

# 浙江大学



题    目： 高血压社区管理系统设计报告

姓    名： 陈  浩  亮

学    号： 3150100908

授课教师： 黄正行  助教：陆  易

专    业： 测控技术与仪器

年    级： 大  二

## 目录

第一章 概述 .....	3
1.1 高血压社区管理系统的设计背景 .....	3
1.2 高血压社区管理系统的意义 .....	3
1.3 高血压社区管理系统的实现工具 .....	3
1.4 高血压社区管理系统的建立过程 .....	4
1.5 高血压社区管理系统的目标功能 .....	4
1.6 高血压社区管理系统的运行平台 .....	4
第二章 需求分析 .....	5
2.1 医生需求 .....	5
2.2 患者需求 .....	5
2.3 管理员需求 .....	6
第三章 系统概要设计 .....	7
3.1 系统总体设计 .....	7
3.2 系统界面设计 .....	8
3.3 数据库设计 .....	9
3.4 异常情况处理 .....	10
第四章 系统详细设计 .....	11
4.1 系统界面详细设计 .....	11
4.2 数据库详细设计 .....	22
4.3 部分函数实现 .....	24
第五章 用户手册 .....	28
5.1 医生使用手册 .....	28
5.2 患者使用手册 .....	32
第六章 回顾总结 .....	37
6.1 搭建系统过程中的问题 .....	37
6.2 该系统尚存在的不足 .....	37
6.3 回顾总结 .....	37

# 第一章 概述

## 1.1 高血压社区管理系统的设计背景

世界卫生组织已经确认高血压为导致心血管病死亡率的主要原因。世界高血压联盟是 85 个国家高血压社团和联盟的联盟组织，该组织认为，世界范围内 50% 以上的高血压患者不了解自身的情况。<sup>1</sup>而高血压依然是严重危害人类健康的常见疾病，具有发病率、致残率、死亡率“三高”及知晓率、治疗率和控制率“三低”的特点。它与脑卒中、冠心病、慢性心功能衰竭、终末期肾病等疾病密切相关。高血压发病隐匿，损害靶器官系统范围广，已成为严重影响人民群众健康水平和生活质量的一大疾病，并且给社会增添了沉重的经济负担。因此，必须大力开展高血压的人群防治，加强全民健康教育，严格对已发现的高血压患者进行科学的网络化管理，以达到降低“三高”率，提高“三低”率的目标。<sup>2</sup>

## 1.2 高血压社区管理系统的意义

高血压社区管理系统为高血压患者与社区医生搭建了沟通的桥梁。患者可以通过该系统与医生进行交流，及时了解自身病情，并获取医疗建议。医生则可以随时查看自己患者的情况并给出医疗建议。高血压社区管理系统的存在，将有效降低发病率、致残率、死亡率的“三高”率，提高知晓率、治疗率和控制率的“三低”率。为高血压疾病的预防、治疗与管理提供了有效渠道。

## 1.3 高血压社区管理系统的实现工具

界面搭建：Eclipse (Neon.2 Release 4.6.2) + Window Builder (for Neon)

数据库表的建立：Power Designer (16.5) + SQL Server Management Studio (2012)

客户端（软件）编写语言：Java

数据库管理工具：SQL Server (2012)

<sup>1</sup> 维基百科：高血压(<https://zh.wikipedia.org/wiki/%E9%AB%98%E8%A1%80%E5%A3%93#.E8.AE.A4.E7.9F.A5>)

<sup>2</sup> 百度百科：高血压病管理信息系统

(<http://baike.baidu.com/item/%E9%AB%98%E8%A1%80%E5%A3%93#.E8.AE.A4.E7.9F.A5>)

## 1.4 高血压社区管理系统的建立过程

- 通过调查进行需求分析，了解医生与患者的不同需求
- 使用 Eclipse + Window Builder 搭建系统的基本页面
- 建立医生患者的 E-R 模型，确定对应的关系
- 使用 Power Designer 生成概念数据模型和物理数据模型，进而生成数据库脚本
- 在 SQL Server 中运行数据库脚本，对于主码、外码等进行相关的设置，如设置主码 id 自增
- 通过 jdbc 实现客户端与数据库的连接，并实现对数据库的增删改操作。
- 对系统进行测试调试，修复错误与提升性能，优化用户体验。
- 编写设计报告

## 1.5 高血压社区管理系统的目标功能

1. 医生与患者的连接功能
  - a) 患者可以关联自己的主治医生
  - b) 患者可以更改关联
  - c) 医生可以取消关联患者
2. 医生与患者的资料查看
  - a) 患者可以了解医生的联系方式、工作时间等信息
  - b) 医生可以了解患者的身体参数、血压值等信息
3. 医生与患者的信息交互
  - a) 医生与患者可以彼此发送消息
  - b) 医生可以为患者提供医疗建议
  - c) 患者可以预约医生进行当面诊断
  - d) 患者可以对主治医生进行评分
4. 医生与患者的隐私保护
  - a) 除患者自己与患者的主治医生外，其他人无法查看患者的资料
  - b) 除医生自己与医生主治的患者外，其他人无法获得医生的详细资料
  - c) 若有需要医生患者可以注销账户，将数据库中与自己有关的数据删除

## 1.6 高血压社区管理系统的运行平台

目前高血压社区管理系统仅支持已装有 JRE (Java Running Environment) 的 Windows/Mac OS 电脑

## 第二章 需求分析

### 2.1 医生需求

1. 了解自己主治的患者的基本信息
  - a) 身体基本参数，如身高体重等
  - b) 患者的家族病史及自身疾病史
  - c) 有无心血管疾病
  - d) 日常饮食、锻炼情况
  - e) 是否曾接受治疗及相关治疗情况
2. 与患者进行交流
  - a) 医疗建议
  - b) 治疗方案
  - c) 消息交互
3. 查看预约信息
  - a) 查看患者的预约情况
  - b) 对患者的预约作出回应
4. 隐私保护
  - a) 除自己的患者外，其他人无法查看自己的详细联系方式
  - b) 有注销账号权限

### 2.2 患者需求

1. 选择自己的主治医生
  - a) 通过医生的姓名查找对应的医生
  - b) 关联或取消关联主治医生
  - c) 查看自己医生的联系方式、工作时间等信息
2. 与医生进行交互
  - a) 获取医生的建议及治疗方案
  - b) 对医生进行评价反馈
  - c) 预约医生进行当面诊断
3. 修改自己的资料
  - a) 更新自己的基本信息
  - b) 更新自己的血压，包括血压记录时间
4. 隐私保护
  - a) 除自己的主治医生外，其他人无法查看自己的信息
  - b) 有注销账号权限

