

PaperPass[免费版]AIGC检测报告

简明打印版

AIGC总体疑似度(高+中+轻): 7.91%

AIGC总体疑似度(高+中+轻): 4.99% (加权计算)

高度疑似AIGC占全文比: 0.0%

中度疑似AIGC占全文比: 7.91%

轻度疑似AIGC占全文比: 0.0%

不予检测文字占比: 0.0%

检测版本: 免费版(仅检测中文)

报告编号: ARV16904512648C9F

论文题目: 实验报告

论文作者: 佚名

论文字数: 2215

段落个数: 25

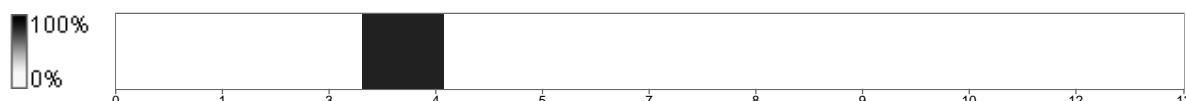
句子个数: 65

片段个数: 13

提交时间: 2025-10-31 14:03:18

查询真伪: <https://www.paperpass.com/check>

疑似度分布图:



正文中片段的不同颜色表示不同的疑似度范围

- 红色: AIGC生成疑似度在70%以上 (高度疑似)
- 橙色: AIGC生成疑似度在60%~70% (中度疑似)
- 紫色: AIGC生成疑似度在50%~60% (轻度疑似)
- 黑色: AIGC生成疑似度在50%以下
- 灰色: 过短片段、标题和英文等不予检测的文字

注: AIGC检测可有效识别文本是否部分或全部由AI模型生成, 检测结果与论文质量无关、仅表示论文中内容片段存在AI生成可能性的概率。

加权计算公式: (片段1疑似度 + 片段2疑似度 + ... + 片段n疑似度) / n
片段疑似度范围0.0~1.0, 合格片段按照0计算, 不计算不予检测文字

嘉宾信息管理系统设计与建模实验报告

1. 软件的主要功能

写在最前面——我认为嘉宾信息管理系统的难度是远大于天气 APP 和任务单 APP 的，当然你可以认为三者难度差不多，也可以认为嘉宾信息管理系统的难度远小于天气 APP 和任务单 APP，幸好我们有这样想的自由。

所以我只能实现我预想中一些很基本的模块。代码使用 C++ 语言写成，通过 Qt 6.8.0 中的 qmake 编译运行（我是新手），数据库使用了 Navicat 17 for SQLite 实现，其版本为 Sqlite3，我实现了登录界面与主界面的 GUI，但在交互层面设计远远达不到预期。还实现了嘉宾数据库的增添、删除、修改、查找四项基本功能。

1. **安全检测模块：**事实就是我只制作了运行在本地的软件，所以原定计划的 DDos 攻击预防，网络漏洞检测都是没有必要的。

我所能想到的安全检测模块包括登录界面的 sql 注入漏洞，验证码防护，将密码在传输到数据库的过程中通过转化为 MD5 以保存防止泄露，但考虑到实验 4 软件分析的任务是找漏洞，故在此处我没有实现这些基本功能。

2. **数据管理模块：**将嘉宾数据保存到数据库中（同上，为了在实验 4 中进行优化，故此处没有对于行程、电话号码等敏感信息进行统一加密，以保证数据的安全性），以防止数据的丢失与损坏。

3. **权限分级模块：**为每一位嘉宾设置对应的权限级别（user 与 administer），使得只有权限等级满足要求的接待组成员才可以查询（或管理）该嘉宾的行程计划、车辆安排等敏感信息。

4. **嘉宾信息模块：**将嘉宾 ID 与姓名相绑定，并指出嘉宾所处分组（可在多个组别中）设置该嘉宾权限级别，最后对其敏感信息进行加密存储（同上）。包括新建嘉宾信息和更新嘉宾信息两种方式以满足无重复和准确性。

我不认为录入时我的系统会发生异常情况，除非人数以万为单位，因为我没有实现多线程的增添、删除、查找、修改，无法解决用户等待不耐烦之后点击反复应用程序导致崩溃的情况。

