大学生心理健康数据库

姓名：

班级：

学号：

目录

1. 需求分析
2. 系统设计
3. 详细设计

3.1前端设计

功能实现

3.1.1学生端

3.1.2老师端

3.1.3管理员

3.2数据库设计

3.3后端设计

四、数据创建

五、数据库优化

1. 需求设计

现代社会中,大学生是一个特殊的群体。这里不乏精英和骄子,他们代表着祖国的未来。因此，他们能否健康成长是社会倍加关注的焦点。心理问题已成为影响大学生健康成长以及高校稳定的突出因素。而除了开展课内外相结合的心理健康育，对大学生的心理健康教育有着积极的影响和有益的作用以外，时刻关注大学生的心理也是不容忽视的，这里我们基于Javafx, Mysql构建了大学生心理健康数据库。

In modern society, college students are a special group.There is no shortage of elites and proud children here, they represent the future of the motherland.Therefore, whether they can grow up healthily is the focus of the society.Psychological problems have become a prominent factor affecting the healthy growth of college students and the stability of colleges and universities.In addition to carrying out the mental health education combined with inside and outside the class, which has a positive influence and beneficial effect on the mental health education of college students, it is also important to pay attention to the psychology of college students. Here, we built the mental health database of college students based on JavaFX and MySQL.

1. 系统设计

用户界面

实现

通过Java进行交互

相关数据库设计

图（2-1）

该系统设计如图（2-1），首先通过Javafx建立用户界面，这里用户界面有3个登录口，管理员登录，学生登录，老师登录，选择登录后用户界面所实现功能界面基于数据库设计，在实现相关功能时，通过Java建立连接调用相关基本表，视图达到实现目的。

