需求分析

建立一个管理用户的基本信息和体质健康信息的网站

使用范围

参与体质测试的上海市民

功能要求

- 注册(Register)：录入新用户的基本信息，包括姓名、密码和邮箱，体质信息包括性别、身高、体重。
- 加密：密码直接存储在数据库中，若数据库被黑客盗取可能会导致大面积用户被黑，而在加入数据库之前使用hashlib下的MD5进行加密再存储可以确保黑客无法获得用户的真实密码。
- 数据处理：针对录入的体质信息计算体质指数，BMI=体重(kg) ÷身高^2(m)
- 登录(Log in)：输入用户的姓名和密码后进入系统，可查看已录入的用户基本信息和体质信息
- 查找(Find unhealthy users)：根据条件（BMI<18.5 or BMI>24）查找出不健康的用户并显示
- 修改(Update)：输入需要修改的用户姓名和密码，可修改该用户的性别、邮箱、身高和体重信息
- 删除(Delete)：输入需要删除的用户姓名和密码，删除该用户的所有信息

逻辑设计

包括两个基本表：设计基本表时考虑到用户的登录信息与健康信息虽然都可算作用户的属性，但为了用户信息安全考虑，防止登录信息与用户的健康及其他信息同时被盗取，我们将它们分成不同的实体，以减少信息泄露的可能性。其中login_info与user_info两个实体的存在都直接依赖于user，因此这两个实体为弱实体，user为强实体。

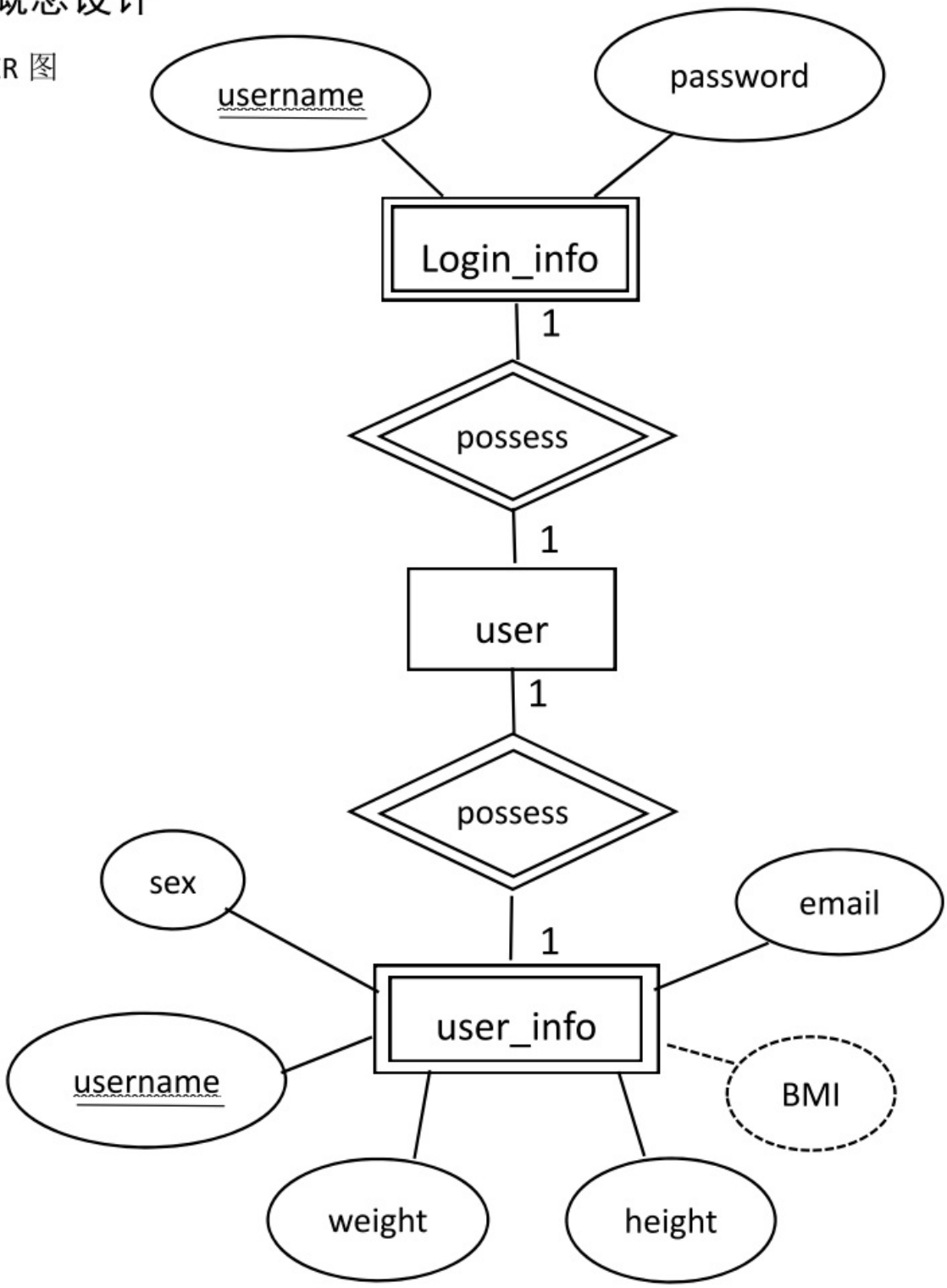
user;

