

# 核酸检测登记系统

朱浩泽 1911530 何坤彬 1911417

April 25, 2022

## 1 软件需求和设计

### 1.1 需求背景

2020 年伊始，一场突如其来的新型冠状病毒肺炎疫情，在春节期间突袭神州大地，面对困难，举国上下团结一心，共同投入到抗击疫情的工作中。在党中央坚强领导下，中国人民风雨同舟、众志成城，发扬一方有难、八方支援精神，构筑起疫情防控的坚固防线。广大医务人员白衣为甲、逆行出征，为保障人民的健康，进行了一轮又一轮的健康监测。而这些检测结果的数据量是庞大的，这就需要对海量的信息进行有效的组织管理，来确保信息的准确性和完整性。于是，我们计划开发一款核酸检测登记系统。

### 1.2 系统设计思路

在拥有需求之后，我们可以对我们的系统进行设计。首先，核酸登记系统需要识别被登记人，而如果用人名作为登记人员的信息，可能会有很多重名的人员导致信息混乱。所以我们需要选取每个人都具有的且为独一无二的表示方式。在现如今的社会，每个人都拥有着自己的身份证号，且是一人一号，具有唯一性。由此可见身份证号可以很好的承载着我们的需求，故选取身份证号作为登记人员的身份标识，配合被测人员的姓名进行辅助。

接下来便是，当我们获得了人员的信息后，需要存储其核酸检测状态，所以需要存储人员的检测情况（阴性/阳性）。在检测后，我们也要给被测人员一个检测情况的反应，并在必要时（万一呈现阳性时）可以找到被测人员，所以我们要记录被测人员的手机号以便进行联系和反馈，并登记被检测人员的住址进行潜在的流行病学调查。由于疫情发展瞬息万变，一个人可能会测多次核酸，所以我们还要记录其检测时间，以区分不同批次检测结果。而且，不同人群的身体反应可能也是不一样的，所以我们还要记录人员的年龄信息。所以这里我们选择了一个简单的数据库，其中包含了人员的身份证号、姓名、检测状态（阴性/阳性）、检测时间、检测结果（阴性/阳性）。

当然，为了加快信息录入的速度，我们允许多人对信息进行登录。同时又避免信息的混乱和保证信息的安全性，我们采取了注册制度。用户需要先进行注册，然后医护人员才可以为用户的核酸检测数据进行上传和更改，由于核酸检测只有最近的结果是有说服性的，所以我们每次登记时只相当于更改被登记人的最后一次见的时间和检测记录。