## 2.3 管理员需求

1. 对数据库进行相关操作
  - a) 对数据库的各表进行增删改查操作
  - b) 对于冗余、无效数据进行删除以减小数据库存储空间
  - c) 定期对数据库进行备份操作
  - d) 避免数据库的数据泄露
2. 对高血压社区管理系统进行维护
  - a) 接收用户在使用高血压社区管理系统中提供的建议反馈
  - b) 发布更新版本供用户下载更新

## 第三章 系统概要设计

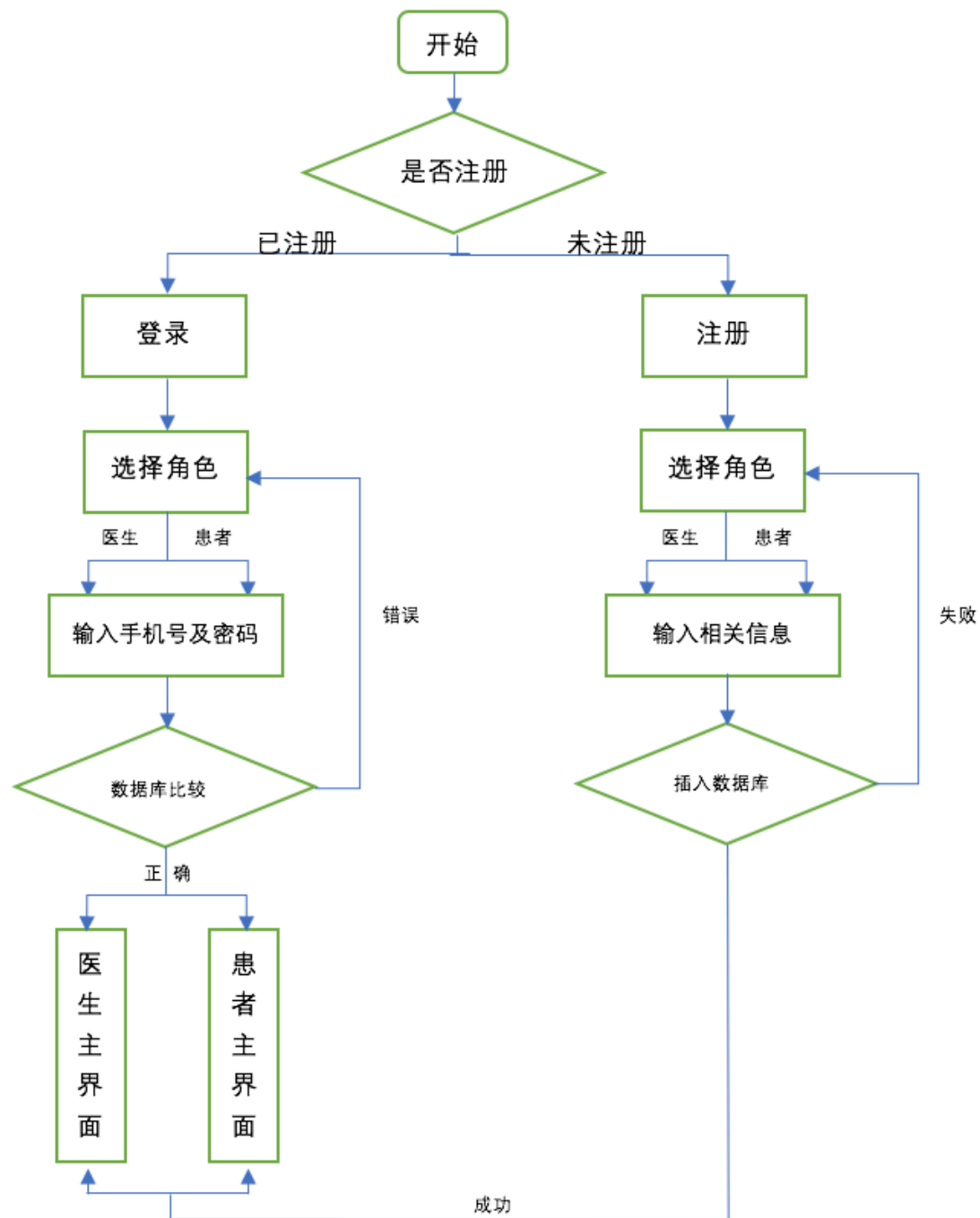
### 3.1 系统总体设计

#### 3.1.0 运行环境

操作系统 Windows/Mac OS

安装 JRE(Java Running Environment)

#### 3.1.1 系统的基本操作流程：



## 3.2 系统界面设计

### 3.2.0 高血压社区管理系统的初始页面：

用户选择自己的角色并进行登录或注册操作。(如图 3-1 所示)

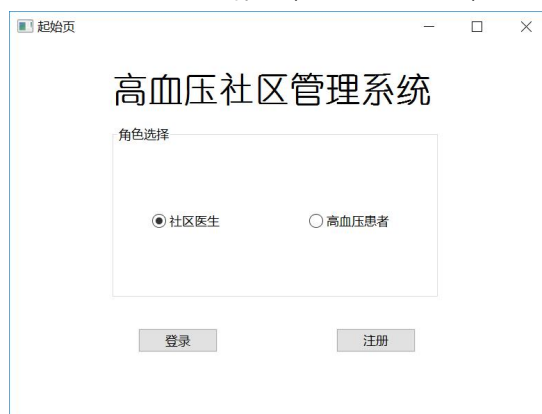


图 3-1 高血压社区管理系统起始页

### 3.2.1 医生的主界面设计

通过菜单栏划分成患者信息、日程安排、设置、关于等四个板块。(如图 3-2 所示)



图 3-2 医生登录后主页面

### 3.1.3 患者的主界面设计

通过菜单栏划分成查看医生、设置、关于等三个板块。(如图 3-3 所示)



