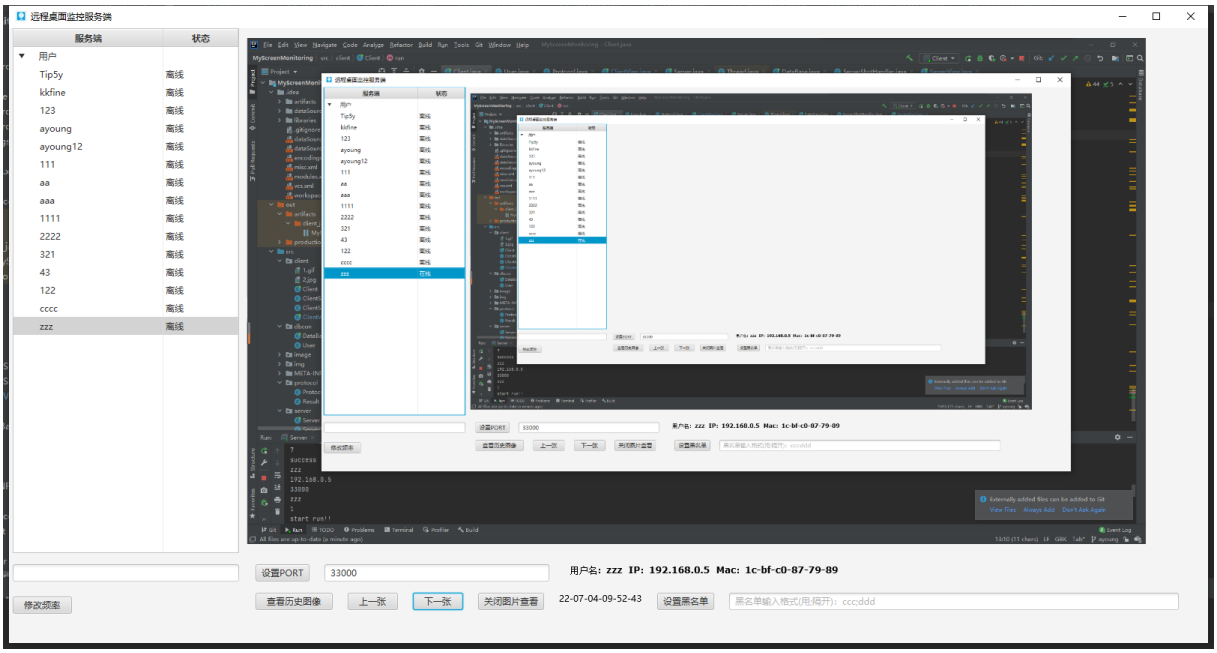
退出：


远程桌面监控客户端
Logout
确定
监控频率
退出
开始监控

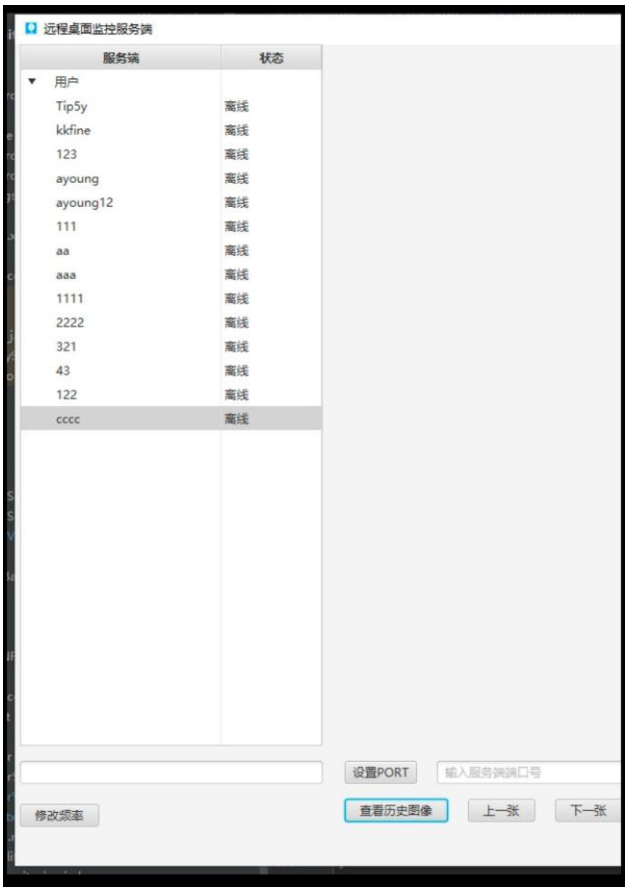
监控画面：



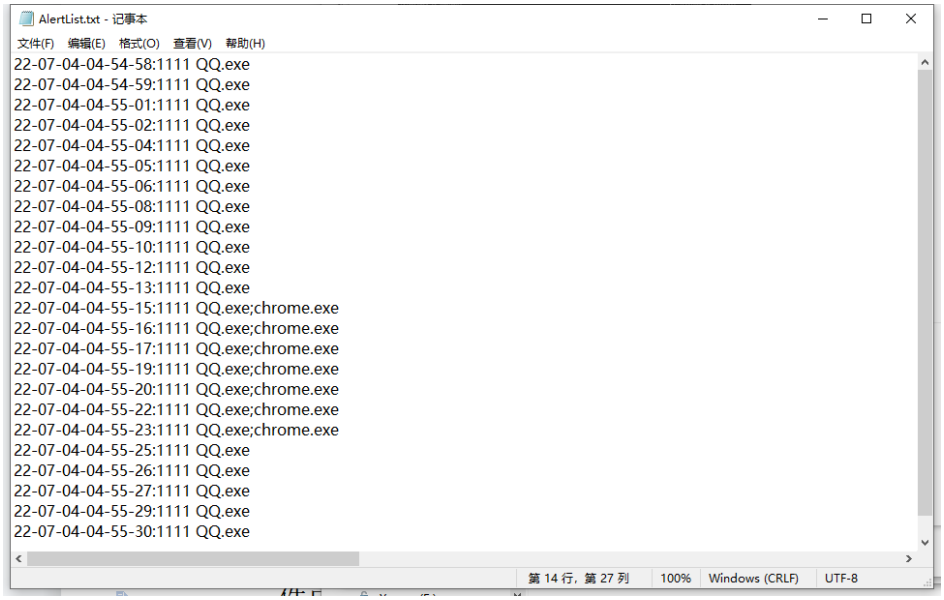
还可以看到监控历史画像：

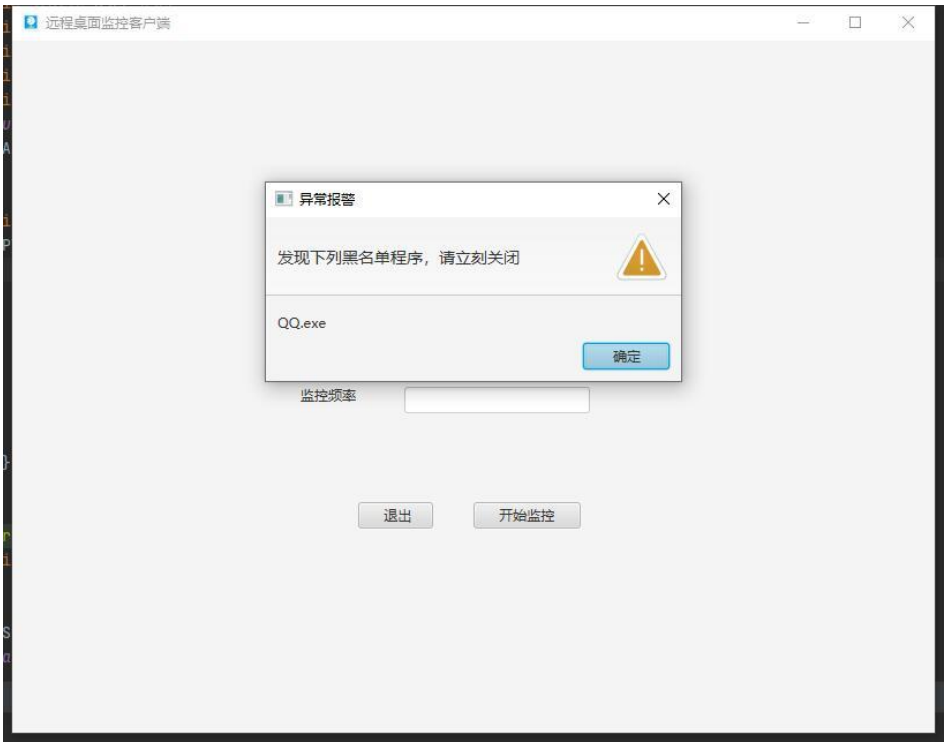


我们在界面左侧可以看到用户树，看哪个用户在线哪个用户退出：



另外我们增加的高级功能，黑名单程序报警，假如客户端打开黑名单中的程序，那么我们监控就在文档中记录打开黑名单程序的记录，并让正在运行黑名单程序的客户端持续弹窗：





最后我们生成了 exe 文件，可以直接运行

jre	2022/7/4 19:35	文件夹	
error.log	2022/7/4 20:17	文本文档	1 KB
RDMS-Client.exe	2022/7/4 20:00	应用程序	32,194 KB