1. 详细设计

3.1前端设计

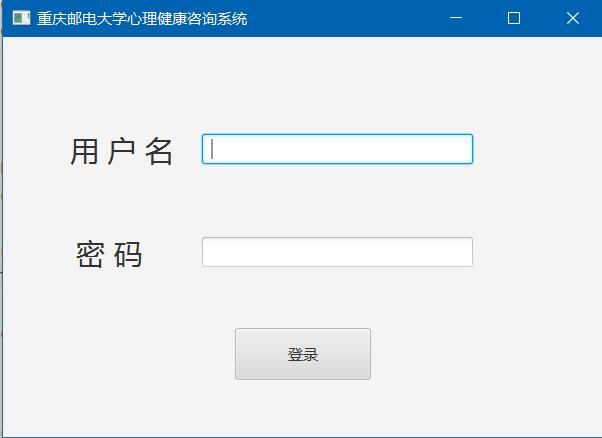
功能实现

基于Javafx搭建出用户界面，登录界面中有3个登录端口学生端，老师端以及管理员，如图（3-1-0）。



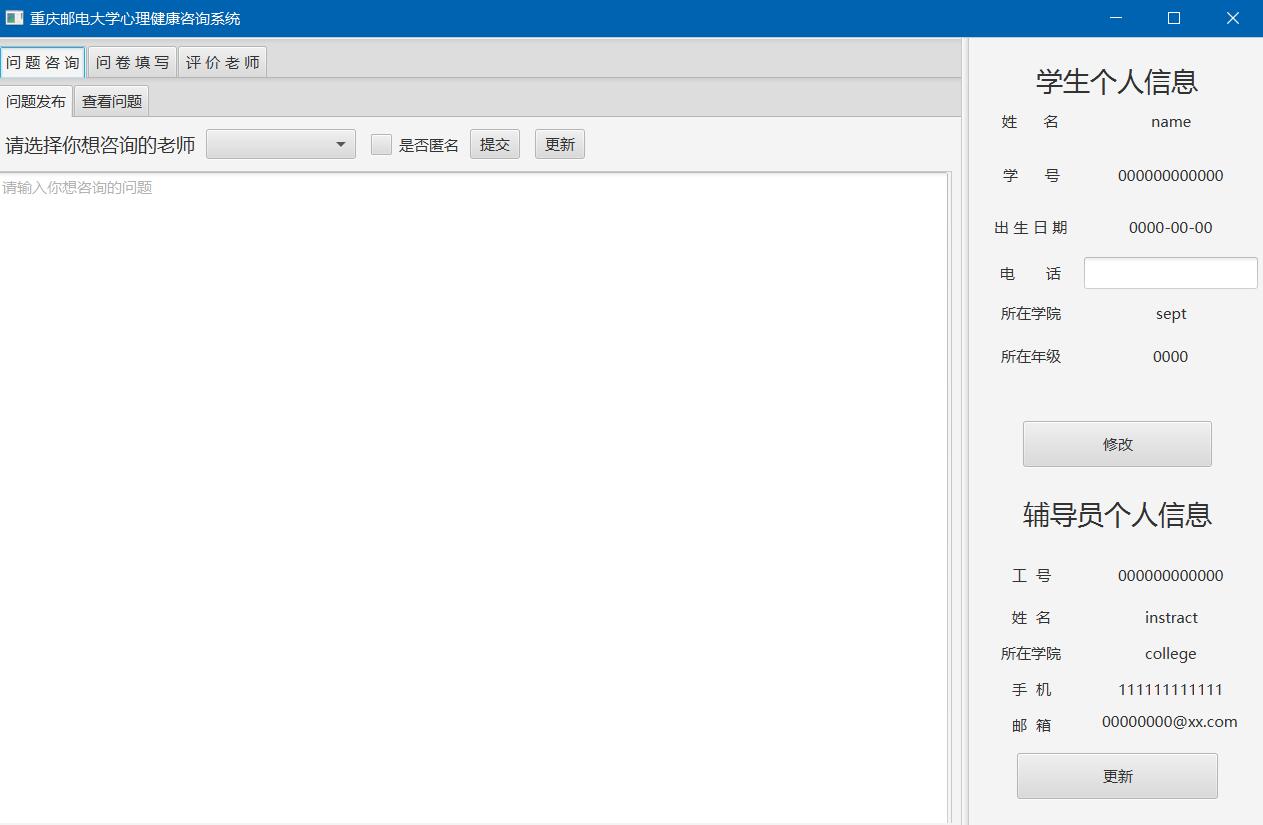
图（3-1-0）

3.1.1学生端



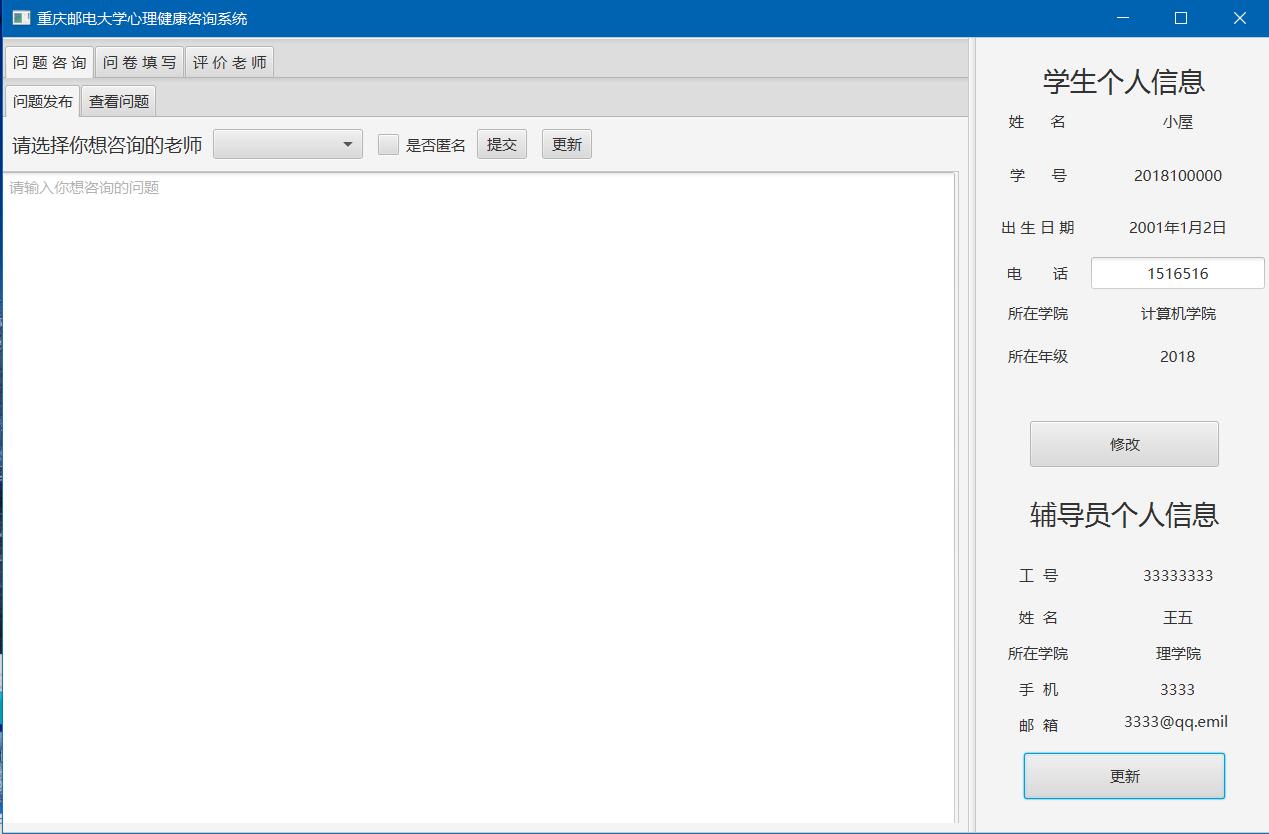
图（3-1-1-1）

当学生登录时出现登录界面如图（3-1-1-1）输入用户名与密码后出现默认界面，如图（3-1-1-2），若用户名或密码输入错误，则会有提示窗口需重新输入，右边点击更新后才能更新到用户自身信息，左边更新则是更新下拉框。



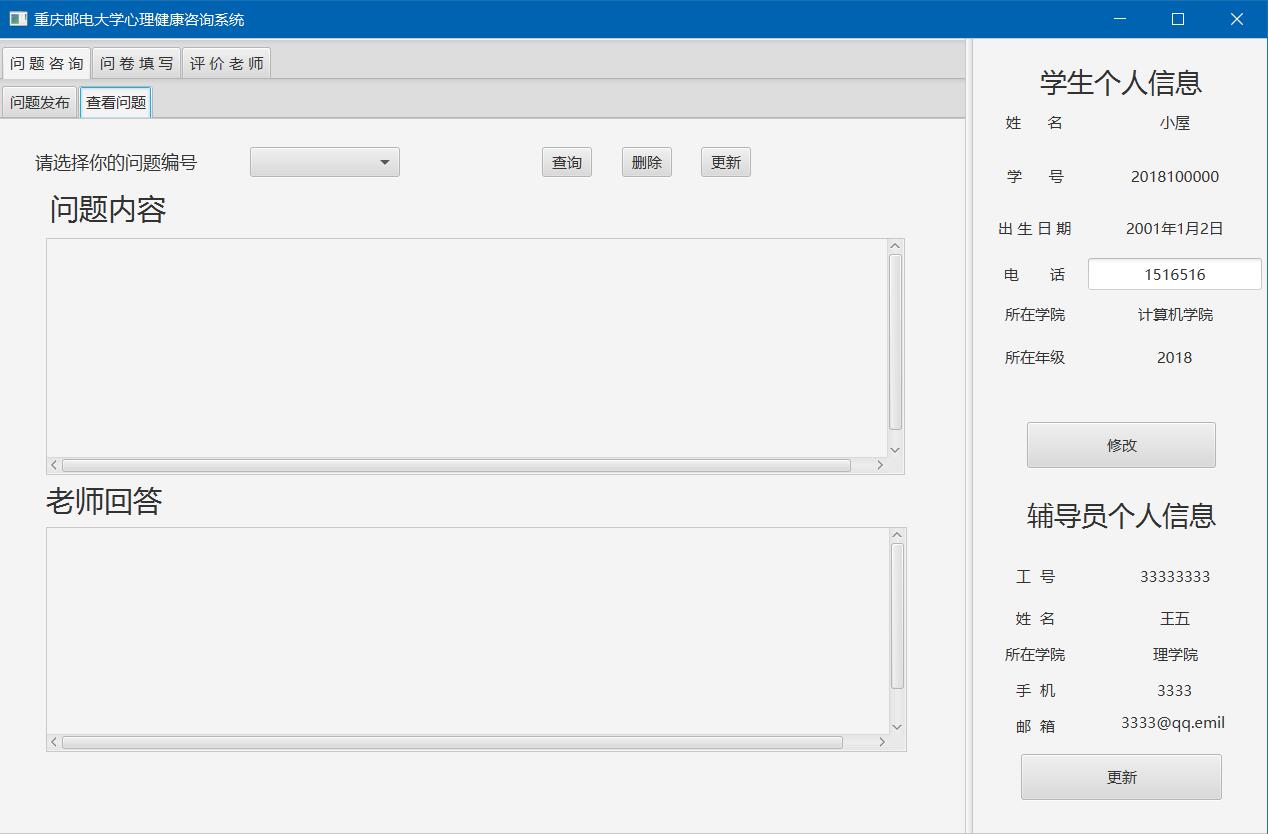
图（3-1-1-2）

问题查询界面学生可以修改自己的电话（修改后及时更新），发布问题，选择自己想咨询的老师，可以选择是否匿名发布，提交后可以查看自己的问题。（提交的问题会自动编号）如图（3-1-1-3）。