图 3-3 患者登录后主页面



### 3.3 数据库设计

#### 3.3.1 概念结构设计

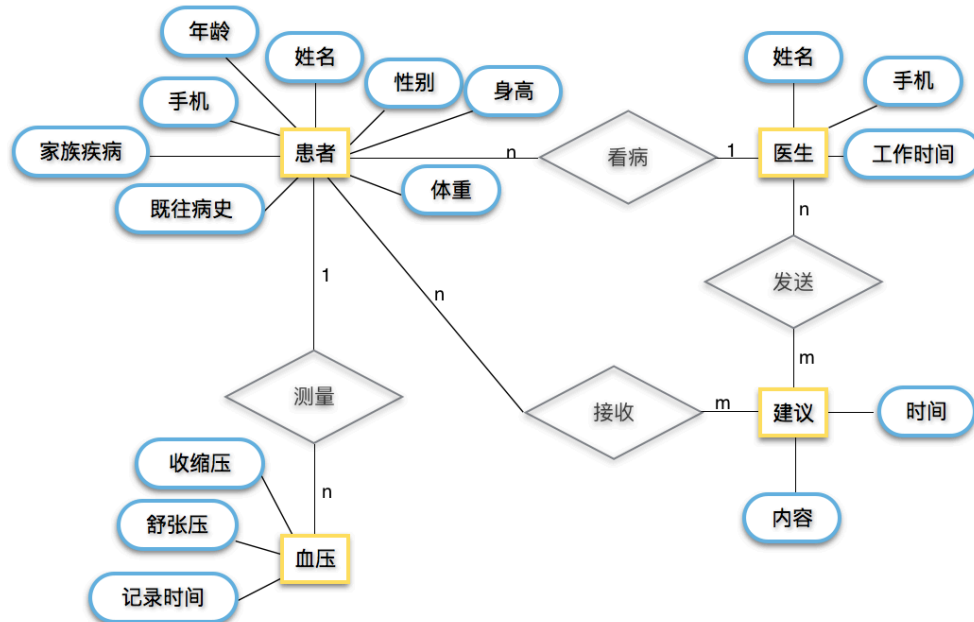


图 3-4 高血压社区管理系统基本 E-R 图

根据高血压社区管理系统明确实体对象包括患者、医生、血压、建议等，并确定各实体的属性，建立高血压社区管理系统的基本 E-R 图（如图 3-4 所示）。

（该基本 E-R 图未包括固定电话等次要属性及评分等次要实体）

#### 3.3.2 逻辑结构设计

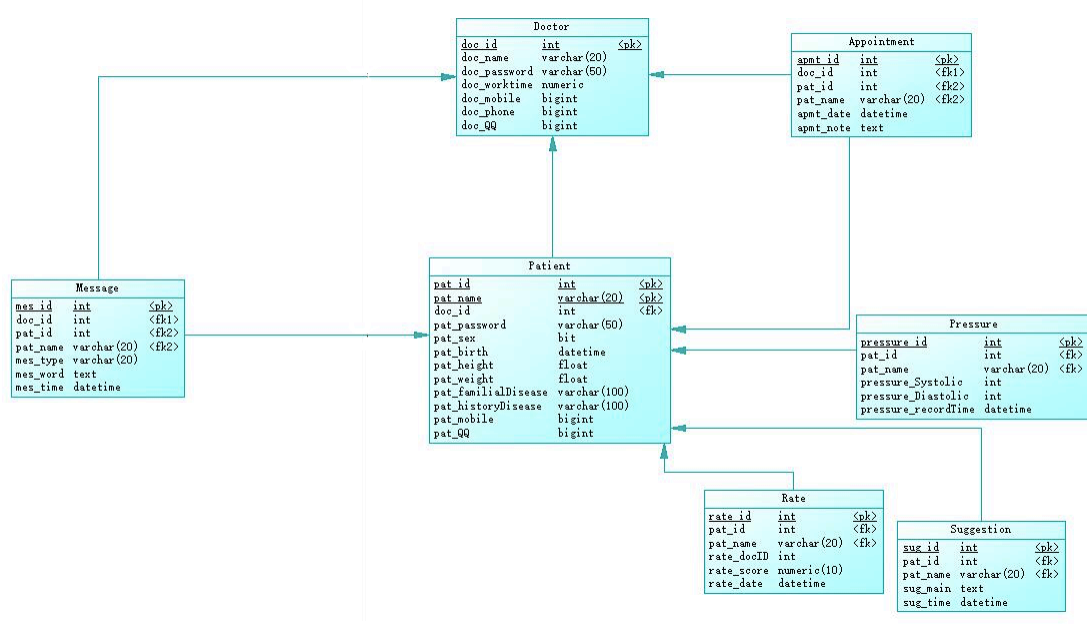


图 3-5 高血压社区管理系统数据模型

将 E-R 图向关系模型转换，得到物理数据模型（如图 3-5 所示）。

### 3.3.3 物理结构设计

根据数据模型建立的数据库中有 7 张表：

医生 (Doctor)：医生 ID、姓名、手机号、密码、工作时间、电话、QQ

患者 (Patient)：患者 ID、姓名、医生 ID、手机号、密码、性别、年龄、身高、体重、既往疾病、家族病史、QQ

血压 (Pressure)：血压 ID、患者 ID、收缩压、舒张压、记录时间

建议 (Suggestion)：建议 ID、患者 ID、建议内容、建议时间

预约 (Appointment)：预约 ID、医生 ID、患者 ID、预约时间、预约备注

消息 (Message)：消息 ID、医生 ID、患者 ID、消息内容、消息时间、消息是否由医生发送

评分 (Rate)：评分 ID、患者 ID、医生 ID、分值、评价日期

## 3.4 异常情况处理

针对一些特殊情况事先作出预防措施，并弹出窗口提示：

注册时对于手机号的填写进行判断，防止无效的手机号注册；

注册时的密码与确认密码需一致，否则无法注册成功；

在填写相关数据时对数据类型进行检查；

患者未关联医生时无法进入自己的主治医生页面；

医生无法访问非关联患者的页面

.....