我们采用 java 框架配合 mysql 数据库进行系统设计。

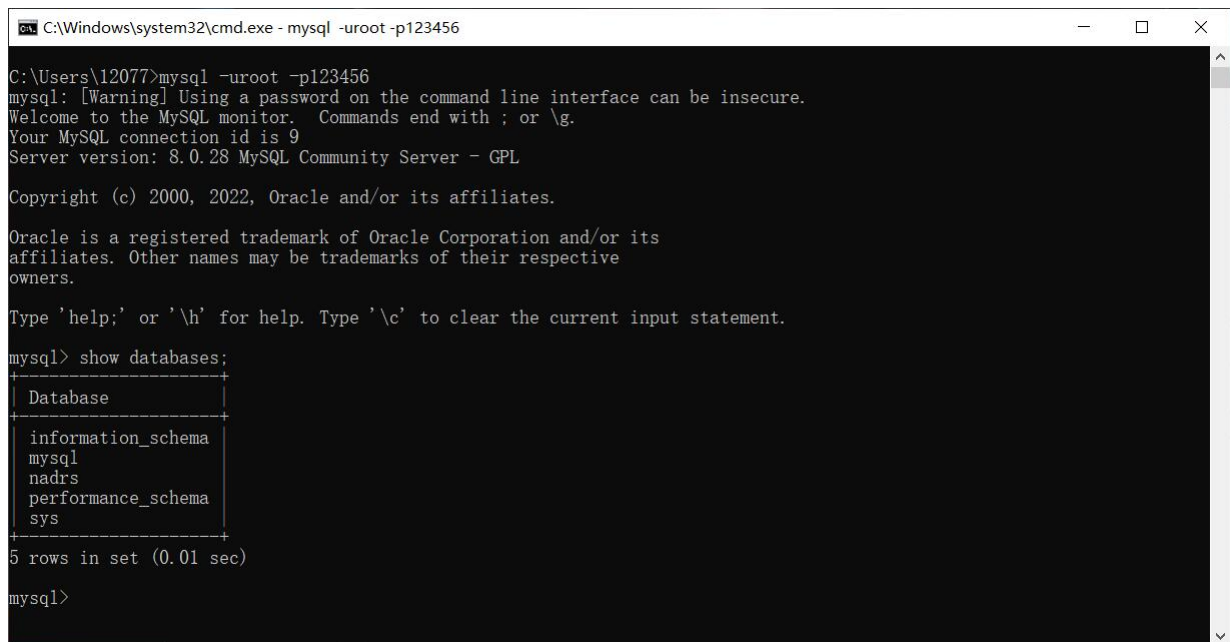
## 2 系统实现方法

### 2.1 编程环境

该核酸检测登记系统在 Windows 64 位的机器上使用 java 语言以及 MySQL 数据库实现。

### 2.2 连接 MySQL 数据库

该系统需要连接数据库 nadrs，使用 root 权限，密码为 123456：



```
C:\Windows\system32\cmd.exe - mysql -uroot -p123456
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.28 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql       |
| nadrs       |
| performance_schema |
| sys         |
+-----+
5 rows in set (0.01 sec)

mysql>
```

```
1 private static String driver = "com.mysql.jdbc.Driver";// 驱动程序名
2 private static String URL = "jdbc:mysql://localhost:3306/nadrs?useSSL=false&allowPublicKeyRetrieval=
   true";// URL指向访问的数据库
3 private static Connection con = null;
4 private static Statement smt = null;
5 private static ResultSet rs = null;
6
7 private static Connection createConnection() {
8     try {
9         Class.forName(driver);// 加载驱动程序
```

```

10         return DriverManager.getConnection(URL, "root", "123456");// 以root权限连接数据库
11     } catch (SQLException e) {
12         System.out.println(e.getMessage());
13         e.printStackTrace();
14     } catch (java.lang.ClassNotFoundException e) {
15         e.printStackTrace();
16         System.out.println("Can't load Driver");
17     }
18     return null;
19 }

```

java 提供连接数据库的包，通过加载“com.mysql.jdbc.Driver”驱动程序实现对数据库的连接。同时能够实现对数据库 SQL 语句的执行，我们将其封装为 runUpdate、runBatch、runQuery 函数以供操作数据库时调用：

```

1  public static int runUpdate(String sql) throws SQLException {
2      int count = 0;
3      if (con == null) {
4          con = createConnection();
5      }
6      if (smt == null) {
7          smt = con.createStatement();
8      }
9
10     count = smt.executeUpdate(sql);
11
12     if (smt != null) {
13         smt.close();
14         smt = null;
15     }
16     if (con != null) {
17         con.close();
18         con = null;
19     }
20     return count;
21 }
22
23 public static void runBatch(ArrayList<String> s) throws SQLException {
24     if (con == null) {
25         con = createConnection();
26     }
27     if (smt == null) {
28         smt = con.createStatement();
29     }
30     for (int i = 0; i < s.size(); i++) {
31         System.out.println(s.get(i));
32         smt.addBatch(s.get(i));
33     }