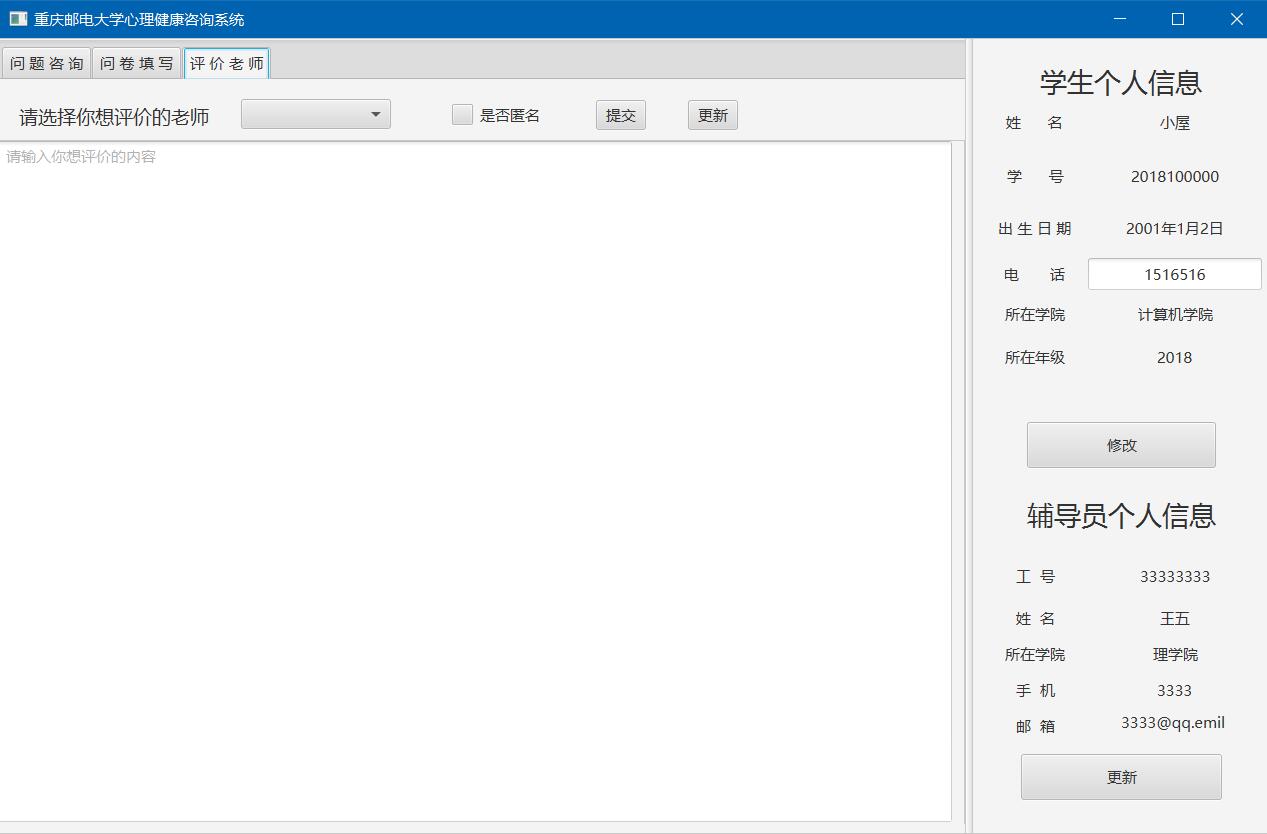
图（3-1-1-3）

查看问题界面可以选择问题的编号进行查看自己问题及老师的相应回答，如图（3-1-1-4）



图（3-1-1-4）

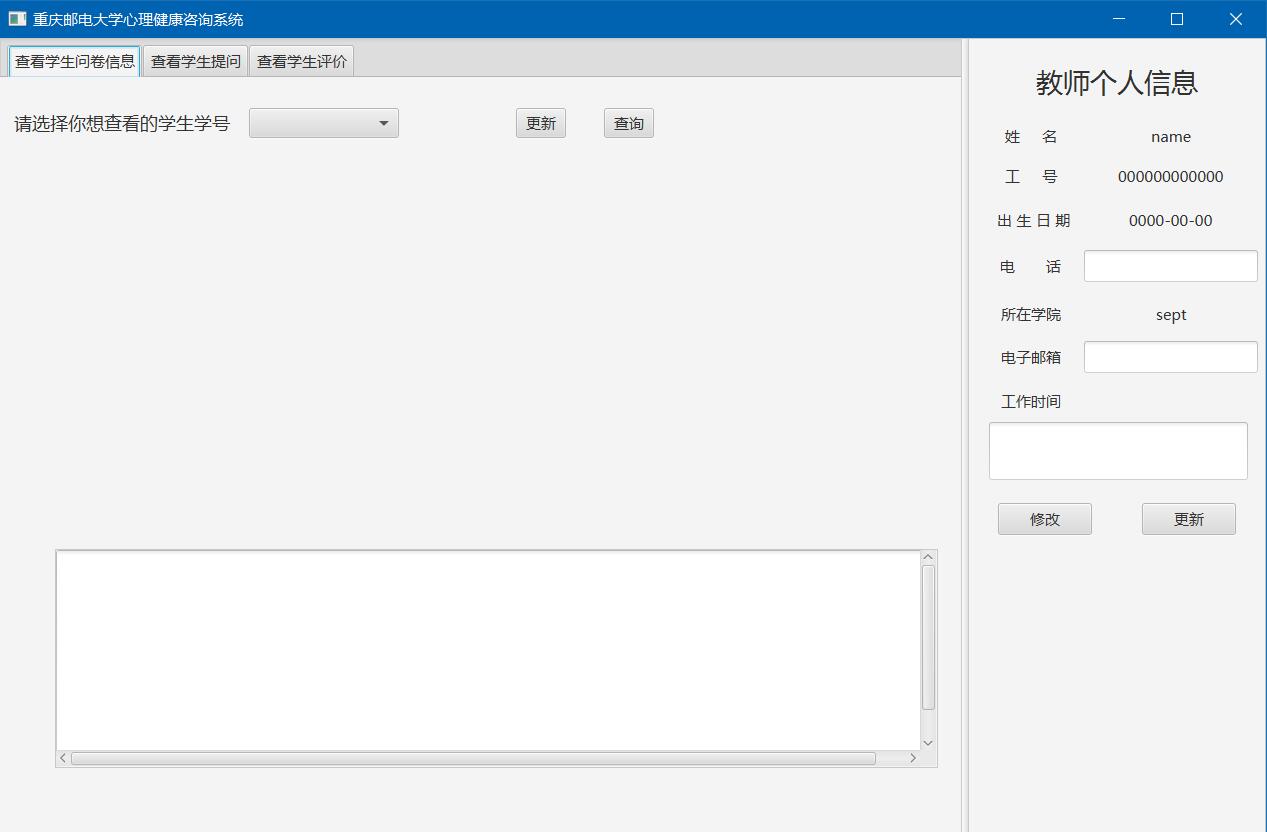
评价老师界面点更新后下拉框显示你想评价的老师并输出评价内容，如图（3-1-1-5）。