login_info (username, password);

user_info (username,sex, email, height, weight, BMI);

概念设计

ER 图



物理设计

```
create table user(
      username varchar(10),
      password varchar(100) not null,
      PRIMARY KEY(username)
);
```

```
create table user_info(
    username varchar(10),
    sex char,
    email varchar(20),
    height smallint,
    weight smallint,
    BMI real,
    PRIMARY KEY(username)
);
```

系统结构和主要模块说明

我采用python中web框架flask与html语言实现该登录系统。

主体flask框架如下：

```
from flask import *
import MySQLdb
import os
import hashlib

def encrypt_password(password, username):
    result = hashlib.sha256(password + username).hexdigest()
    return result

def connect_db():
    db=MySQLdb.connect("localhost","root","123456","test")
    return db

app = Flask(__name__)
app.config['SECRET_KEY'] = '123456'
app.secret_key = "123456"

def init_db():
    db = connect_db()
    db.cursor().execute("drop table if exists user")
    db.cursor().execute("drop table if exists user_info")
    db.cursor().execute("create table user(username varchar(10),password varcha
r(100) not null,PRIMARY KEY(username));")
    db.cursor().execute("create table user_info(username varchar(10),sex char,e
mail varchar(20),height smallint,weight smallint,BMI real,PRIMARY KEY(username)
);")
    db.commit()

init_db()
```

该部分实现了各类库的导入，加密函数的定义，数据库的连接函数定义，以及数据库的初始化函数。

```python
@app.route('/')
@app.route('/index',methods=['GET','POST'])
def index():
    if request.method == 'GET':
        return render_template("register.html")
    if request.method == 'POST':
        db = connect_db()
        cur = db.cursor()
        if not request.form.get('username') == None:
            u = request.form['username']
            p= encrypt_password(request.form['password'],u)
            bmi = 10000*int(request.form.get('weight'))/(int(request.form.get('height'))*int(request.form.get('height')))
            cur.execute('insert into user values (%s, %s)',(u,p))
            cur.execute('insert into user_info (username, email, sex, height, weight, BMI) values (%s, %s, %s, %s, %s, %s)',[request.form.get('username'),request.form.get('email'),request.form.get('sex'),request.form.get('height'),request.form.get('weight'),bmi])
            db.commit()
        flash('New users was successfully posted')
        return redirect(url_for('login'))
```

该部分定义了主页面即注册页面，允许用户注册同时输入个人信息，并通过sql插入语句将登录信息与用户个人信息导入到到对应的数据库中。最后重新定向至登录页面。

```python
class loginError(Exception):
    def __init__(self, value):
        self.value = value
    def __str__(self):
        return repr(self.value)

@app.route('/login', methods=['GET','POST'])
def login():
    db = connect_db()
    cur = db.cursor()
    if request.method == 'GET':
        return render_template("login.html")
    if request.method == 'POST':
        u = request.form['username']
        p = request.form['password']
        try:
            cur.execute("SELECT username FROM user WHERE username = %s",(u,))
            if not cur.fetchone():
                raise loginError('wrong username!')
            cur.execute('SELECT password FROM user WHERE username = %s',(u,))
            password = cur.fetchone()[0]
            if encrypt_password(p,u) == password:
                return redirect(url_for('show'))
            else:
                raise loginError('wrong password')
        except loginError as e:
            return render_template('login.html',error=e.value)
```

该段代码允许用户输入之前用户名与密码，并通过sql查询语句从登录信息数据库中查到对应的用

户名与密码的MD5码，将之与用户输入进行匹配，如果未获取对应用户名，则在页面标题显示“`wrong username!`”