2. 根据 UML 图实现代码

1. **写在前面:** 软件源代码总计 78.2kb, 除去 GUI 自动生成的约 430 行代码, 另有大约 550 行代码。

2. **将 UML 图转换为代码:** 我对 UML 图中指出的功能只选择了最基本的嘉宾登录功能进行了实现, 并将其与一个数据库连接, 并实现了数据库的增添、删除、修改、查找四项基本功能。

这样我们可以对嘉宾的行程进行规划, 通过电话号码及时联系嘉宾进行沟通, 并告诉了嘉宾为他提供服务的人员。

服务人员也可以通过了解嘉宾的年龄、性别、行程的相关信息提供更人性化的服务。

3. **使用大模型辅助代码实现:** 事实上我发现大模型在实验 3 这项任务的实现上远远比不上其在实验 2 中的功能, 甚至是有一些失望的, 大模型所作的一切就是翻译, 把 puml 语言翻译成 C++ 语言, 但对于我实现具体的功能没有任何帮助。

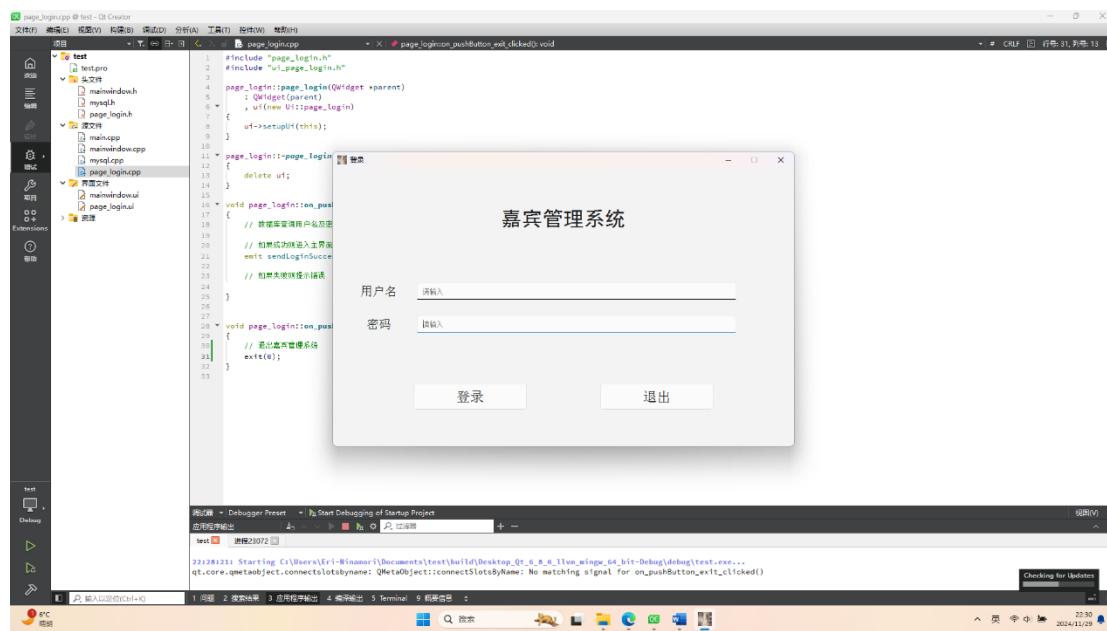
这就是事实, 在我多次添加额外细节信息后会得到更加详细的代码, 但它们往往不够统一, ChatGPT 4o 对我起到的帮助不比通过搜索引擎查询报错信息或者是查看文档、源码带来的帮助更有价值。