图（3-1-1-5）

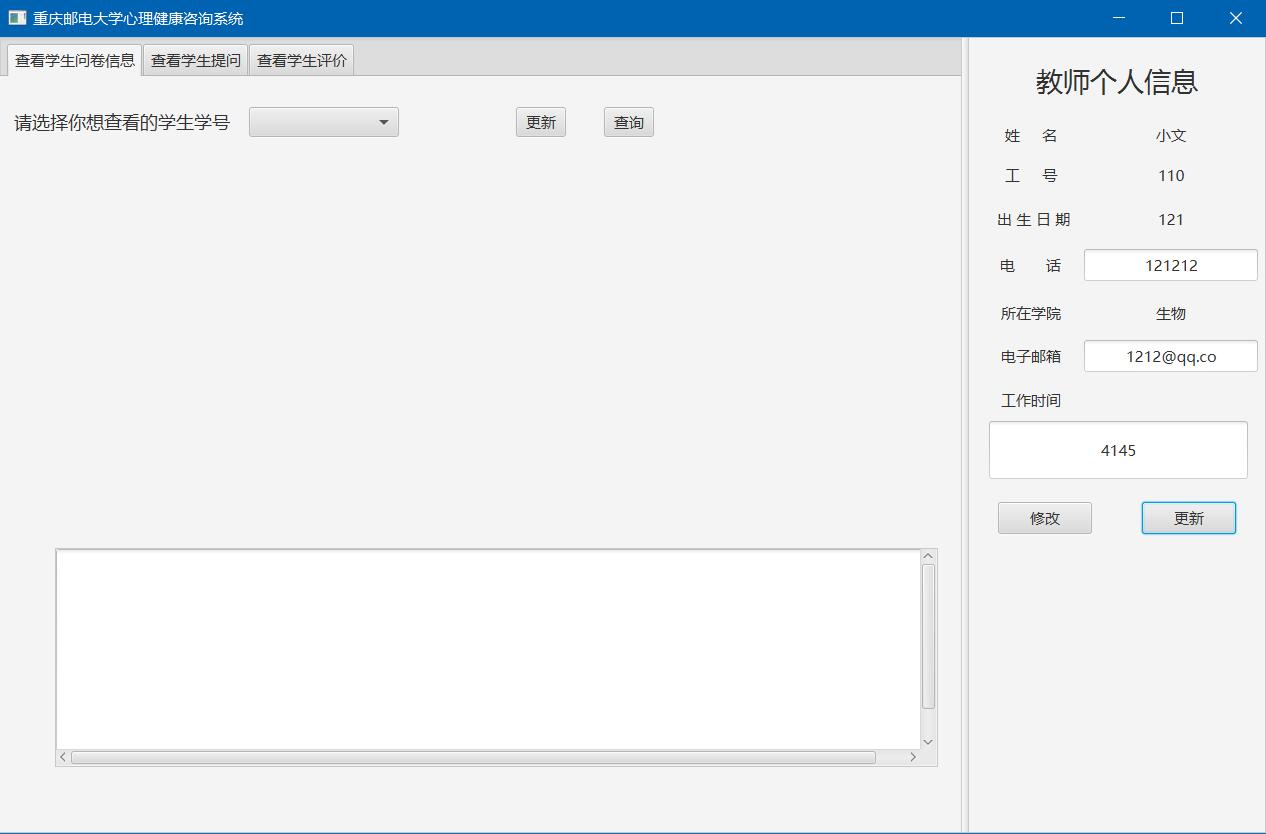
3.1.2老师端

选择老师登录后登录界面同图（3-1-1-1），输入用户名及密码，登录后出现默认用户界面，需要点击右边更新进行信息更新如图（3-1-2-1）。



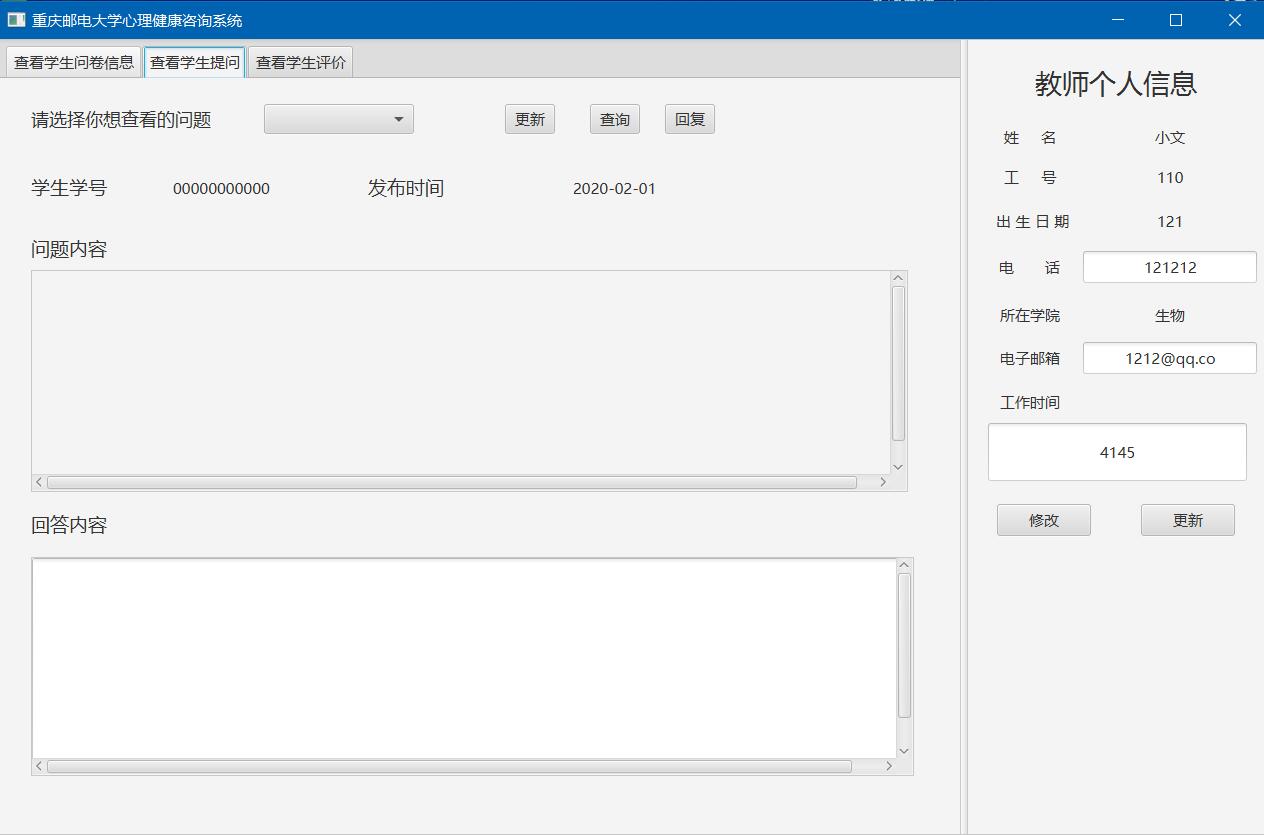
图（3-1-2-1）

查看学生问卷信息界面可以修改老师电话，电子邮箱，工作时间个人信息，左边更新后可以下拉选择想查看的学生学号如图（3-1-2-2）



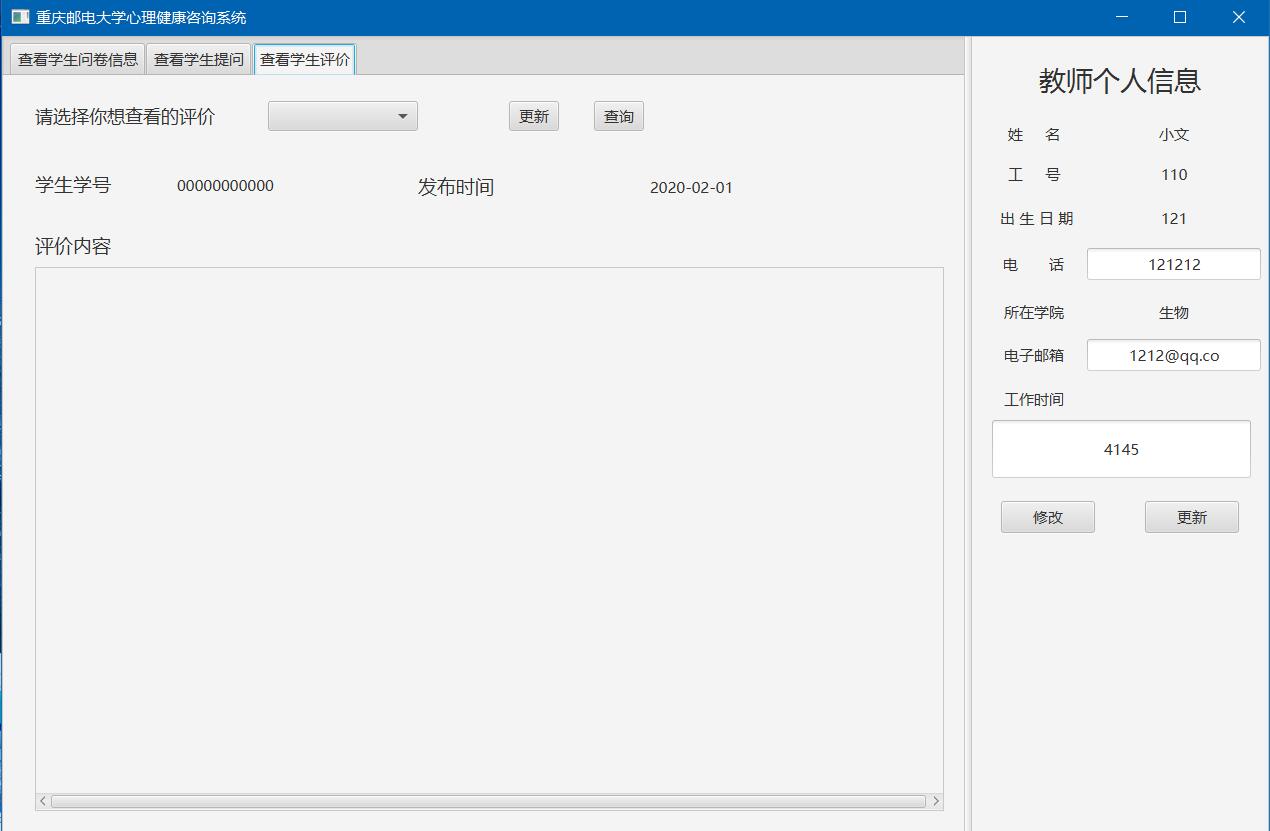
图（3-1-2-2）

查看学生提问界面更新下拉框后可以选择想查看的问题，查看问题内容并回答学生问题，如图（3-1-2-3）。



图（3-1-2-3）

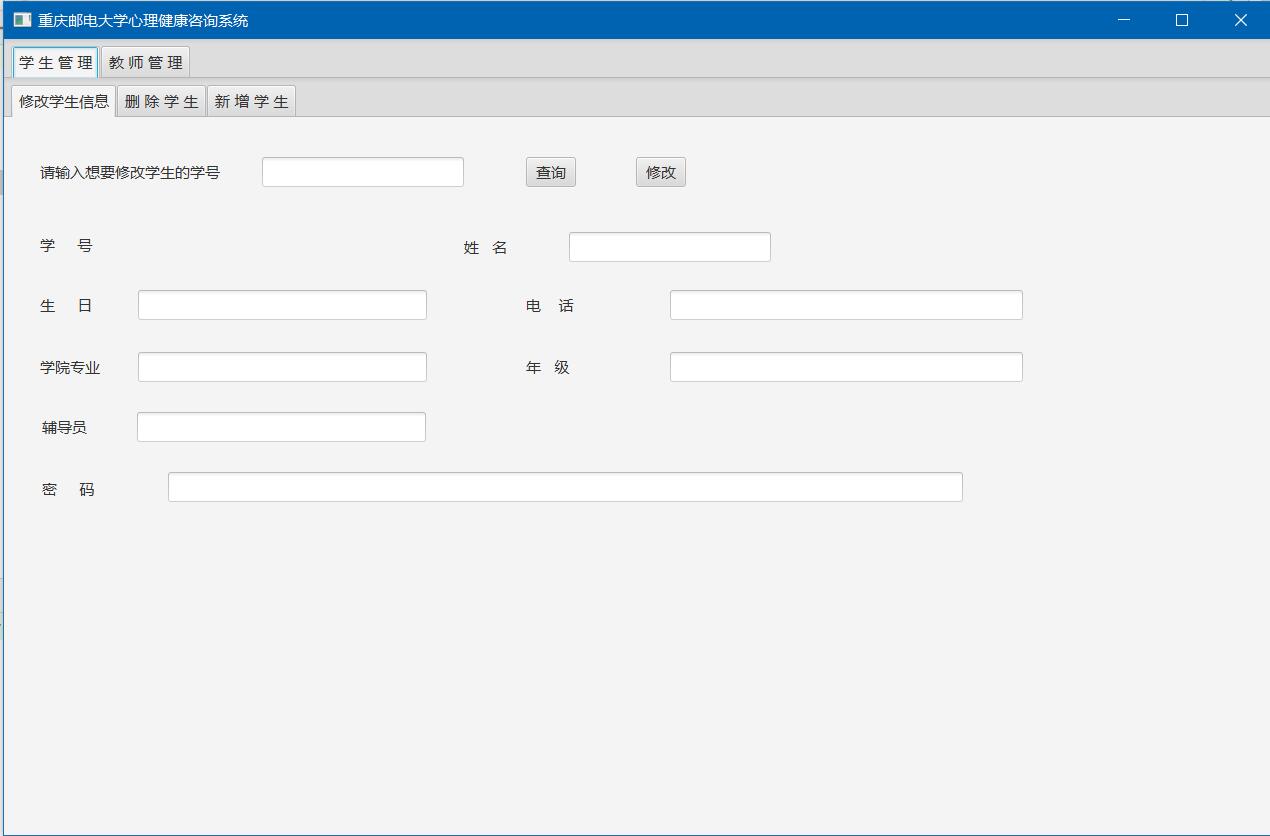
查看学生评价界面更新下拉框后可以查看学生对自己问题回答的评价如图（3-1-2-4）。