RDMS-Server.exe	2022/7/4 20:02	应用程序	32,207 KB
userdata.db	2022/7/4 20:17	Data Base File	3 KB

5 设计心得

这次大型程序设计，我们团队选择使用 JAVA 编程语言。设计远程屏幕控制系统，以下是团队心得体会：

本次程序设计中，团队首先是确定了程序大致框架，我们选择使用 C/S 模型来交互数据。这一次的项目又让我们对 C/S 模型有了更深入的了解以及更加了解 socket 原理。

在系统图形化方面我们也是增加许多选项，我们一致认为图形化界面

也是一个十分重要的部分，这关系到用户的体验。也利于我们管理用户输入的信息种类。

在客户端屏幕截取功能的基础上我们又增加截取频率的变更，这是根据实际使用过程而做出的相应举措，能够让系统更加灵活的监控客户端。比如在考试监控系统中就能避免考试利用频率间隔时间作弊的问题。

对于收集好的图像信息，我们要将其传输到服务端，所以我们就要思考如何将我们的信息安全地送到服务端，于是我们对消息进行加密以及让 C/S 之间双向认证才能够正常交互信息。

对于后端数据库我们也是支持大量用户注册的，让这个系统不再是对单人使用的，我们可以进行登录，数据库大量管理我们用户的信息方便操作。

在团队工作的过程中并不是一帆风顺，大家都出现了问题，我们再最开始会有每个人代码部分习惯不一样的特点，以至于整合需要我们磨合，还有注释部分的不一样也让文档编写人员有点吃力。不过团队很快就磨合过来了，成员都各自有序地进行着自己那部分程序的工作，努力与他人交流磨合。大大提升团队人员磨合程度以及与他人协作的能力。

而困难问题的出现也磨练了我们处理问题的能力，这一次程序设计，要学习的是很新的知识，难度很大，但是经过团队成员的共同努力，我们克服了各种难题，这让团队每个成员都对 java 程序设计又有了新的认识，以及都得到了更深入的 C/S 交互的各种知识。思考问题的思维，与人探讨问题的技术都有一定程度提升。

亮点：

高级功能和基本功能全部实现

实时设置黑名单并且检测弹窗，实时修改频率

用户树实时显示用户状态和用户名

基于 SSLsocket 和 TLS 协议的双向认证

问题：

小组的分工合作处理，整合难题，前后端对接问题等等

技术上的问题：

后端频率修改实时传送给前端进行更改，用户树状态实时显示等等