在本次实验中大语言模型对我起到的作用更像一个伙伴, 在我想要生成合适样例的时候快速生成样例, 为我节省时间。遇到报错时向其提问, 获得解决方法。

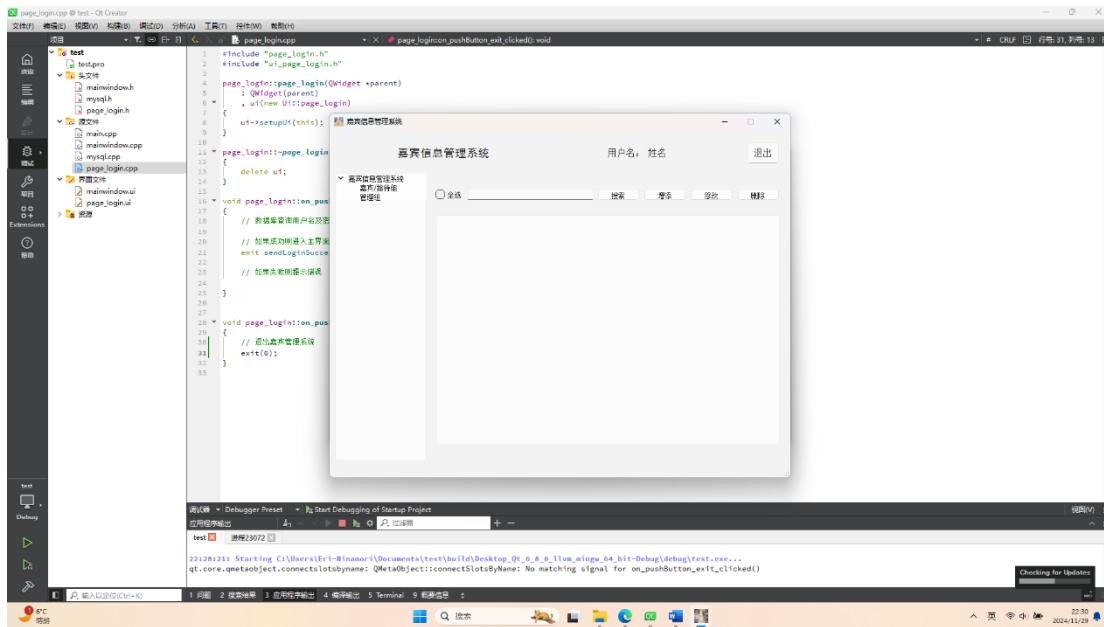
相较于 ChatGPT 4o, 本次实验中我设计的软件更多是通过学习 Qt 6.8.0 中已经实现好的 SQL Brower 范例完成的。

3. 代码编译与运行结果:

1. 登录界面的实现:



2. 嘉宾管理系统界面的实现: (此时还没有将数据导入到数据库当中)



3. 嘉宾信息（进入嘉宾管理系统后使用）与用户信息（登陆嘉宾管理系统、打印证件时使用）的实现：使用面向对象的思想，嘉宾信息的实现过程中，编号（id）自增以确保不相同；用户信息的实现过程中，用户名（username）通过 Unique 确保不可重复，保证了数据库使用时的安全性、准确性。

```

7  struct GuestInfo {
8      quint32 id;
9      QString name;
10     quint8 age;
11     QString sex;
12     QString phone;
13     quint32 guestid;
14     QString scheduling;
15     QString server;
16 };
17
18 struct UserInfo {
19     QString username;
20     QString password;
21     QString authority;
22 };

```

4. 嘉宾信息的增添测试以及运行结果：

```

13     init();
14     GuestInfo g;
15     g.name = "李天成";
16     g.age = 18;
17     g.sex = "男";
18     g.phone = "13912345678";
19     g.guestid = 123456789;
20     g.scheduling = "玄武湖公园";
21     g.server = "李佳宁";
22     addGuest(g);
23
24     g.name = "张书华";
25     g.age = 22;
26     g.sex = "女";
27     g.phone = "15898765432";
28     g.guestid = 987654321;
29     g.scheduling = "中山陵";
30     g.server = "王宇航";
31     addGuest(g);

```

The screenshot shows the Navicat for SQLite interface. The left sidebar displays the database structure with a 'Guest' table selected under the 'main' schema. The right pane shows the 'Guest' table's data in a grid format.

	id	name	age	sex	phone	guestid	scheduling	server
61	李天成	18	男	13912345678	123456789	玄武湖公园	李佳宁	
62	张书华	22	女	15898765432	987654321	中山陵	王宇航	
63	王子涵	27	男	18654329876	456789123	新街口	张语嫣	
64	刘雅静	30	女	13265478901	741852963	鼓楼广场	陈浩轩	
65	陈琪晖	35	女	15012348765	369258147	秦淮河畔	周静怡	
66	林志远	40	男	13876543210	258147369	栖霞山	刘晨曦	
67	周梦瑶	45	女	15678901234	192837465	浦口桥北	黄子睿	
68	董俊杰	50	男	17765432109	678123945	雨花台烈士陵园	赵梦婷	
69	赵雪琳	55	男	18987654321	834759216	江宁大学城	杨子骞	
70	郑雨然	60	女	13701234567	543216789	夫子庙	林梓豪	