图（3-1-2-4）

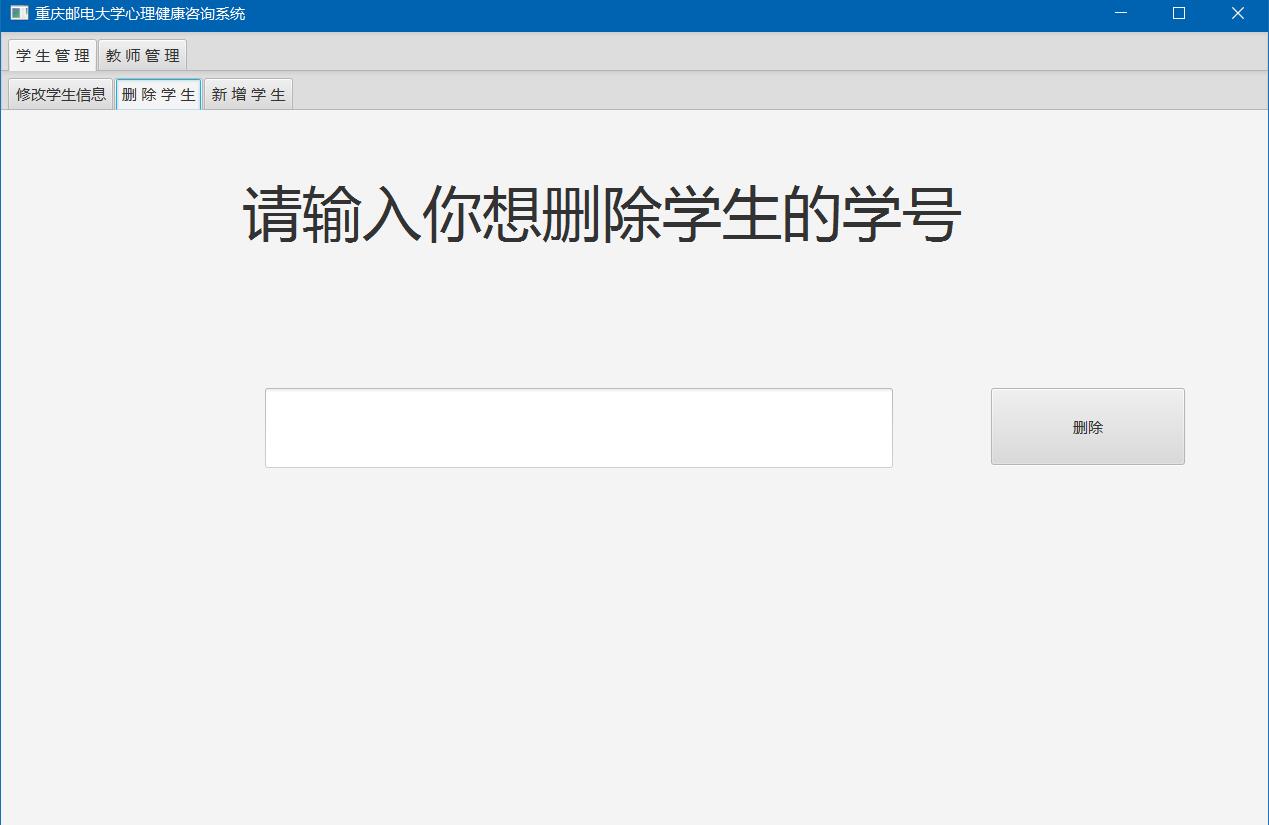
3.1.3管理员

管理员登录界面同图（3-1-1-1），登录后学生管理中修改学生信息界面输入想修改学生的学号，姓名，生日，电话等如图（3-1-3-1）。



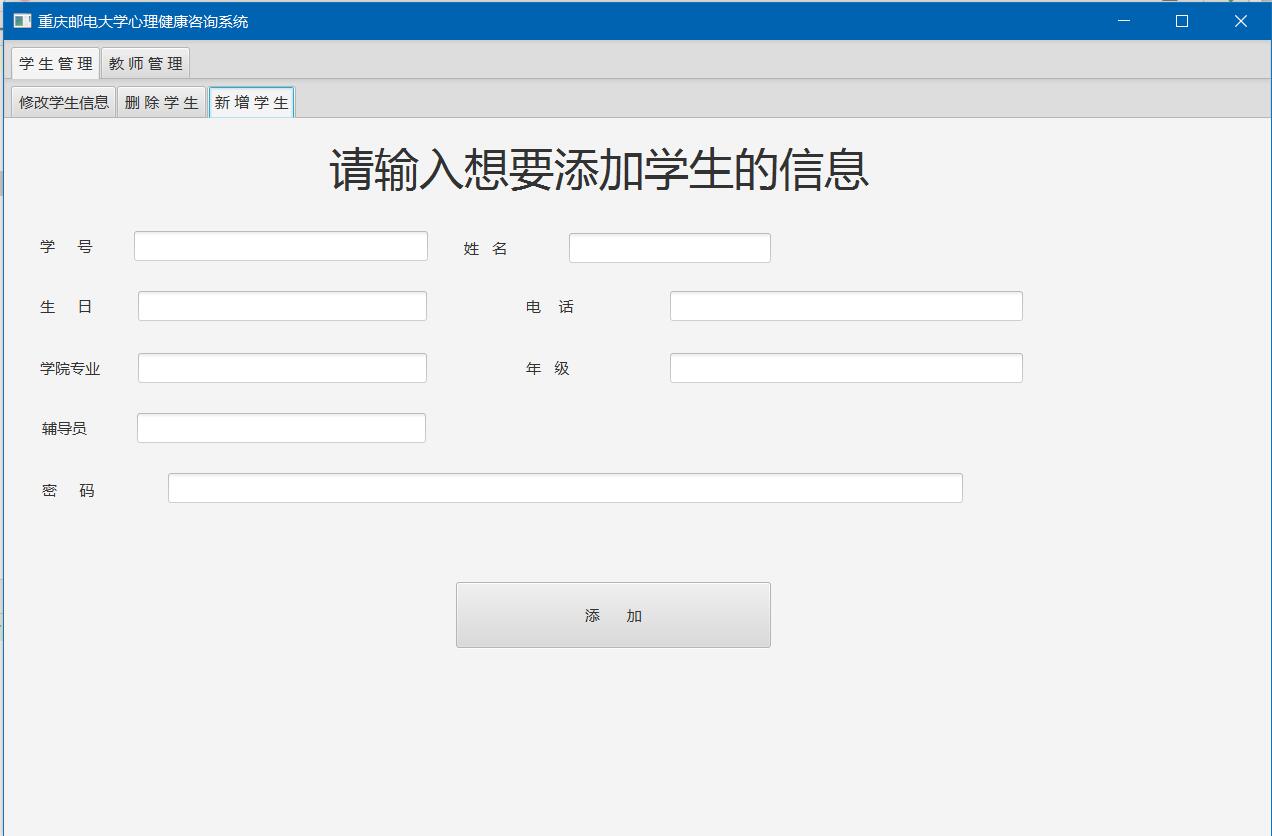
图（3-1-3-1）

学生管理中删除学生界面可以删除学生学号，如图（3-1-3-2）。



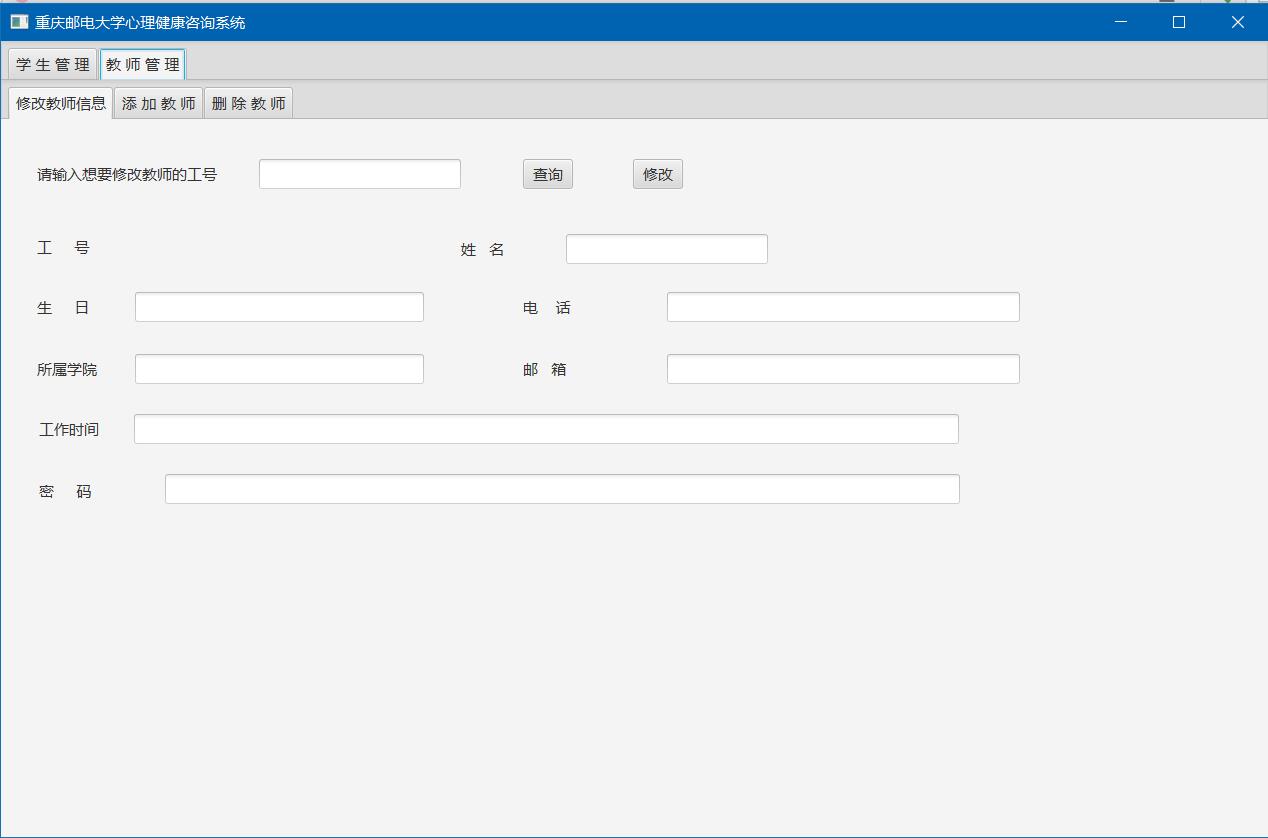
图（3-1-3-2）

学生管理中新增学生界面可以输入学生学号，姓名，生日等信息，如图（3-1-3-3）。



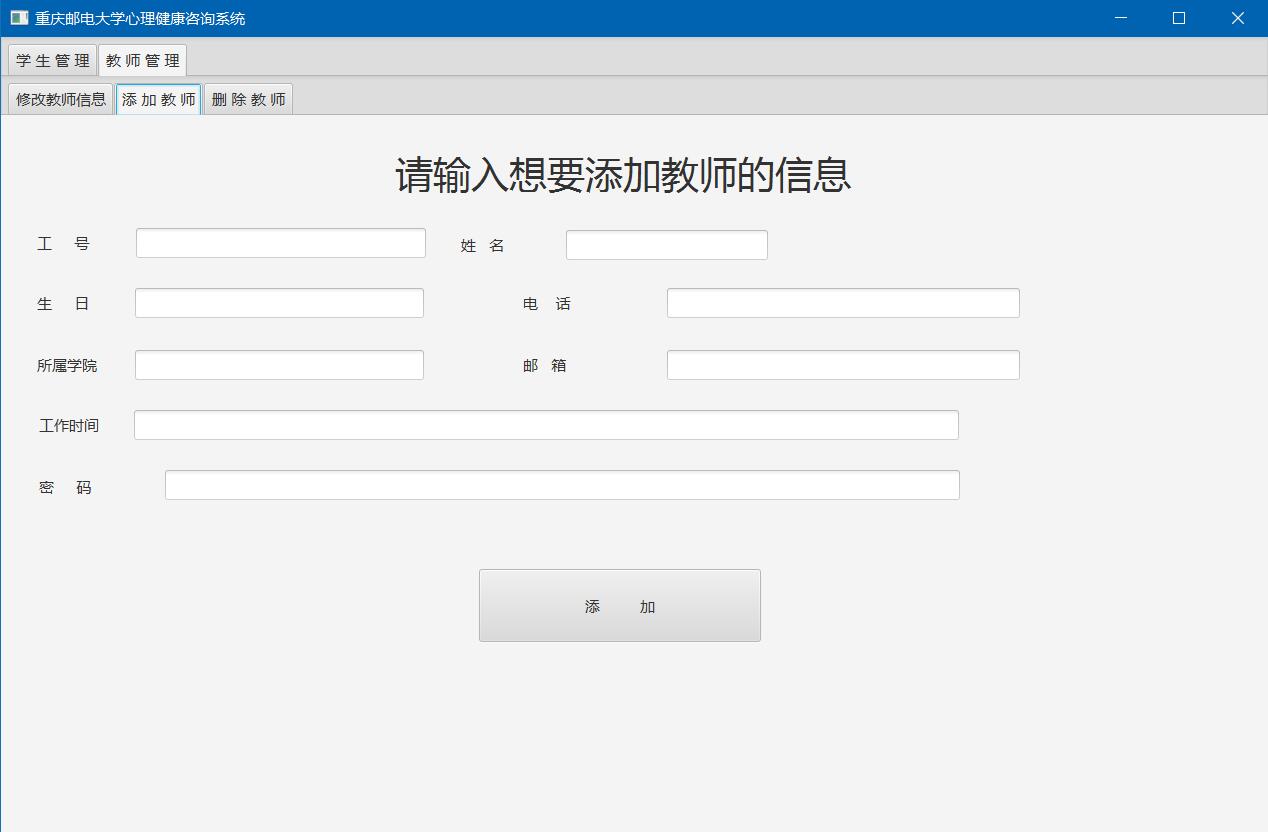