登录失败，弹窗提示失败原因：

手机号未注册，数据库中未保存对应手机号

密码错误，与数据库中保存的密码不一致

数据库错误：

及时进行数据库备份等操作

## 第四章 系统详细设计

### 4.1 系统界面详细设计

#### 4.1.1 系统主页面（启动系统时显示的初识界面）

用户选择自己对应的身份，并跳转到对应的登录注册页面。（如图 4-1 所示）



图 4-1 高血压社区管理系统起始页

#### 4.1.2 登录注册页面

医生注册页面：输入自己的姓名、手机号、密码，选择工作时间。（如图 4-2 所示）



图 4-2 医生注册页面

医生登录页面：医生输入已注册的手机号与密码，进行登录操作。（如图 4-3 所示）



图 4-3 医生登录页面

患者注册页面：依次输入姓名、手机号、密码等基础信息，并输入性别、年龄、身高、体重、收缩压、舒张压、家族病史、既往病史、备注等详细信息。(如图 4-4、图 4-5 所示)

注册-患者

姓名:

手机号:

密码:

确认密码:

图 4-4 患者注册页面 1-基础信息

完善个人信息

性 别:

年 龄:

身 高:  m

体 重:  kg

血 压:  (收缩压/mmHg)  (舒张压/mmHg)

家族病史:

既往病史:

备 注:

图 4-5 患者注册页面 2-详细信息

患者登录页面：患者输入已注册的手机号及密码，进行登录操作。(如图 4-6 所示)

登录-患者

手机号:

密码:

图 4-6 患者登录页面

### 4.1.3 医生相关页面

#### 4.1.3.0 医生登陆后主界面

医生登录后，进入医生端的主界面。屏幕中央显示欢迎登录，及医生的姓名。表示每一名医生登录后的页面都唯一。(如图 4-7 所示)



图 4-7 医生登录后主界面

其中菜单栏包括患者信息、日程安排、设置、关于四个板块。板块的子版块划分如图 4-8~11 所示。

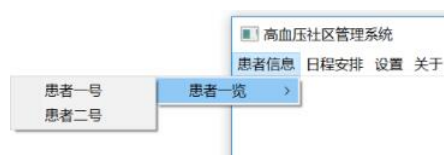


图 4-8 患者信息板块

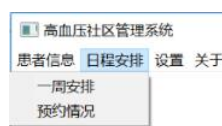


图 4-9 日程安排板块



图 4-10 设置板块

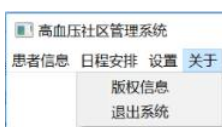


图 4-11 关于板块

#### 4.1.3.1 医生查看患者界面

点击患者信息的患者一览可以查看关联自己的所有病人的姓名，点击进入后可以查看该患者的详细信息，或者取消关联该患者。(如图 4-12 所示)



图 4-12 医生查看患者详细信息界面

基本信息一栏中包括年龄、性别、身高、体重、家族疾病、既往病史等资料。(如图 4-13 所示)



图 4-13 医生查看患者基本信息

血压变化一栏中可以查看患者的血压变化 (如图 4-14 所示)。单击按钮<>可以进行翻页操作。血压为倒序查看，最先显示的是最新一次血压。



图 4-14 医生查看患者血压变化

消息记录一栏中可以查看与患者的消息记录。默认显示最新的消息。(如图 4-15 所示)

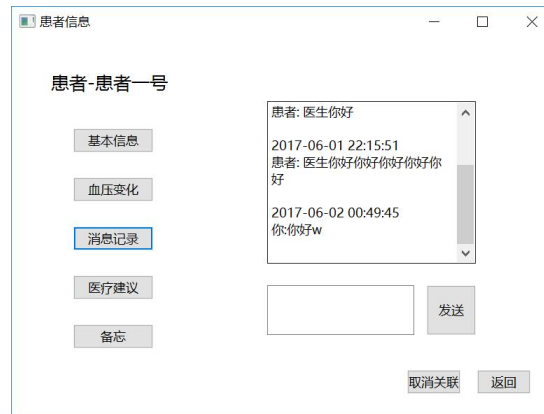


图 4-15 医生查看与患者消息记录

医疗建议一栏中可以给患者发送相关医疗建议。(如图 4-16 所示)



图 4-16 医生发送医疗建议

备忘一栏可以记录一些事项。(如图 4-17 所示)



图 4-17 医生记录备忘内容

#### 4.1.3.2 医生其他界面

点击设置的个人信息，可以对自己的姓名、电话、QQ、工作时间等进行修改操作。  
(如图 4-18 所示)

图 4-18 医生修改个人信息

点击设置的修改密码，可以修改自己的密码。(如图 4-19 所示)

图 4-19 医生修改密码

点击关于的版权信息，可查看该系统的作者信息。(如图 4-20 所示)

图 4-20 版权信息



#### 4.1.4 患者相关页面

##### 4.1.4.0 患者登录后主界面

患者登录后，进入患者端的主界面。屏幕中央显示欢迎登录，及患者的姓名。表示每一名患者登录后的页面都唯一。(如图 4-21 所示)