5. 查询嘉宾总数测试以及运行结果：结果为 10，通过 qDebug() 将结果打印在屏幕下方

The screenshot shows the Qt Creator terminal window. It displays the output of the qDebug() statement, which prints the value 10. The terminal also shows other log messages related to the application's startup and signal connections.

```

105     qDebug() << getGuestCnt();
应用程序输出 + - 过滤器
test [进程中止] 23072
qt.core.QMetaObject.connectSlotsByName: QMetaObject::connectSlotsByName: No matching signal for on_pushButton_exit_clicked()
21:53:11: C:\Users\Eri-Ninamori\Documents\test\build\Desktop_Qt_6_8_0_llvm_mingw_64_bit-Debug\test.exe 退出, 退出代码: 0
{1 ?} {2?}

22:02:37: Starting C:\Users\Eri-Ninamori\Documents\test\build\Desktop_Qt_6_8_0_llvm_mingw_64_bit-Debug\test.exe...
qt.core.QMetaObject.connectSlotsByName: QMetaObject::connectSlotsByName: No matching signal for on_pushButton_exit_clicked()
10
22:03:11: C:\Users\Eri-Ninamori\Documents\test\build\Desktop_Qt_6_8_0_llvm_mingw_64_bit-Debug\test.exe 退出, 退出代码: 0
{1 ?} {2?}

```

6. 查询某一页的嘉宾信息测试以及运行结果：注意页数从 0 开始（符合计算机世界）

```

72 // g.phone = "15678901234";
73 // g.guestid = 192837465;
74 // g.scheduling = "浦口桥北";
75 // g.server = "狮子巷";
76 // addGuest(g);
77
78 // g.name = "黄浩然";
79 // g.age = 58;
80 // g.sex = "男";
81 // g.phone = "17765432109";
82 // g.guestid = 678123945;
83 // g.scheduling = "雨花台烈士公园";
84 // g.server = "梦剧场";
85 // addGuest(g);
86
87 // g.name = "赵雪琳";
88 // g.age = 55;
89 // g.sex = "男";
90 // g.phone = "18987654321";
91 // g.guestid = 834759216;
92 // g.scheduling = "江宁大学城";
93 // g.server = "狮子巷";
94 // addGuest(g);
95
96 // g.name = "郑浩然";
97 // g.age = 66;
98 // g.sex = "女";
99 // g.phone = "13701234567";
100 // g.guestid = 543216789;
101 // g.scheduling = "夫子庙";
102 // g.server = "林梓豪";
103 // addGuest(g);
104
105 // qDebug() << getGuestCnt();
106 // getPageGuest(0, 3);
107 qDebug() << delGuest(66);

```

7. 通过独有的 id 删除嘉宾信息的测试以及运行结果:

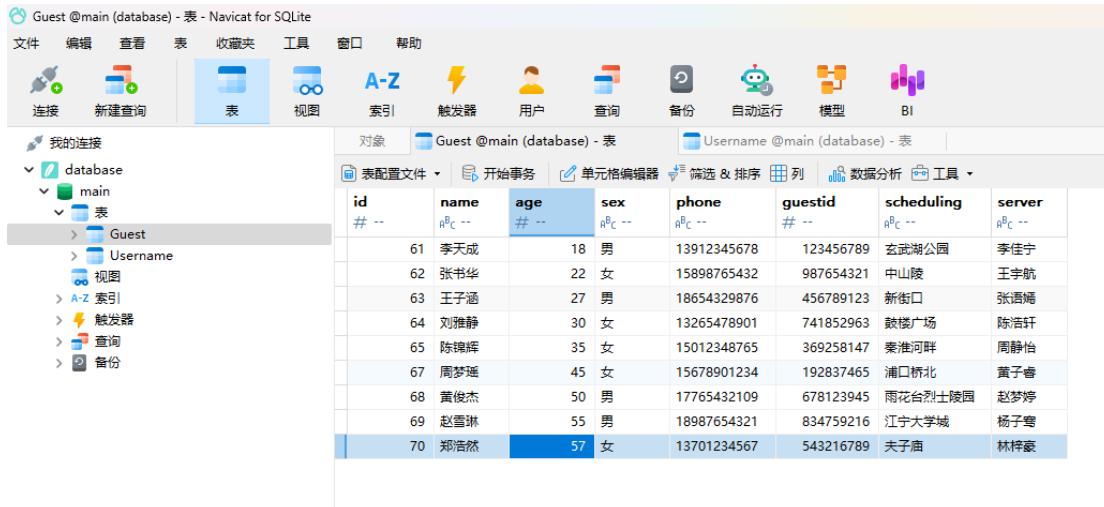
id	name	age	sex	phone	guestid	scheduling	server
61	李天成	18	男	13912345678	123456789	玄武湖公园	李佳宁
62	张书华	22	女	15898765432	987654321	中山陵	王宇航
63	王子涵	27	男	18654329876	456789123	新街口	张语嫣
64	刘雅静	30	女	13265478901	741852963	鼓楼广场	陈浩轩
65	陈锦辉	35	女	15012348765	369258147	秦淮河畔	周静怡
67	周梦瑶	45	女	15678901234	192837465	浦口桥北	黄子睿
68	黄俊杰	50	男	17765432109	678123945	雨花台烈士陵园	赵梦婷
69	赵雪琳	55	男	18987654321	834759216	江宁大学城	杨子睿
70	郑浩然	60	女	13701234567	543216789	夫子庙	林梓豪

8. 通过独有的 id 修改嘉宾信息的测试以及运行结果:

```

96 g.name = "郑浩然";
97 g.age = 60;
98 g.sex = "女";
99 g.phone = "13701234567";
100 g.guestid = 543216789;
101 g.scheduling = "夫子庙";
102 g.server = "林梓豪";
103 // addGuest(g);
104
105 // qDebug() << getGuestCnt();
106 // getPageGuest(0, 3);
107 // qDebug() << delGuest(66);
108 // clearGuestTable();
109 g.id = 70;
110 g.age = 57;
111 qDebug() << updateGuest(g);

```

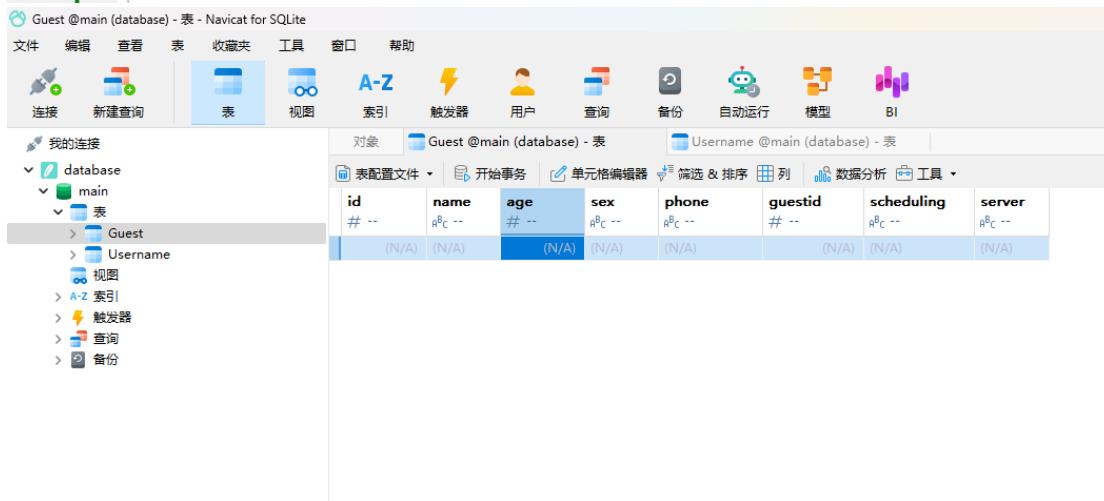


9. 删除所有嘉宾信息的测试以及运行结果：针对大规模出现错误情况时使用，后续可通过备份恢复至出错前状态（备份功能后续会考虑实现的）

```

105 // qDebug() << getGuestCnt();
106 // getPageGuest(0, 3);
107 // qDebug() << delGuest(66);
108 clearGuestTable();|
109 // g.id = 70;
110 // g.age = 57;
111 // qDebug() << updateGuest(g);