```

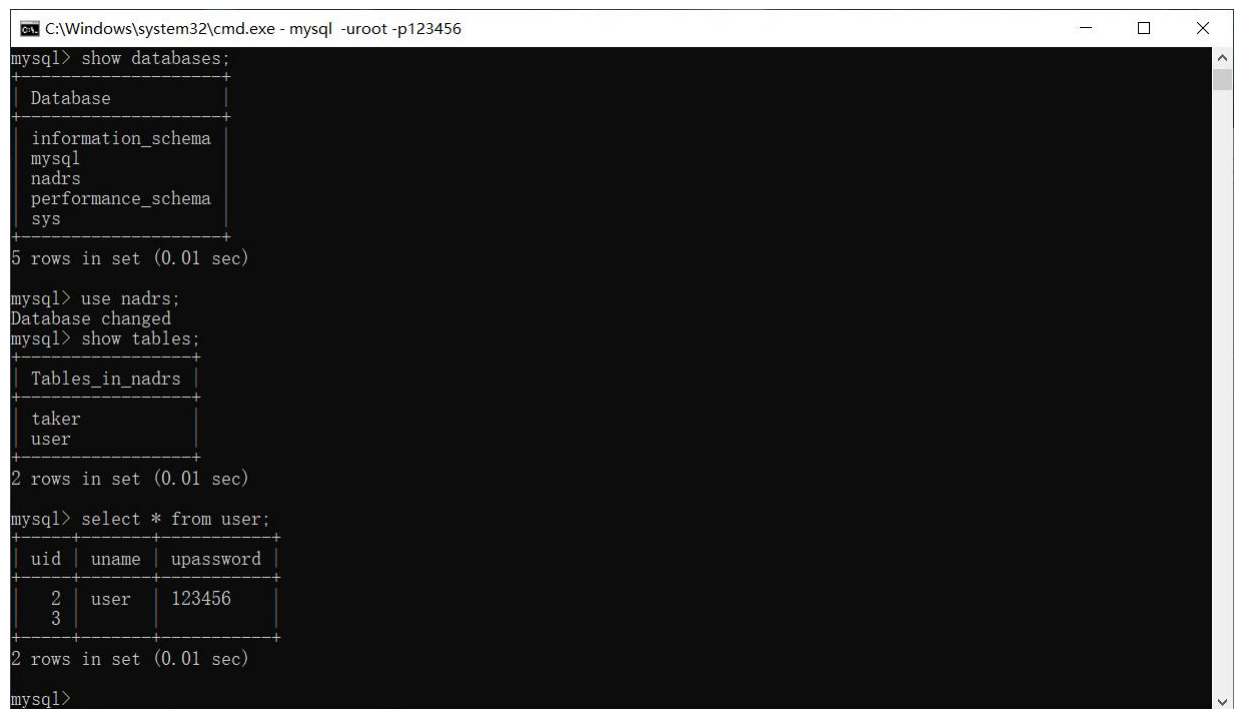
```

34     smt.executeBatch();
35     return;
36 }
37
38 public static ResultSet runQuery(String sql) throws SQLException {
39     if (con == null) {
40         con = createConnection();
41     }
42     if (smt == null) {
43         smt = con.createStatement();
44     }
45     return smt.executeQuery(sql);
46 }

```

## 2.3 登录界面的实现

登录界面需要使用到数据库中的 user 表，user 表在数据库中结构为：



```

C:\Windows\system32\cmd.exe - mysql -uroot -p123456
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql        |
| nadrs        |
| performance_schema |
| sys          |
+-----+
5 rows in set (0.01 sec)

mysql> use nadrs;
Database changed
mysql> show tables;
+-----+
| Tables_in_nadrs |
+-----+
| taker            |
| user             |
+-----+
2 rows in set (0.01 sec)

mysql> select * from user;
+----+-----+-----+
| uid | uname | upassword |
+----+-----+-----+
| 2   | user  | 123456   |
| 3   |       |          |
+----+-----+-----+
2 rows in set (0.01 sec)

mysql>

```

用户在文本框中输入用户名和密码，点击登录按钮后，程序会读取文本框中的内容，并且去数据库中查询 user 表中是否有对应的用户以及其对应的密码是否正确，若密码错误则登录失败。当然也可以点击注册按钮，程序会按照你输入的用户名和密码在 user 表中新增一项记录，这样就可以使用该用户名和密码登录了，其登录和注册对应的 java 代码如下：

```

1  @Override
2  public boolean certifyUser(String uname, String upassword) {
3      String select = "select * from user where uname='" + uname + "' and upassword='" + upassword + "'";
4      ;

```

```

4      boolean isCertifyUser = false;
5      try {
6          ResultSet rs = DBUtil.runQuery(select);
7          if (rs != null) {
8              isCertifyUser = rs.next();
9              DBUtil.realeaseAll();
10         }
11     } catch (SQLException ex) {
12         Logger.getLogger(UserDaoImp.class.getName()).log(Level.SEVERE, null, ex);
13     }
14     return isCertifyUser;
15 }
16
17 @Override
18 public boolean addUser(User user) {
19     String insert = "insert into user(uname,upassword) values('" + user.getUsername() + "','" + user.
20         getUpassword()
21         + "')";
22     try {
23         DBUtil.runUpdate(insert);
24         return true;
25     } catch (SQLException ex) {
26         Logger.getLogger(UserDaoImp.class.getName()).log(Level.SEVERE, null, ex);
27     }
28     return false;
29 }