图 4-21 患者登录后主界面

其中菜单栏包括查看医生、设置、关于三个板块。板块的子版块划分如图 4-22~24 所示。



图 4-22 查看医生板块

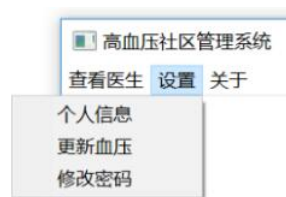


图 4-23 设置板块

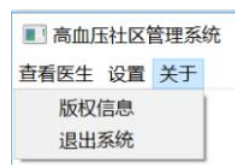


图 4-24 关于板块

#### 4.1.4.1 患者主治医生界面

点击查看医生的主治医生，进入自己的主治医生界面（如图 4-25 所示），若未关联主治医生则无法访问该页面。



图 4-25 患者查看主治医生界面

点击联系方式一览，可以查看医生的联系方式及工作时间（如图 4-26 所示）。

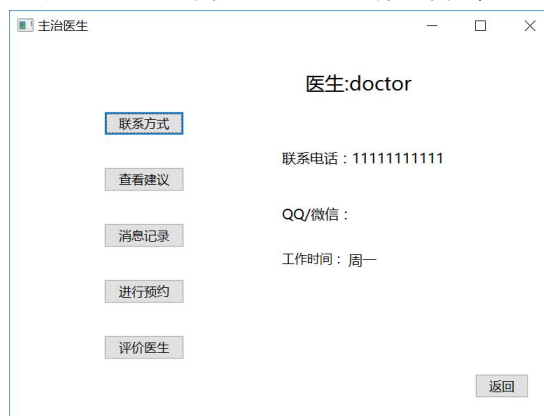


图 4-26 患者查看联系方式

点击查看建议，可以获得主治医生给的医疗建议（如图 4-27 所示）。



图 4-27 患者查看建议

点击消息记录，可以查看自己与主治医生的聊天记录（如图 4-28 所示）。



图 4-28 患者查看消息记录

点击进行预约，可以对主治医生进行预约（如图 4-29 所示）。



图 4-29 患者进行预约操作

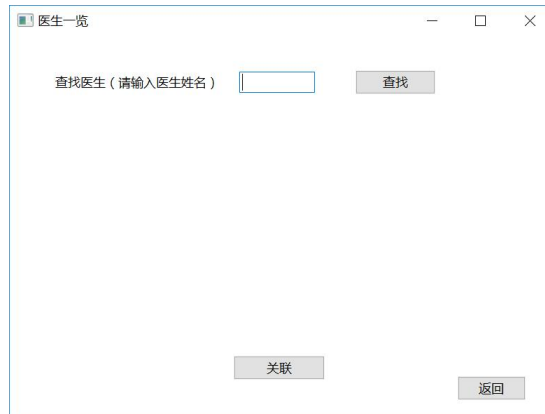
点击评价医生，可以对主治医生进行评分操作（如图 4-30 所示）。



图 4-30 患者对医生进行评分

#### 4.1.4.2 患者其他界面

点击查看医生的医生一览菜单，可以进行查找医生的相关操作（如图 4-31 所示）。

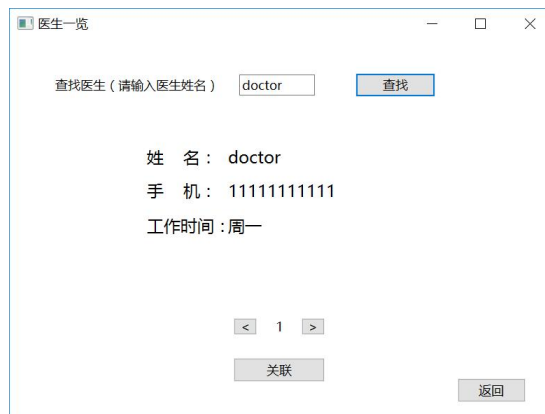


医生一览

查找医生 (请输入医生姓名)

图 4-31 医生一览界面

输入医生姓名查找后若数据库中存在相关医生，则显示对应医生的信息（如图 4-32 所示），支持模糊查找、查找结果多项显示。患者可以在该页面进行关联医生操作。



医生一览

查找医生 (请输入医生姓名)

姓 名: doctor  
手 机: 11111111111  
工作时间: 周一

< 1 >

图 4-32 查找医生结果

点击设置的个人信息，可以对自己的姓名、年龄等进行修改操作（如图 4-33 所示）。



个人信息

姓名  年龄

身高  m 体重  kg

家族疾病  既往病史

QQ  手机号

图 4-33 患者编辑个人信息

点击设置的更新血压，患者可以更新自己的血压值（如图 4-34 所示）。



图 4-34 患者更新血压

点击设置的修改密码，患者可以修改自己的密码（如图 4-35 所示）。



图 4-35 患者修改密码

点击关于的版权信息，可查看该系统的作者信息。（如图 4-36 所示）

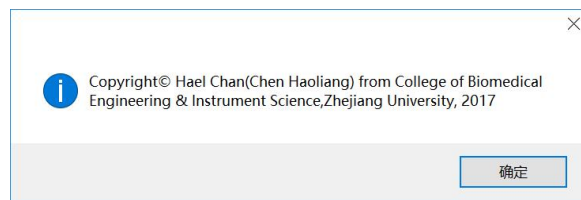


图 4-36 版权信息

## 4.2 数据库详细设计

Appointment（预约）表：

列名	字段名称	数据类型	允 许 Null 值	主键	外键
预约标识	apmt_id	int		√	
医生标识	doc_id	int			Doctor 表 doc_id
患者标识	pat_id	int			Patient 表 pat_id
预约日期	apmt_date	datetime			
预约备注	apmt_note	nvarchar(50)	√		

Doctor（医生）表：

列名	字段名称	数据类型	允 许 Null 值	主键	外键
医生标识	doc_id	int		√	
医生姓名	doc_name	varchar(20)			
医生密码	doc_password	varchar(50)			
医生工作日	doc_workday	varchar(70)	√		
医生手机	doc_mobile	varchar(20)			
医生电话	doc_phone	varchar(20)	√		
医生 QQ	doc_QQ	varchar(20)	√		

Message（消息）表：

列名	字段名称	数据类型	允 许 Null 值	主键	外键
消息标识	mes_id	int		√	
医生标识	doc_id	int			Doctor 表 doc_id
患者标识	pat_id	int			Patient 表 pat_id
消息内容	mes_text	varchar(100)	√		
消息时间	mes_time	varchar(50)	√		
是否医生发送	mes_sentByDoc	int	√		

Rate（评分）表：

列名	字段名称	数据类型	允 许 Null 值	主键	外键
评分标识	rate_id	int		√	
医生标识	doc_id	int			Doctor 表 doc_id
患者标识	pat_id	int			Patient 表 pat_id
评分分值	rate_score	numeric(10,0)			
评分日期	rate_date	datetime	√		

Patient（患者）表：

列名	字段名称	数据类型	允 许 Null 值	主键	外键
患者标识	pat_id	int		√	
患者姓名	pat_name	varchar(20)			
医生标识	doc_id	int	√		Doctor 表 doc_id
患者密码	pat_password	varchar(50)			
患者性别	pat_sex	varchar(50)	√		
患者年龄	pat_age	varchar(50)	√		
患者身高	pat_height	varchar(50)	√		
患者体重	pat_weight	varchar(50)	√		
家族疾病	pat_familialDisease	varchar(100)	√		
既往病史	pat_historyDisease	varchar(100)	√		
患者手机	pat_mobile	varchar(50)			
患者 QQ	pat_qq	varchar(50)	√		

Pressure（血压）表：

列名	字段名称	数据类型	允 许 Null 值	主键	外键
血压标识	pressure_id	int		√	
患者标识	pat_id	int			Patient 表 pat_id
收缩压	pressure_Systolic	varchar(10)			
舒张压	pressure_Diastolic	varchar(10)			
记录时间	pressure_recordTime	varchar(50)	√		

Suggestion（建议）表：

列名	字段名称	数据类型	允 许 Null 值	主键	外键
建议标识	sug_id	int		√	
患者标识	pat_id	int			Patient 表 pat_id
建议内容	sug_main	text			
建议时间	sug_time	datetime	√		