```



10. 用户信息的增添测试以及运行结果：

```

113     UserInfo u;
114     u.username = "Alice";
115     u.password = "X7q$eL92";
116     u.authority = "administer";
117     qDebug() << addUser(u);
118
119     u.username = "Alex";
120     u.password = "P@9wL3rT";
121     u.authority = "user";
122     qDebug() << addUser(u);
123
124     u.username = "Jack";
125     u.password = "J8k%Z1yQ";
126     u.authority = "user";
127     qDebug() << addUser(u);
128     qDebug() << addUser(u);
129
130     u.username = "Emma";
131     u.password = "T4m#G9nX";
132     u.authority = "user";
133     qDebug() << addUser(u);
134
135     u.username = "Bob";
136     u.password = "L2a!V7pR";
137     u.authority = "administer";
138     qDebug() << addUser(u);
139
140     u.username = "Lucy";
141     u.password = "Y6d&X3qZ";
142     u.authority = "user";
143     qDebug() << addUser(u);
144

```

调试器 - Debugger Preset | Start Debugging of Startup Project

应用程序输出 | 进程23072

```

22:11:13: C:\Users\Eri-Ninamori\Documents\test\build\Desktop_Qt_6_8_0_llvm_mingw_64_bit-Debug\debug\test.exe 退出, 退出代码: 0
{1 ?} {2?}

22:13:49: Starting C:\Users\Eri-Ninamori\Documents\test\build\Desktop_Qt_6_8_0_llvm_mingw_64_bit-Debug\debug\test.exe...
qt.core.qmetaobject.connectslotsbyname: QMetaObject::connectSlotsByName: No matching signal for on_pushButton_exit_clicked()
true
true
true
false
true
true
true

```

Username @main (database) - 表 - Navicat for SQLite

username	password	authority
Alice	X7q\$eL92	administer
Alex	P@9wL3rT	user
Jack	J8k%Z1yQ	user
Emma	T4m#G9nX	user
Bob	L2a!V7pR	administer
Lucy	Y6d&X3qZ	user

11. 检测用户名是否存在测试以及运行结果：在注册时用于判断合法用户名，大小写敏感，结果分别为 true 和 false，通过 qDebug() 将结果打印在屏幕下方

```

145     qDebug() << isExit("Bob");
146     qDebug() << isExit("bob");

```

应用程序输出 | 进程23072

```

true
22:15:55: The process was ended forcefully.

22:15:59: Starting C:\Users\Eri-Ninamori\Documents\test\build\Desktop_Qt_6_8_0_llvm_mingw_64_bit-Debug\debug\test.exe...
qt.core.qmetaobject.connectslotsbyname: QMetaObject::connectSlotsByName: No matching signal for on_pushButton_exit_clicked()
true
false
22:16:04: C:\Users\Eri-Ninamori\Documents\test\build\Desktop_Qt_6_8_0_llvm_mingw_64_bit-Debug\debug\test.exe 退出, 退出代码: 0
{1 ?} {2?}

```

12. 获取所有用户信息的测试以及运行结果:

The screenshot shows the Qt Creator IDE. The left pane displays the project structure for 'mysqlapp' with files like 'test.pro', 'mainwindow.h', 'mainwindow.cpp', 'mainwindow.ui', 'mysql.cpp', 'mysql.h', 'page_login.cpp', and 'page_login.h'. The right pane shows the code for the 'getAllUser()' function in 'mysql.cpp'. The code initializes a QList of UserInfo objects with various usernames, passwords, and authorities. A qDebug() call adds each user to the list. The bottom pane shows the debugger's stack, breakpoints, and threads.

```

112 // UserInfo u;
113 // u.username = "Alice";
114 // u.password = "X7q$eL92";
115 // u.authority = "administer";
116 // qDebug() << addUser(u);
117
118 // u.username = "Alex";
119 // u.password = "P@9wL3rT";
120 // u.authority = "user";
121 // qDebug() << addUser(u);
122
123 // u.username = "Jack";
124 // u.password = "J8k%Z1yQ";
125 // u.authority = "user";
126 // qDebug() << addUser(u);
127
128 // u.username = "Bob";
129 // u.password = "L2aIV7pR";
130 // u.authority = "administer";
131 // qDebug() << addUser(u);
132
133 // u.username = "Emma";
134 // u.password = "T4m#G9nX";
135 // u.authority = "user";
136 // qDebug() << addUser(u);
137
138 // u.username = "Lucy";
139 // u.password = "Y6d&X3qZ";
140 // u.authority = "user";
141 // qDebug() << addUser(u);
142
143 // qDebug() << isExit("Bob");
144 // qDebug() << isExit("bob");
145 // getAllUser();
146
147 // u.password = "$X9mL@5qR7!Zp8#T";
148 updateUser(u);
149
150
151 }

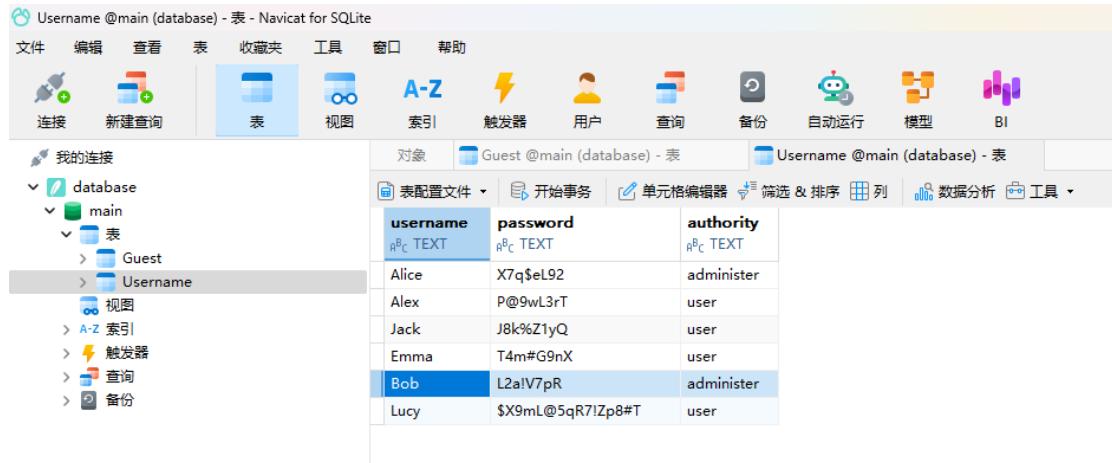
```

13. 通过独有的 username 修改嘉宾信息的测试以及运行结果: 换句话说仅可修改密码 (password) 与权限 (authority)

```

140
141     u.username = "Lucy";
142     u.password = "Y6d&X3qZ";
143     u.authority = "user";
144     // qDebug() << addUser(u);
145
146     // qDebug() << isExit("Bob");
147     // qDebug() << isExit("bob");
148     // getAllUser();
149     u.password = "$X9mL@5qR7!Zp8#T";
150     updateUser(u);