```

## 2.4 主界面实现

主界面需要使用到数据库中的 taker 表，其结构如图所示：

```
C:\Windows\system32\cmd.exe - mysql -uroot -p123456

mysql> use nadrs;
Database changed
mysql> show tables;
+-----+
| Tables_in_nadrs |
+-----+
| taker            |
| user             |
+-----+
2 rows in set (0.01 sec)

mysql> select * from user;
+----+-----+-----+
| uid | uname | upassword |
+----+-----+-----+
| 2   | user  | 123456    |
| 3   |       |           |
+----+-----+-----+
2 rows in set (0.01 sec)

mysql> select * from taker;
+----+-----+-----+-----+-----+-----+-----+
| sid | sname | snumber | sage | sphone | saddress | state | stime |
+----+-----+-----+-----+-----+-----+-----+
| 3   | 小熊维尼 | 11112357 | 5    | 134679852 | 天津 | 阴性 | 2022.4.19 |
| 4   | 小猪佩奇 | 22223333 | 1    | 852314679 | 乌鲁木齐 | 阳性 | 2022.4.18 |
| 7   | 张三   | 55556667 | 15   | 1568113135 | 北京 | 阳性 | 2022.4.20 |
| 8   | 李四   | 987654321 | 30   | 123456789 | 重庆 | 阴性 | 2022.4.23 |
+----+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

程序通过 `getAllTaker` 函数获取数据库中 `taker` 表所有的内容，并通过 java 中 `JScrollPane` 将所有人员的信息展示在主界面上：

```
1  @Override
2  public List<NadTaker> getAllTaker() {
3      String select = "select * from taker";
4      try {
5          List<NadTaker> takers = new ArrayList<NadTaker>();
6          ResultSet rs = DBUtil.runQuery(select);
7          while (rs.next()) {
8              NadTaker taker = new NadTaker();
9              taker.setSid(rs.getInt("sid"));
10             taker.setSname(rs.getString("sname"));
11             taker.setSnumber(rs.getString("snumber"));
12             taker.setSage(rs.getInt("sage"));
13             taker.setSphone(rs.getString("sphone"));
14             taker.setSaddress(rs.getString("saddress"));
15             taker.setState(rs.getString("state"));
16             taker.setStime(rs.getString("stime"));
17             takers.add(taker);
18         }
19         DBUtil.releaseAll();
20         return takers;
21     } catch (SQLException ex) {
22         Logger.getLogger(UserDaoImp.class.getName()).log(Level.SEVERE, null, ex);
23     }
24     return null;
25 }
```

## 2.5 检测人员注册功能实现

在一个人进行检测之前首先需要将其身份信息登记进入系统中，这一功能在选择菜单中的登记按钮下实现，点击这一按钮会弹出登记界面，在文本框中输入对应内容后点击注册，系统会读取文本框中内容并在数据库的 taker 表中 insert 一条对应的记录。关键函数 registerTaker 代码如下：

```
1  @Override
2  public boolean registerTaker(NadTaker nadTaker) {
3      String insert = "insert into taker(sname,snumber,sage,sphone,saddress,state,stime) " + "values('"
4          + nadTaker.getSname() + "','" + nadTaker.getSnumber() + "','" + nadTaker.getSage() + "','"
5          + nadTaker.getSphone() + "','" + nadTaker.getSaddress() + "','" + nadTaker.getState() + "','"
6          + nadTaker.getSime() + "')";
7
8      try {
9          // System.out.println(insert);
10         DBUtil.runUpdate(insert);
11         return true;
12     } catch (SQLException ex) {
13         Logger.getLogger(UserDaoImp.class.getName()).log(Level.SEVERE, null, ex);
14     }
15     return false;
16 }
```

## 2.6 检测人员登记功能实现

我们可以通过点击选择菜单中的登记按钮对已经录入系统的人员进行核酸检测结果登记，在弹出的界面的文本框中输入其身份证号和检测时间、检测结果后，系统会读取文本框中的内容，执行 checkNadTaker 函数对 user 表中对应身份证号的记录进行核酸检测时间和核酸检测结果的更新：