图（3-1-3-3）

教师管理中修改教师信息可以修改教师工号，姓名，生日等信息，如图（3-1-3-4）。



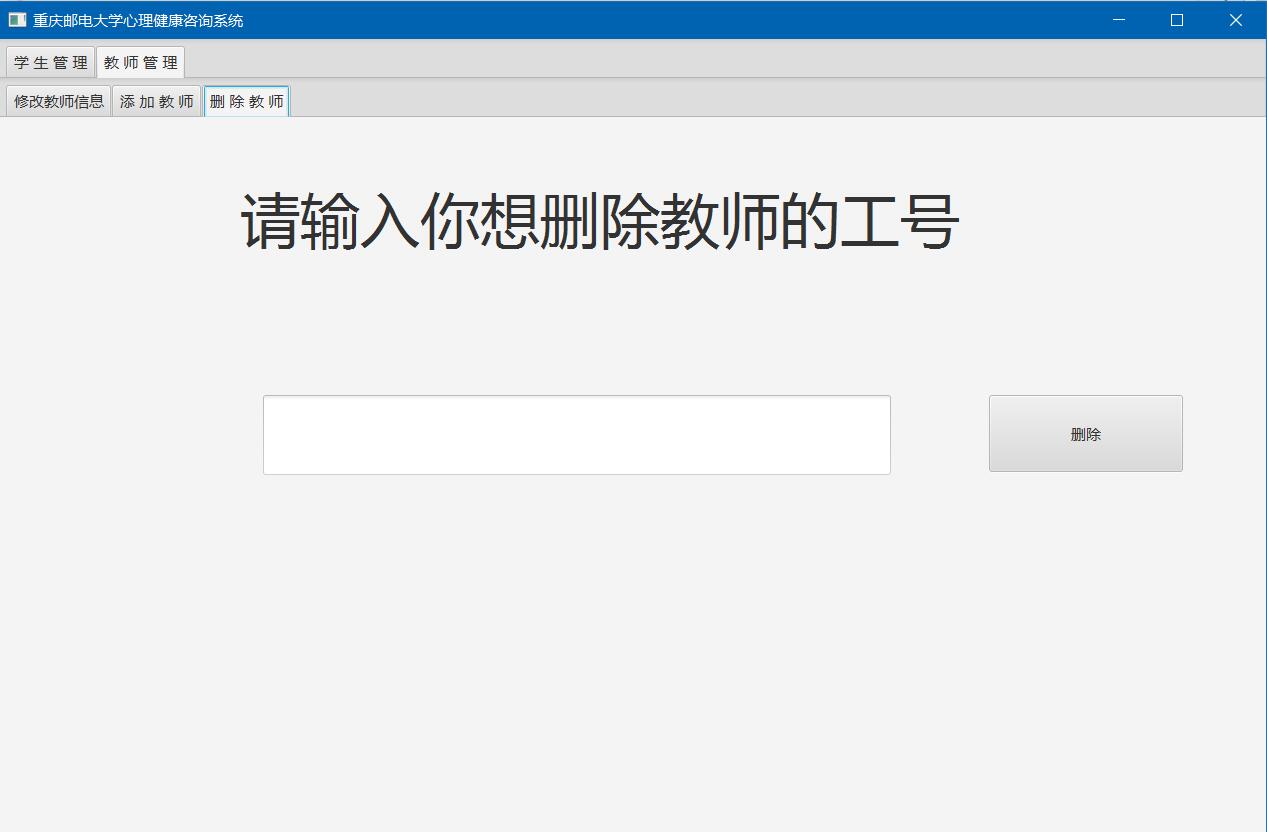
图（3-1-3-4）

教师管理中添加教师可以添加教师工号，姓名，生日等信息，如图（3-1-3-5）。



图（3-1-3-5）

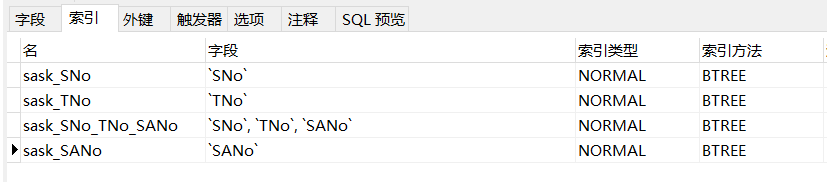
教师管理中删除教师界面输入想删除教师工号，如图（3-1-3-6）。



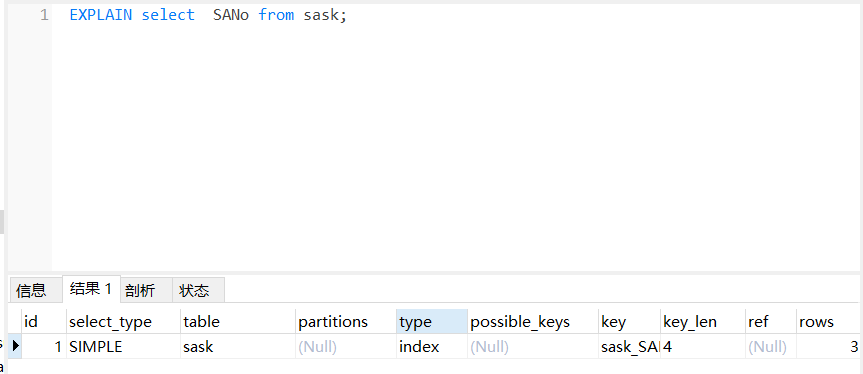
图（3-1-3-6）

3.2数据库设计

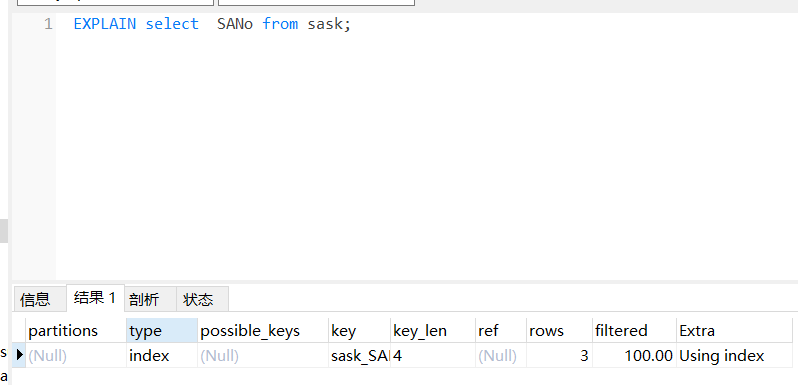