注：所有设置为主键的标识均设置了自增。

标识规范：是；

（是规范）：是

标识增量：1

标识种子：1

## 4.3 部分函数实现

### 4.2.1：窗体居中显示

实现功能：默认状态下，该高血压管理系统始终位于屏幕中央。

算法实现：使用 Toolkit 工具包中的 `getScreenSize()` 函数获得屏幕的宽与高。Java 中 Object 的坐标默认为左上角顶点坐标，故将窗体的坐标其设为（屏幕宽-窗体宽）/2，（屏幕高-窗体高）/2，即实现窗体居中显示。

代码实现：

```
/*
 * Set the window to the center of the screen
 */
int screenHeight = Toolkit.getDefaultToolkit().getScreenSize().height;
int screenWidth = Toolkit.getDefaultToolkit().getScreenSize().width;
int x = (screenWidth - 800) / 2;
int y = (screenHeight - 600) / 2;
shell.setText("\u8D77\u59CB\u9875");
shell.setLocation(x, y);
```

### 4.2.2：文本框相关设置

实现功能：密码框中用“\*”隐藏输入的密码

代码实现：`text.setEchoChar('*');`

实现功能：文本框不可更改：部分文本框应设为不可编辑（not editable）。

代码实现：`text.setEditable(false);`

实现功能：文本框的内容分多行显示，且显示滚动条可以滑动。

算法实现：定义 text 时包括 `SWT.BORDER | SWT.MULTI | SWT.WRAP | SWT.V_SCROLL` 等属性。

实现功能：文本框默认显示最下方内容。（例如消息文本框中消息是正序排序，最下方的消息为最新发送，默认应显示最新消息）

代码实现：`text.setTopIndex(Integer.MAX_VALUE);`

现以 `textMessage` 的创建为例，涵盖上述相关功能。

```
textMessage = new Text(shell, SWT.BORDER | SWT.MULTI | SWT.WRAP | SWT.V_SCROLL);
textMessage.setText("");
textMessage.setEditable(false);
textMessage.setBackground(SWTResourceManager.getColor(SWT.COLOR_TRANSPARENT));
textMessage.setBounds(365, 91, 298, 232);
textMessage.setVisible(false);

textSendMessage = new Text(shell, SWT.BORDER | SWT.MULTI | SWT.WRAP); //reference:http://bbs.csdn.net/topics/260045661
textSendMessage.setBounds(365, 354, 210, 71);
textSendMessage.setVisible(false);

int messageNum = messageDao.countMessages(pat_id, patient.getDoc_id());

for(int i = 0; i < messageNum; ++i) {
    Message message = messageDao.findMessage(pat_id, patient.getDoc_id(), i);
    //textMessage.setText(textMessage.getText() + message.getMes_time() + " ");
    textMessage.append("\n" + message.getMes_time() + " ");

    if(message.getMes_sentByDoc() == 0) {
        //textMessage.setText(textMessage.getText() + "\n你: " + message.getMes_text() + "\n\n");
        textMessage.append("\n患者: " + message.getMes_text() + "\n");
    }
    else {
        //textMessage.setText(textMessage.getText() + "\n医生: " + message.getMes_text() + "\n\n");
        textMessage.append("\n你: " + message.getMes_text() + "\n");
    }

    //textMessage.setText(textMessage.getText() + message.getMes_text() + "\n\n");
}

textMessage.setTopIndex(Integer.MAX_VALUE); //http://blog.csdn.net/java2000_net/article/details/3905307
```



#### 4.2.3：消息发送者区分

实现功能：在消息记录中能直观区分消息的发送方与接收方。如图 4-37 的患者端消息记录中，患者能明显判断哪条消息是自己发送的，哪一条是医生发送的。医生端同理。



图 4-37 患者端消息记录

算法实现：因为医生和患者都能对 Message 表进行操作，故需设置一个标志位以表示发送方的属性。Message 实体中新增一项 mes\_sentByDoc。若医生发送消息则该值为 1，若患者发送消息则该值为 0。显示消息时即可根据该值直观显示消息发送者。

代码实现：略。

#### 4.2.4：医生查看自己的患者是通过菜单栏全部显示，如图 4-38 所示。

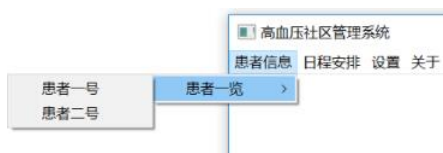


图 4-38 显示自己的所有患者

实现功能：医生在主页面的查看患者->患者一览菜单栏中可以查看自己所有的患者。

算法实现：通过获得数据库中关联该医生的患者数量，使用 for 循环逐个添加菜单栏并添加相关信息。

代码实现：

```
int patientNum = doctorDao.countYourPatients(doctor.getDoctor_id());

if (patientNum == 0) {
    MenuItem menuItem = new MenuItem(menu_Name, SWT.NONE);
    menuItem.setText("目前还没有患者关联您");
} else {
    for (int i = 0; i < patientNum; ++i) {
        MenuItem menuItem = new MenuItem(menu_Name, SWT.NONE);
        Patient patient = doctorDao.findYourPatients(doctor.getDoctor_id(), i);
        menuItem.addSelectionListener(new SelectionAdapter() {

            public void widgetSelected(SelectionEvent e) {
                display.close();
                PatInfo patInfo = new PatInfo(patient.getPat_id());
                patInfo.open();
            }

        });
        menuItem.setText(patient.getPat_name());
    }
}

public Patient findYourPatients(String doc_id, int index) {
    getJdbcTemplate();
    try {
        String sql = "select * from dbo.Patient where doc_id=?";
        List query = jdbcTemplate.query(sql, new Object[] {doc_id}, new RowMapper() {
            @Override
            public Object mapRow(ResultSet set, int rowNum)
                throws SQLException {
                Patient patient = new Patient();
                patient.setPat_id(set.getString("pat_id"));
                patient.setPat_name(set.getString("pat_name"));
                return patient;
            }
        });
        if (query.size() == 0 || index > query.size()) {
            return null;
        }
        return (Patient) query.get(index);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
```