```
1  @Override
2  public boolean checkNadTaker(String number, String time, String res) {
3      String select = "select * from taker where snumber='" + number + "'";
4      try {
5          ResultSet rs = DBUtil.runQuery(select);
6          if (rs.next()) {
7              String update = "update taker set stime='" + time + "', state='" + res + "' where snumber='"
8                  + number
9                  + "'";
10             DBUtil.runUpdate(update);
11             return true;
12         }
13     } catch (SQLException ex) {
14         Logger.getLogger(UserDaoImp.class.getName()).log(Level.SEVERE, null, ex);
15     }
```

```
15     }
16     return false;
17 }
```

## 2.7 查询功能实现

点击选择菜单中的查询按钮，在弹出的界面的文本框中输入需要查询的身份证号后点击查询按钮，系统会读取你输入的身份证号后在数据库的 taker 表中进行查询并返回对应的核酸检测结果：

```
1  @Override
2  public String getAnswer(String number) {
3      String select = "select state from taker where snumber='" + number + "'";
4      try {
5          ResultSet rs = DBUtil.runQuery(select);
6          while (rs.next()) {
7              return rs.getString("state");
8          }
9
10     } catch (SQLException ex) {
11         Logger.getLogger(UserDaoImp.class.getName()).log(Level.SEVERE, null, ex);
12     }
13     return "查无此人";
14 }
```



### 3 用户界面交互展示

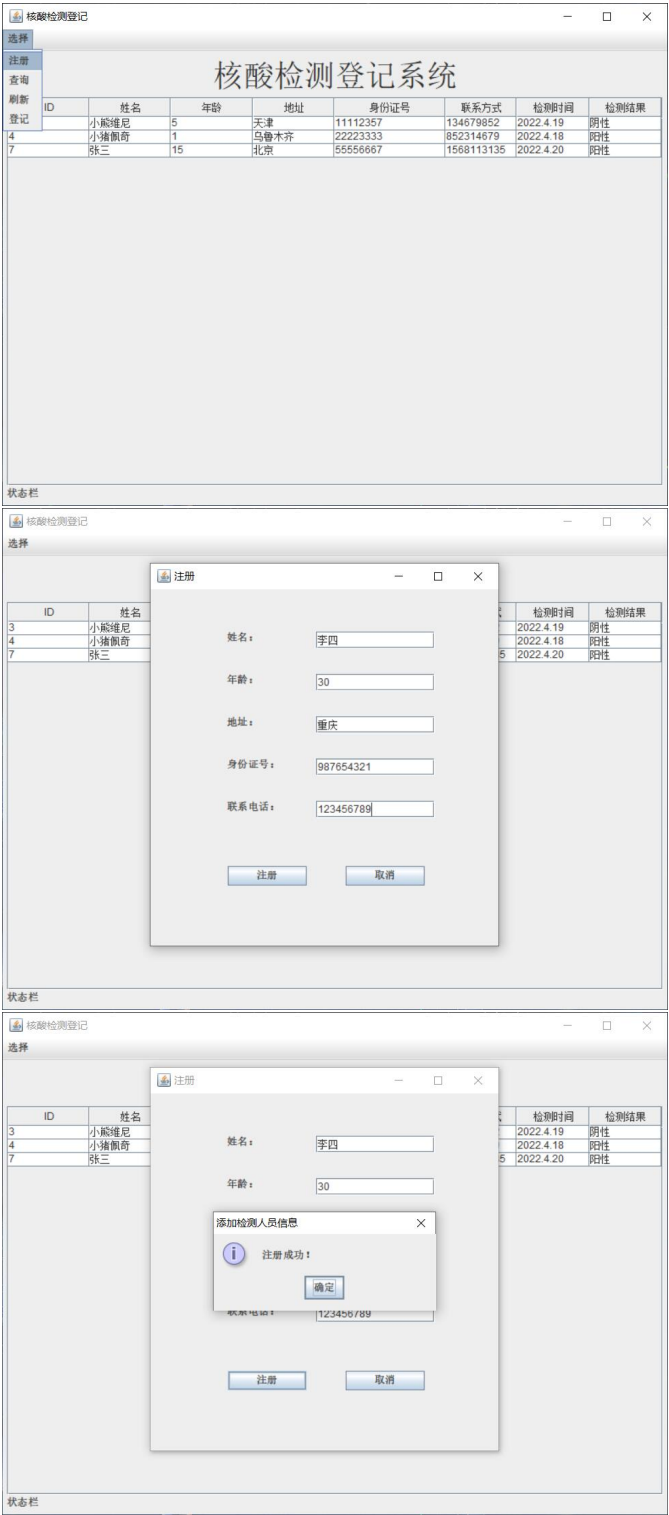
#### 3.1 登录



#### 3.2 登录后的界面



3.3 用户注册



注册成功后刷新便可以看到人员信息

核酸检测登记

选择

注册

查询

刷新

登记

核酸检测登记系统

ID	姓名	年龄	地址	身份证号	联系方式	检测时间	检测结果
4	小熊维尼	5	天津	11112357	134679852	2022.4.19	阴性
7	小猪佩奇	1	乌鲁木齐	22223333	852314679	2022.4.18	阳性
7	张三	15	北京	55556667	1568113135	2022.4.20	阳性

状态栏

核酸检测登记

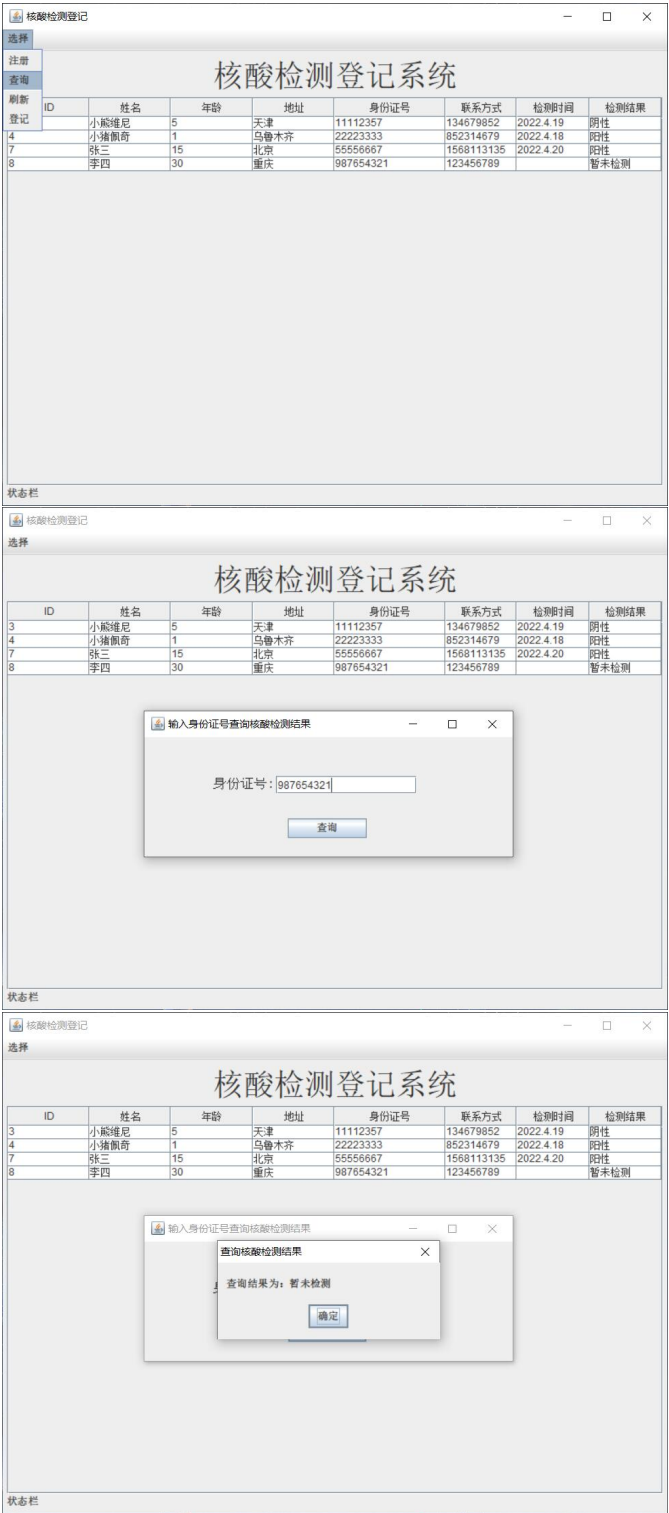
选择

核酸检测登记系统

ID	姓名	年龄	地址	身份证号	联系方式	检测时间	检测结果
3	小熊维尼	5	天津	11112357	134679852	2022.4.19	阴性
4	小猪佩奇	1	乌鲁木齐	22223333	852314679	2022.4.18	阳性
7	张三	15	北京	55556667	1568113135	2022.4.20	阳性
8	李四	30	重庆	987654321	123456789		暂未检测

状态栏

3.4 查询



核酸检测登记

选择

核酸检测登记系统

ID	姓名	年龄	地址	身份证号	联系方式	检测时间	检测结果
3	小熊维尼	5	天津	11112357	134679852	2022.4.19	阴性
4	小猪佩奇	1	乌鲁木齐	22223333	852314679	2022.4.18	阳性
7	张三	15	北京	55556667	1568113135	2022.4.20	阳性
8	李四	30	重庆	987654321	123456789		暂未检测