密码不正确则在标题显示“`wrong password`”，当用户成功登陆将重新定向至显示用户信息的页面。

```
1.
2.  @app.route('/show',methods=['GET','POST'])
3.  def show():
4.      db = connect_db()
5.      cur = db.cursor()
6.      count = cur.execute('select * from user')
7.      cur.execute('select * from user_info')
8.      result = [dict(username=row[0], sex=row[1], email=row[2], height=row[3], we
    ight=row[4], bmi=row[5]) for row in cur.fetchall()]
9.      return render_template('show_users.html', result=result)
```

该段从数据库中提取用户信息，如email以及身高体重BMI，并显示在对应的html页面上。在页面上存在三个超链接，分别指向update页面，注册新用户页面，删除旧用户的页面以及查找不健康用户页面。

```python
@app.route('/update', methods=['GET','POST'])
def update():
    db = connect_db()
    cur = db.cursor()
    if request.method == 'GET':
        return render_template("update.html")
    if request.method == 'POST':
        u = request.form['username']
        op = request.form['password']
        np = request.form['new_password']
        e = request.form['email']
        h = request.form['height']
        w = request.form['weight']
        try:
            cur.execute("SELECT username FROM user WHERE username = %s",(u,))
            if not cur.fetchone():
                raise loginError('wrong username!')
            cur.execute('SELECT password FROM user WHERE username = %s',(u,))
            password = cur.fetchone()[0]
            if encrypt_password(op,u) == password:
                cur.execute('UPDATE user_info SET email = %s WHERE username = %s',(e,u))
                p = encrypt_password(np,u)
                cur.execute('UPDATE user SET password = %s WHERE username = %s',(p,u))
                cur.execute('UPDATE user_info SET height = %s WHERE username = %s',(h,u))
                cur.execute('UPDATE user_info SET weight = %s WHERE username = %s',(w,u))
                db.commit()
                return redirect(url_for('show'))
            else:
                raise loginError('wrong password')
        except loginError as e:
                return render_template('update.html',error=e.value)
```

该段为update页面，主要实现对已存在用户的信息更新，在用户输入之前正确的用户名与密码后，可以更改密码，email地址以及身高体重。

```python
@app.route('/delete', methods=['GET','POST'])
def delete():
    db = connect_db()
    cur = db.cursor()
    if request.method == 'GET':
        return render_template("delete.html")
    if request.method == 'POST':
        u = request.form['username']
        p = request.form['password']
        try:
            cur.execute("SELECT username FROM user WHERE username = %s",(u,))
            if not cur.fetchone():
                raise loginError('wrong username!')
            cur.execute('SELECT password FROM user WHERE username = %s',(u,))
            password = cur.fetchone()[0]
            if encrypt_password(p,u) == password:
                cur.execute('DELETE FROM user WHERE username = %s',(u,))
                cur.execute('DELETE FROM user_info WHERE username = %s',(u,))
                db.commit()
                return redirect(url_for('show'))
            else:
                raise loginError('wrong password')
        except loginError as e:
            return render_template('delete.html',error=e.value)
```

该段为删除页面，可以将正确输入用户名与密码的用户所有信息从数据库中删除，利用了sql删除语句。

```python
@app.route('/unhealthy', methods=['GET','POST'])
def show_unhealthy():
    db = connect_db()
    cur = db.cursor()
    cur.execute('select * from user_info where BMI not between 18.5 and 24')
    result = [dict(username=row[0], sex=row[1], email=row[2], height=row[3], we
ight=row[4], bmi=row[5]) for row in cur.fetchall()]
    return render_template('show_unhealthy.html', result=result)
```