设计数据库我们首先想构建一个学生与老师的数据库，学生可以通过数据库选择并填写问卷，还可以选择老师进行提问，并查看老师给自己问卷及问题的反馈，老师可以查看学生的问题，这里我们建立了一个索引如图（3-2-1），图（3-2-2），图（3-2-3），这样可以大大减少查询的时间，老师能对学生问题进行回答，并查看学生对于自己的回答是否满意。



图（3-2-1）

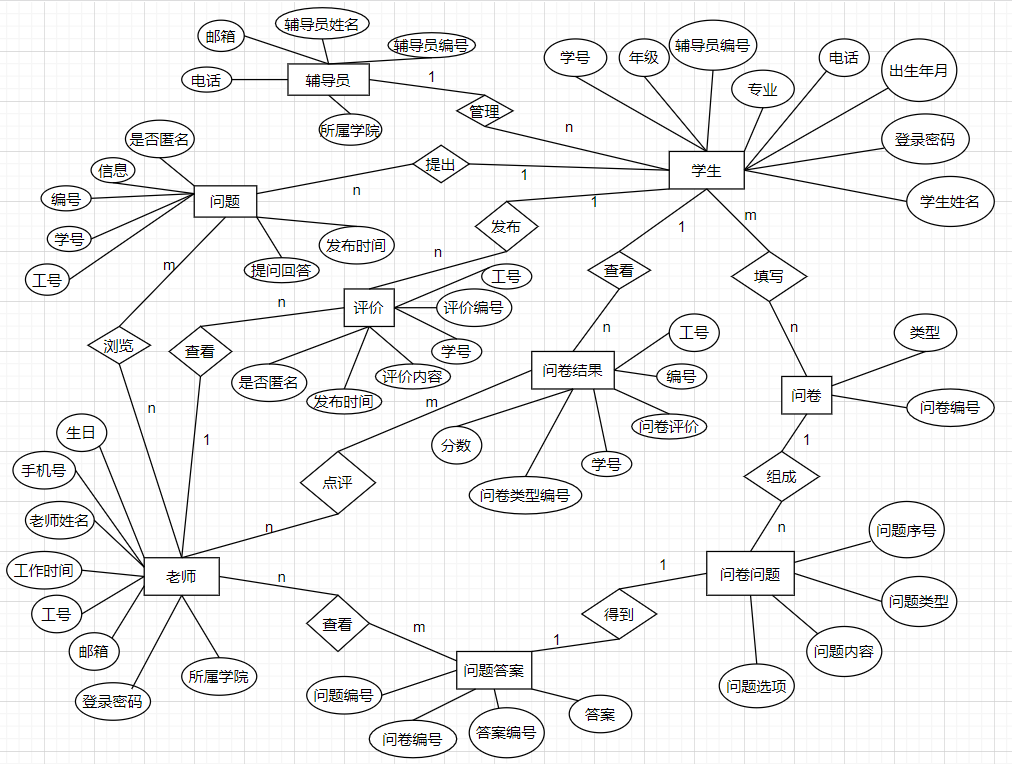


图（3-2-2）



图（3-2-3）

数据库设计对于一个给定的应用环境，构造最优的数据库模式，建立数据库及其应用系统，使之能够有效地存储数据，满足各种用户的应用需求，这里我们通过E-R图（3-2-4）展示该数据库结构。