4.2.5 在患者查看主治医生或者医生查看患者的信息包括了大量元素。

实现功能：同一界面中大量元素显示或隐藏。

算法实现：使用 setVisible()函数，在点击某一按钮时使其他元素不可见。

代码实现：

```
Button btnAdvice = new Button(shell, SWT.NONE);
btnAdvice.addSelectionListener(new SelectionAdapter() {
    @Override
    public void widgetSelected(SelectionEvent e) {
        lblPhone.setVisible(false);
        lblTencent.setVisible(false);
        lblWorkTime.setVisible(false);
        lbl.setVisible(false);
        //lblAddress.setVisible(false);
        txtAdvice.setVisible(true);
        txtMessage.setVisible(false);
        txtSendMessage.setVisible(false);
        txtApmtNote.setVisible(false);
        dateTime.setVisible(false);
        btnSend.setVisible(false);
        btnSubmit.setVisible(false);
        scaleScore.setVisible(false);
        lblPoint.setVisible(false);
        btnRate.setVisible(false);
    }
});
btnAdvice.setBounds(135, 180, 114, 34);
btnAdvice.setText("\u67E5\u770B\u5EFA\u8BAE");
```

4.2.6 患者在医生一览表页面查找医生时实现模糊查找

实现功能：患者在不知道医生的全名的情况下也能实现查找。

算法实现：使用数据库理论课上所学的字符匹配，通过 LIKE %等关键字实现

代码实现：

```
public Doctor findByNameAndIndex(String doc_name, int index){
    getJdbcTemplate();
    try{
        String sql="select * from dbo.Doctor where doc_name like ?";
        List query = jdbcTemplate.query(sql, new Object[]{"%" + doc_name + "%"}, new RowMapper() {
            @Override
            public Object mapRow(ResultSet set, int rowNum)
                throws SQLException {
                Doctor doctor=new Doctor();
                doctor.setDoctor_id(set.getString("doc_id"));
                doctor.setDoctor_name(set.getString("doc_name"));
                //doctor.setPassword(set.getString("doc_password"));
                doctor.setMobile(set.getString("doc_mobile"));
                String workday= set.getString("doc_workday");
                if(workday.contains("Monday")) {
                    doctor.setWorkOnMon(true);
                }
                else {
                    doctor.setWorkOnMon(false);
                }
                if(workday.contains("Tuesday")) {
                    doctor.setWorkOnTue(true);
                }
                else {
                    doctor.setWorkOnTue(false);
                }
                if(workday.contains("Wednesday")) {
                    doctor.setWorkOnWed(true);
                }
                else {
                    doctor.setWorkOnWed(false);
                }
                if(workday.contains("Thursday")) {
                    doctor.setWorkOnThu(true);
                }
                else {
                    doctor.setWorkOnThu(false);
                }
            }
        });
        if(query.size() == 0 || index > query.size()){
            return null;
        }
        return (Doctor) query.get(index - 1);
    }catch(Exception e){
        e.printStackTrace();
        return null;
    }
}
```

附：

```
        else {
            doctor.setWorkOnThu(false);
        }
        if(workday.contains("Friday")) {
            doctor.setWorkOnFri(true);
        }
        else {
            doctor.setWorkOnFri(false);
        }
        if(workday.contains("Saturday")) {
            doctor.setWorkOnSat(true);
        }
        else {
            doctor.setWorkOnSat(false);
        }
        if(workday.contains("Sunday")) {
            doctor.setWorkOnSun(true);
        }
        else {
            doctor.setWorkOnSun(false);
        }
        return doctor;
    }
    if(query.size() == 0 || index > query.size()){
        return null;
    }
    return (Doctor) query.get(index - 1);
}
catch(Exception e){
    e.printStackTrace();
    return null;
}
}
```

#### 4.2.7 判断手机号是否有效

实现功能：在注册时粗略判断该手机号是否为合法的手机号

算法实现：判断字符串长度是否为 11 位，是否以 1 开头，是否每一位都是数字。（仅以此粗略判断）

代码实现：

```
private boolean checkPhoneNumber(String mobile) {  
    int n = mobile.length();  
    if(n != 11) {  
        return false;  
    }  
    if(mobile.charAt(0) != '1') {  
        return false;  
    }  
    for(int i = 0; i < n; ++i) {  
        if(!Character.isDigit(mobile.charAt(i))) {  
            return false;  
        }  
    }  
    return true;  
}
```

## 第五章 用户手册

### 5.1 医生使用手册

在使用该系统前先确保电脑已安装 Java ([https://www.java.com/zh\\_CN/download/manual.jsp](https://www.java.com/zh_CN/download/manual.jsp)) 和 JRE ( <http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html>)。



打开高血压社区管理系统，默认选择的角色是社区医生。点击注册进入注册页面。



填写相关信息进行注册。注意姓名不能为空，手机号须有效（11 位数字），密码与确认密码须一致。否则无法注册成功。



完成注册后会提示“注册成功”。点击确定后跳转到主页面。以后使用可以登录进入主页面。屏幕中央显示欢迎登录，及医生您的姓名  
其中菜单栏包括患者信息、日程安排、设置、关于四个板块。

如果没有患者关联您，则查看患者一览表会提示“目前还没有患者关联您”。



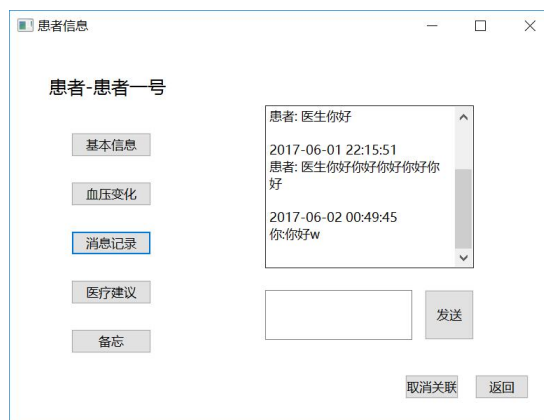
若已有患者关联您，则点击患者信息的患者一览可以查看关联自己的所有病人的姓名，点击进入后可以查看该患者的详细信息，或者取消关联该患者。



基本信息一栏中包括年龄、性别、身高、体重、家族疾病、既往病史等资料。



血压变化一栏中可以查看患者的血压变化。单击按钮<>可以进行翻页操作。血压为倒序查看，最先显示的是最新一次血压。



消息记录一栏中可以查看与患者的消息记录。默认显示最新的消息。



医疗建议一栏中可以给患者发送相关医疗建议。

备忘一栏可以记录一些事项。

点击设置的个人信息，可以对自己的姓名、电话、QQ、工作时间等进行修改操作。

点击设置的修改密码，可以修改自己的密码。

点击关于的版权信息，可查看该系统的作者信息。

点击关于的退出即可退出该系统。

## 5.2 患者使用手册

在使用该系统前先确保电脑已安装 Java ([https://www.java.com/zh\\_CN/download/manual.jsp](https://www.java.com/zh_CN/download/manual.jsp)) 和 JRE ( <http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html>)。