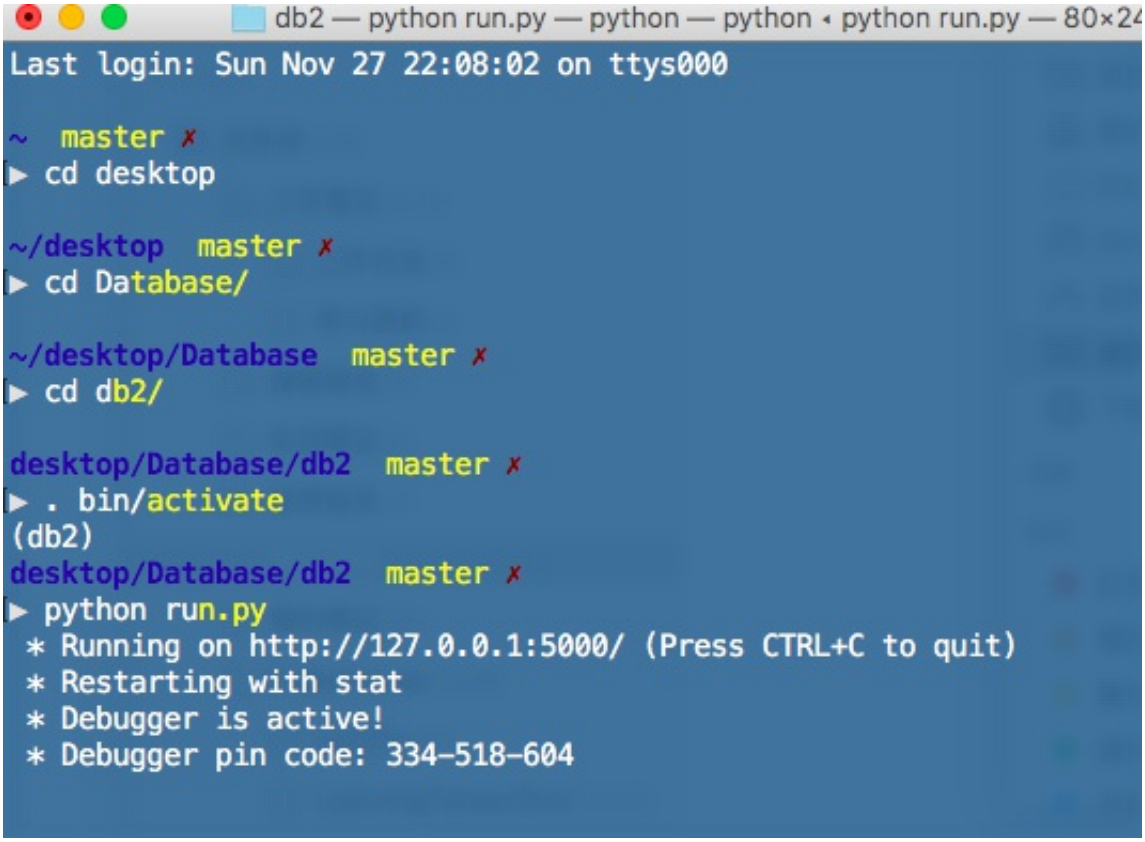
该段为显示不健康用户的页面，将BMI指数不在健康范围用户信息显示在页面上。

```python
app.run(debug = True)
```