```

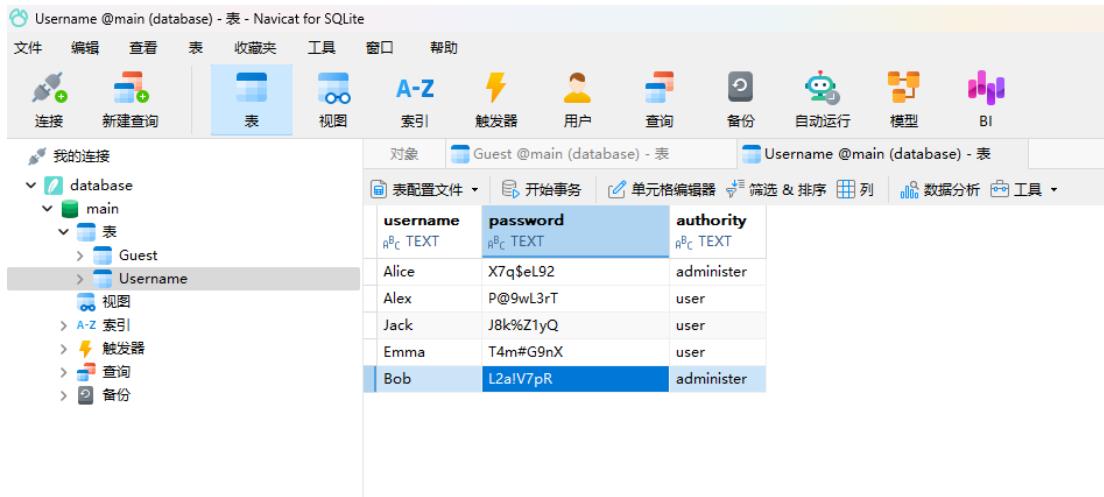


14. 通过独有的 username 删除嘉宾信息的测试以及运行结果:

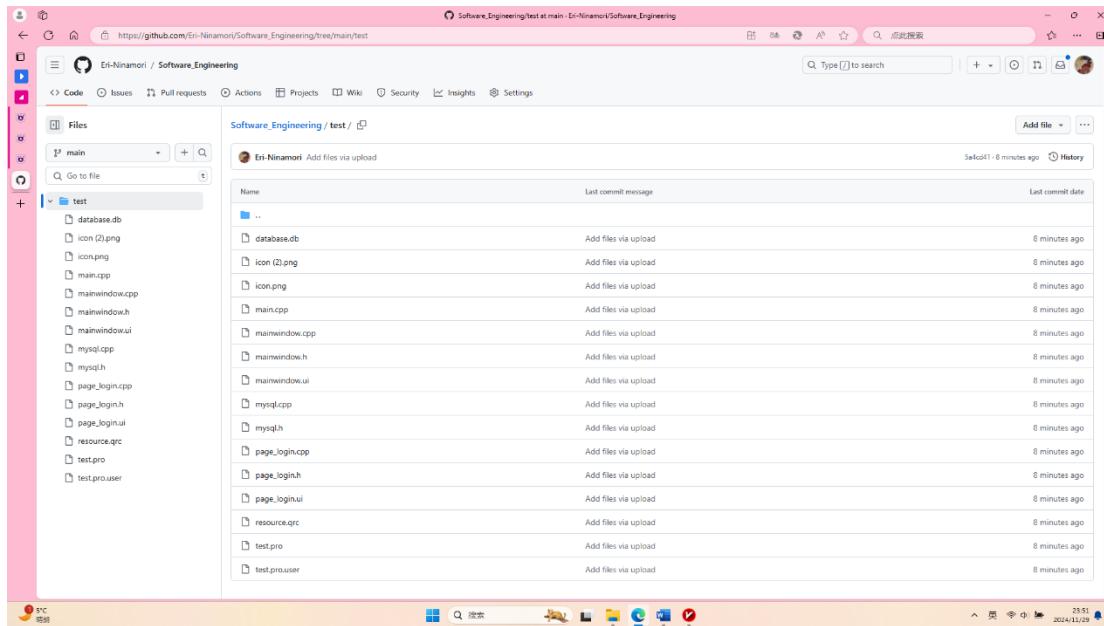
```

145     // qDebug() << isExit("Bob");
146     // qDebug() << isExit("bob");
147     // getAllUser();
148     // u.password = "$X9mL@5qR7!Zp8#T";
149     // qDebug() << updateUser(u);
150     qDebug() << delUser("Lucy");
151 }

```



4. Git 远程代码管理展示：我已经将代码上传至 github，并使用它
github desktop 进行管理了，要求太不明确了啊，我根本不知道
要说什么啊



5. 大语言模型使用心得：

本次实验中我使用了大模型帮我生成样例，毕竟手打样例是重复度很高的工作，同时借助ChatGPT 4o 帮我生成样例中虚构人物的姓名也让我脱离了构思顺耳姓名的困扰只是大语言模型生成的随机电话号码很奇怪，出现了“01234567”这种连续的情况。

不过使用大语言模型生成密码是一种很不错的方式，这算是一种收获了吧。