初次使用，打开高血压社区管理系统，选择角色为高血压患者。点击注册进入注册页面。

先填写姓名手机号和密码等信息。注意姓名栏不能为空，手机号须有效（11 位以 1 开头的数字），密码和确认密码须一致。

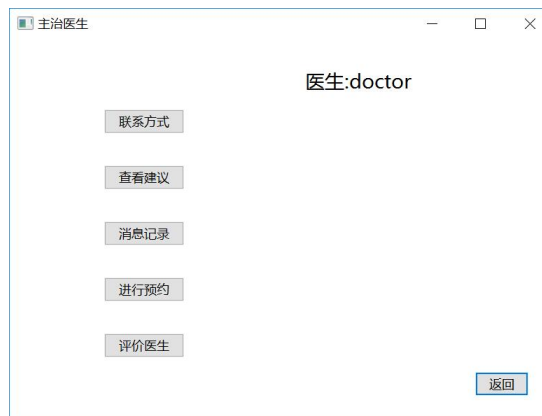
跳转到下一页面填写自己的详细资料。



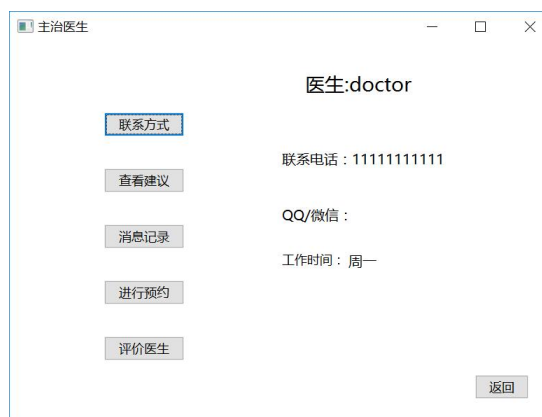


注册成功后，跳转到主页面，该页面会显示欢迎登录，您的名字。

其中菜单栏包括查看医生、设置、关于三个板块。



点击查看医生的主治医生，进入自己的主治医生界面，若未关联主治医生则无法访问该页面。



点击联系方式一览，可以查看医生的联系方式及工作时间。



点击查看建议，可以获得主治医生给的医疗建议。



点击消息记录，可以查看自己与主治医生的聊天记录。



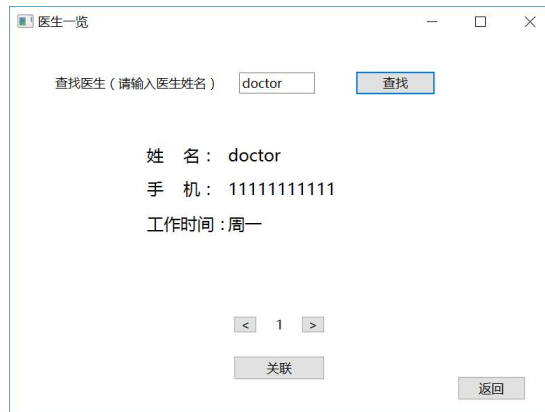
点击进行预约，可以对主治医生进行预约。



点击评价医生，可以对主治医生进行评分操作。



点击查看医生的医生一览菜单，可以进行查找医生的相关操作。  
如果输入内容为空，点击查找则显示所有医生的信息。



可以在该页面进行关联医生操作。

个人信息

姓名: 患者一号      年龄: 80

身高: 1.7 m      体重: 50 kg

家族疾病: 无1      既往病史: 无2

QQ: 11111111      手机号: 111111111111

提交      返回

点击设置的个人信息，可以对自己的姓名、年龄等进行修改操作。

更新血压

收缩压: 123      舒张压: 82

测定时间: 2017/ 6/23      20:43:38

提交      返回

点击设置的更新血压，可以更新自己的血压值。

修改密码

原密码:      新密码:      确认密码:

确认      返回

点击设置的修改密码，患者可以修改自己的密码。

点击关于的退出系统即退出该系统。

## 第六章 回顾总结

### 6.1 搭建系统过程中的问题

问题：对数据库进行查找操作，若查找结果为空产生了 `java.lang.IndexOutOfBoundsException` 错误。

解决：将 Dao 中的 find 函数的判断条件 `if(query == null)` 改为 `if(query.size()==0)`

问题：当数据库中的主键设置为自增时，插入操作若涉及到该主键则无法成功插入数据。

解决：不对数据库中的主键进行操作，仅插入除主键之外的数据。

问题：在页面之间切换时起初不了解如何实现参数在各窗体间的传递。

解决：通过编写传递函数，在窗体打开时传入某特定参数或实体，实现参数传递。

过程中还存在一些小问题通过 CSDN 博客、StackOverflow 等平台得到了解决。

### 6.2 该系统尚存在的不足

- 数据库仅实现本地操作，未实现远程访问数据库的功能（尝试用自己的 VPS 作为服务器地址访问也没成功）
- 未添加找回密码功能（由于不具有短信验证的功能，无法确认找回密码的操作者是不是用户自身，为了安全起见故暂时不设找回密码）
- 血压显示不太直观，需要医生手动翻页查看，未制成折线图形式
- 查找医生的模糊查找只有“%输入内容%”的方式。若输入 AB 则无法查找到名为 ACB 的医生
- 未完善注销账号功能

### 6.3 回顾总结

寒假的时候自学了一些 Java 的知识，所以在数据库的实验课上当大部分同学一头雾水时还是较好地跟上了老师的节奏。在一些同学需要帮助的时候也帮她们解决了许多问题。第一次完成代码量如此浩大的项目，4 个月，30 个 Java 文件，近万行的代码量……在 GitHub 上传更新了无数遍，生怕自己的一处修改使整个系统崩溃。最后还是较好地完成了高血压社区管理系统的设计，对自己的代码设计有了很大的提升，也巩固了相关的数据库理论知识。并且通过编写设计报告，明确了一个数据库项目的设计需要大量的准备工作。虽然耗时很长，不过收获也颇丰。