图（3-2-4）

数据库中基本表是数据库设计的基本要求，其能把每一个实体表示出来，以下便是心理健康数据库相关基本表：

Studentinfo(学生个人信息表格)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 学号  SNO(key) | 学生姓名SName | 出生年月Birth | 联系方式电话SPNo | 所在院系专业Major | 年级Grade | 辅导员编号SInstructor (外码) | 登录密码Spwd |
| Varchar(20) | Varchar(10) | Varchar(10) | Varchar(11) | Varchar(20) | Varchar(4) | Varchar(20) | Varchar(10) |

辅导员的信息表instructorinfo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 辅导员编号  SInstructor(key) | 辅导员姓名  IName | 所属学院  Birth | 电话  IPNo | 邮箱  Email |
| Varchar(20) | Varchar(10) | Varchar(20) | Varchar(11) | Varchar(20) |

问卷调查的形式qtype

|  |  |
| --- | --- |
| 问卷类型名Type | 问卷类型的编号QNo(key) |
| 情感类 Varchar(15) | Int |
| 学业类 varchar(15) | Int |
| 生活类 varchar(15) | Int |
| 未来发展 varchar(15) | int |

问卷的表，导入，几个类型分别有其基本表，当用户选择了那种类型之后，我们再在其中某个表中随记生成部分问题。通过当前学生选择--->问卷类型--->生成问卷，

每个类型的基本表：应该用视图生成一个表，这个视图要关联当前学生的学号，或者在基本表中加入学号作为外码

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 问题序号  Num(key) | 问题类型Type(外码) | 问题内容  Question | 问题的选项  AnswerA | 问题的选项  AnswerB | 问题的选项  AnswerC | 问题的选项  AnswerD |
| Varchar(5) | int | text | Varchar(20) | Varchar(20) | Varchar(20) | Varchar(20) |

学生的提问表sask

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编号  SANo  (key) | 学生学号  SNo（外码） | 老师工号/选择老师  TNo(外码) | 信息  SAInfo | 是否匿名  Ano | 提问的回答  TAnsewr | 发布时间，时间戳  Time |
| int | Varchar(20) | Varchar(20) | text | Varchar(1) | text | Varchar(30) |

学生对老师的评价表tevalute

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 评价表编号  TENo(key) | 学生学号  SNo(外码) | 评价内容  Evalute | 发布的时间  TETime | 是否匿名  TAno | 老师工号  TNo(外码) |
| int | Varchar(20) | text | Varchar(30) | Varchar(1) | Varchar(20) |

老师的信息表格teacherinfo

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 工号  TNo(key) | 姓名  TName | 生日  TBirth | 邮箱  TEmail | 手机号  TPhone | 所属学院  TCollege | 工作时间  TWork | 登录密码  TPwd |
| Varchar(20) | Varchar(10) | Varchar(20) | Varchar(20) | Varchar(11) | Varchar(20) | text | Varchar(10) |

问卷答案表格questions

|  |  |  |  |
| --- | --- | --- | --- |
| 答案的编号  QuestionNo(key) | 所答问卷的编号  QRNo(外码) | 问题编号QNum  (外码) | 答案QAnswer |
| 自动生成int | Varchar(40) | int | Varchar(40)默认单选 |

问卷结果的表qresult

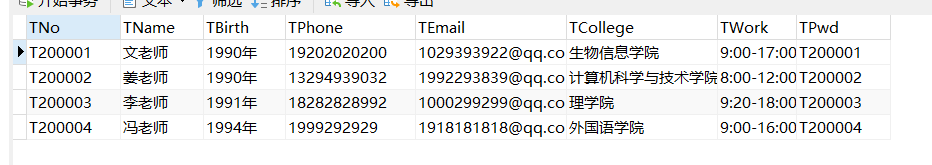
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号  QRNo(key) | 问卷类型的编号QNo(外码) | 老师对学生的心理问卷的评估QREva | 分数  QGrade | 学生学号  SNo(外码) | 老师编号  TNo(外码) |
| Varchar(40) | int | text | int | Varchar(20) | Varchar(20) |

3.3后端设计

这里我们基于jdbc将数据库与Java进行连接在Conn类中首先通过Class.forName("com.mysql.cj.jdbc.Driver");加载数据库驱动，加载成功后输入数据库用户名，密码，调用con = DriverManager.getConnection("jdbc:mysql://localhost:3306/test?useUnicode=true&characterEncoding=gbk" , user1, passord1);判断用户名，密码是否和数据库中用户名，密码相符，若相符则数据库连接成功，这里localhost为本地，3306为接口，test为数据库。成功后通过后端运行登录界面，当前端点击学生界面时及操控Sign类选择学生端，接着后端操控操控StudentE类进行用户密码识别，若正确则跳转到学生操作界面，后端StudentInfo类中会显示学生个人信息，当前端学生进行电话修改时，这里将会相应进行修改，更新后反馈与前端，对应于QuestionnaireType类，操控学生选择的问卷类型，Questionnaire类，操控学生问卷结果，StudentAsk类，操控学生提问，自此完成前后端学生交互。老师端同学生一样后端操控TeacherE进行用户密码识别，相关功能实现为TeacherInfo类实现老师个人信息更改，SInstructor类显示辅导员信息，TeacherLook类实现老师对于学生问题查看，TeacherSearch类实现老师查看学生对自己的评价。管理员端则在登录管理员后ManageS实现管理员对于学生的增删改查，ManageT实现管理员对于老师的增删改查。详细见附录。

1. 数据创建







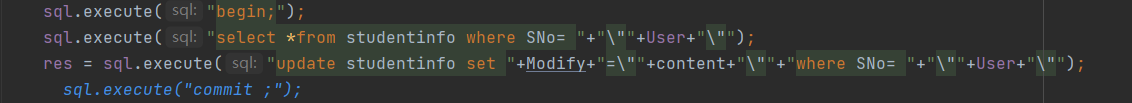
运行示例见视频。

五、数据库优化

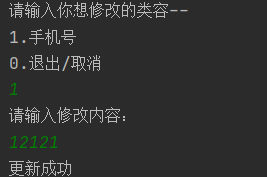
由于学生端和老师端需要对部分基本信息进行修改的操作，于此同时，管理员端也能对学生老师的信息进行修改和删除操作，那么我们就不得不考虑数据库的并发操作。如果当学生和管理员两种角色都在对studentinfo这个表进行修改的话，或者管理员在进行删除操作时就可能会出现脏读等操作，使得数据出现错误。

对此，我们对studentinfo ，和teacherinfo进行加排他锁，主要是加的行锁，实现本用户在读取数据，并进行修改等操作时，其他用户不能再对此行数据进行加锁操作，只能读取数据，也不能进行修改等操作。只有在排它锁得到释放之后，其余用户才能对其进行加锁操作。代码如下：

sql.execute("begin ");  
                sql.execute("select \*from teacherinfo where TNo="+"\""+TNo+"\""+" for update ");  
                res = sql.execute("update teacherinfo set "+Modify+"="+"\""+content+"\""+"where TNo="+"\""+TNo+"\";");  
                sql.execute("commit;");



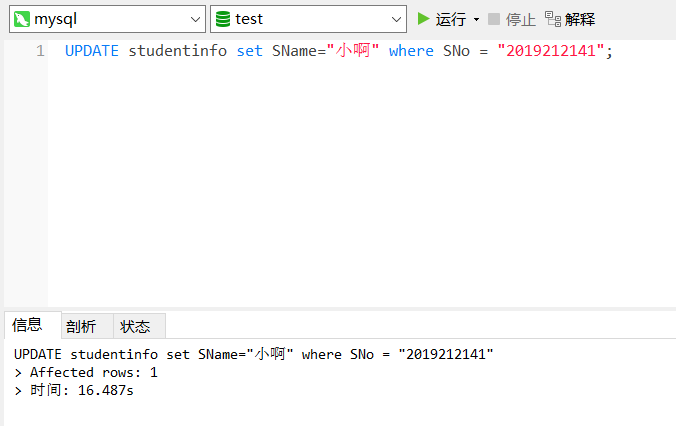
图（5-1）



图（5-2）



图（5-3）



图（5-4）

由图（5-1），图（5-2），图（5-3），图（5-4），当我们在程序中将commit操作取出后，再执行对studentinfo表相应数据加锁的操作之后，本用户就会一直持有该锁，那么我们在Navicat中建立查询，对其这条数据进行修改操作，我们会发现，其一直阻塞在这里，只有当我们结束程序运行之后，该锁才释放，最后这条修改语句才执行成功。