输入身份证号查询核酸检测结果

身份证号: 55556667

查询

状态栏

核酸检测登记

选择

核酸检测登记系统

ID	姓名	年龄	地址	身份证号	联系方式	检测时间	检测结果
3	小熊维尼	5	天津	11112357	134679852	2022.4.19	阴性
4	小猪佩奇	1	乌鲁木齐	22223333	852314679	2022.4.18	阳性
7	张三	15	北京	55556667	1568113135	2022.4.20	阳性
8	李四	30	重庆	987654321	123456789		暂未检测

输入身份证号查询核酸检测结果

查询核酸检测结果

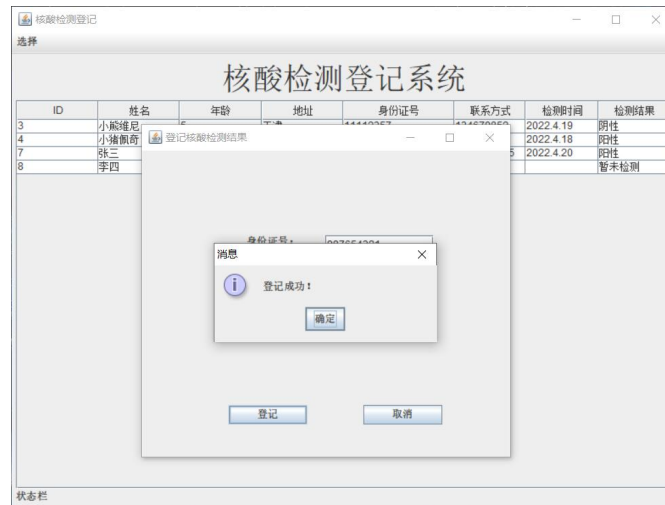
查询结果为: 阳性

确定

状态栏

3.5 登记检测结果





刷新后可以看到新登记的检测结果





## 4 角色轮换安排

我们的角色轮换较为明确，在编写数据库和设计数据库和 java 编程的后端这两部分时，由何坤彬充当领航员，朱浩泽充当驾驶员；在进行 java 部分的可视化操作的前端部分由朱浩泽充当领航员，何坤彬充当驾驶员。

## 5 代码复审及讨论改进的过程

在最初，我们设计的 mysql 表非常简单，只有一个检测结果表（身份证号，姓名，检测结果）和一个用户表（用户名，密码），但此时领航员立刻指出，这样无法将用户表和检测结果表关联起来，所以增加了用户 id 这一特征将两张表中的内容进行关联。

随后，领航员又一次指出，这样设计的功能过于薄弱，核酸检测结果可能随时需要更新，所以又增加了检测时间这一特征，将检测结果和检测时间关联起来。随后，又考虑到溯源和通知的问题，我们又加入了被测人的家庭住址和联系方式，方便进行通知和溯源工作。



在进行 java 编程时，领航员随时关注着驾驶员的操作进度，驾驶员每次操作完成后，领航员会提醒他，并且提供一个操作完成后的提示，这样驾驶员就可以更加清晰的了解自己的操作进度。每次驾驶员编写代码时，领航员根据功能指挥驾驶员添加注释信息等，并按照代码规范对驾驶员进行变量命名的提示，每完成一个模块，驾驶员都要对代码进行测试，测试代码的正确性。领航员在每次驾驶员编写完代码进行测试后，对领航员编写的代码进行

## 6 双人合作的工作照片



## 7 对方编程习惯总结和性格评价

## 8 心得体会