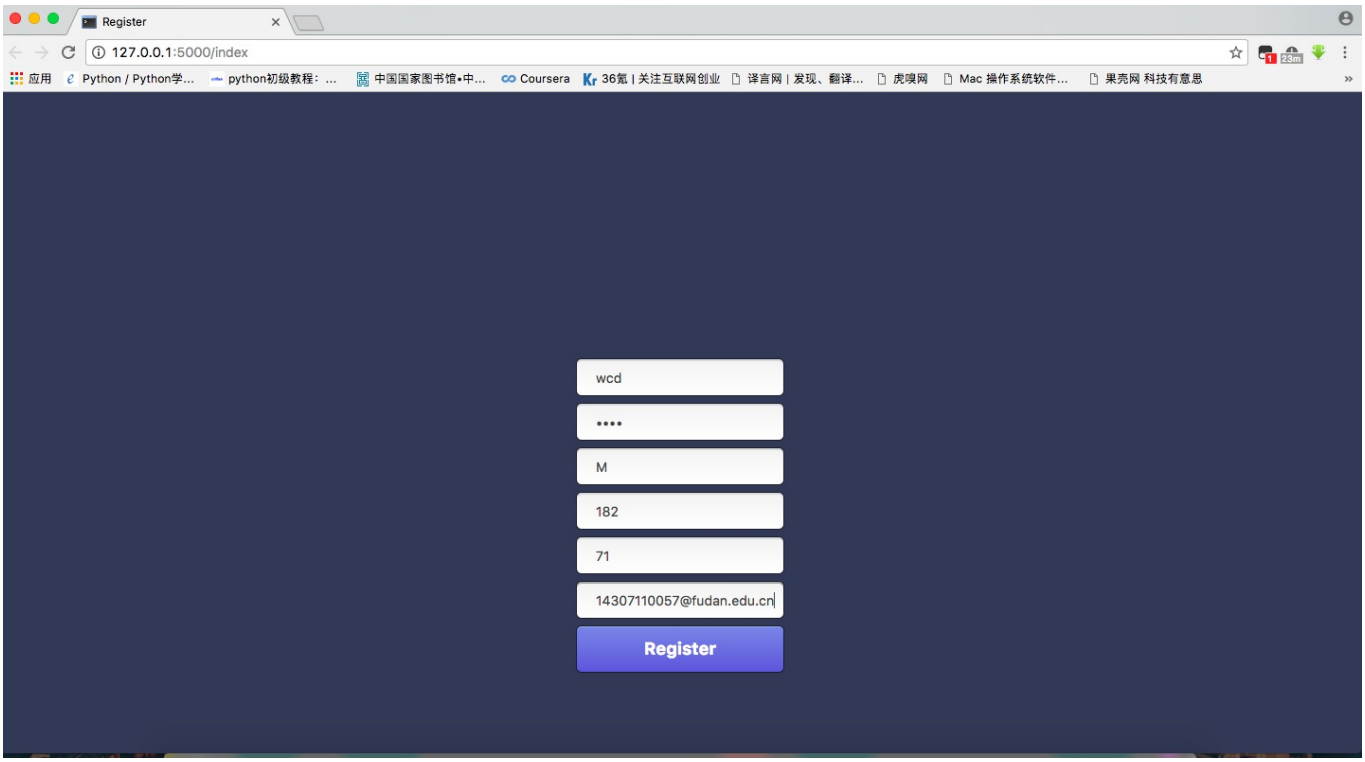
程序的运行句，并可以在网站上线情况下进行调试。

数据库系统语句被包含在了python代码中，另有一些html文件可以在/templates文件夹下找到。

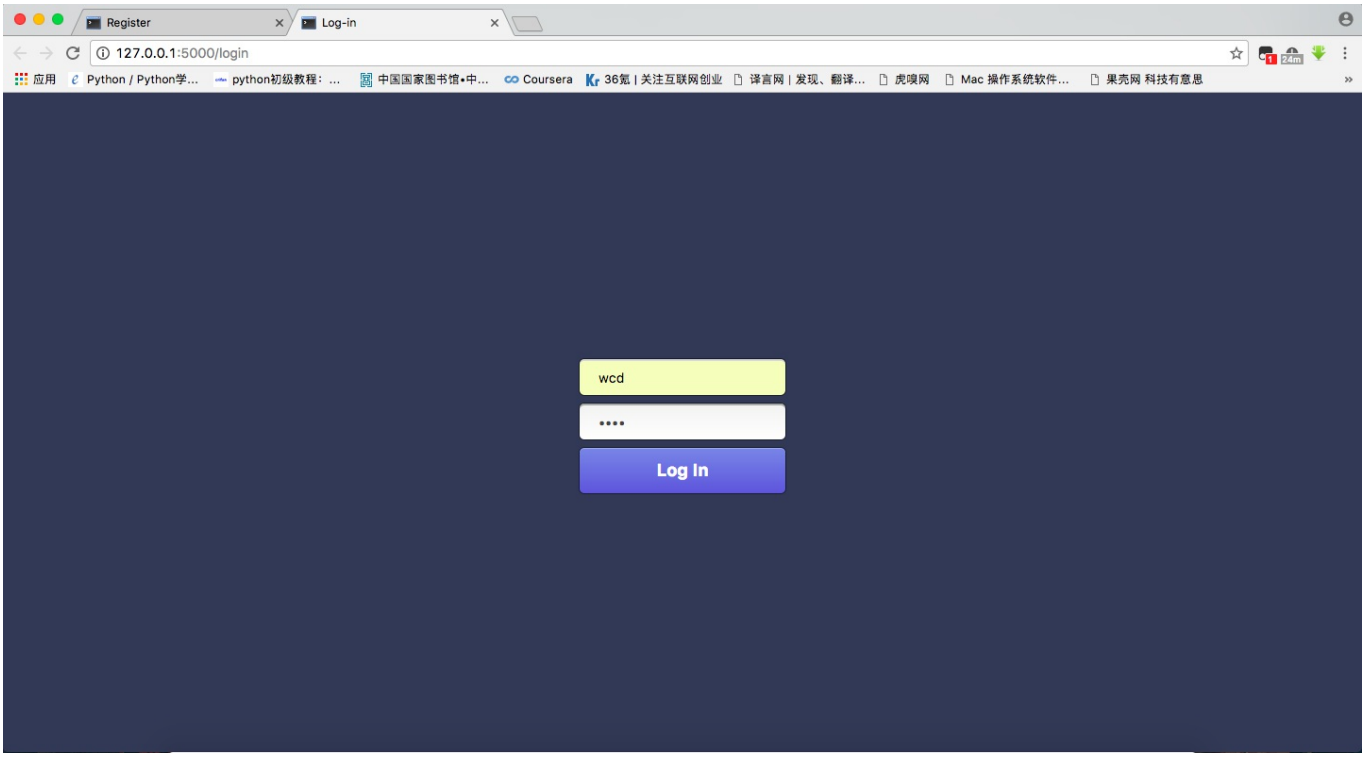
界面截图



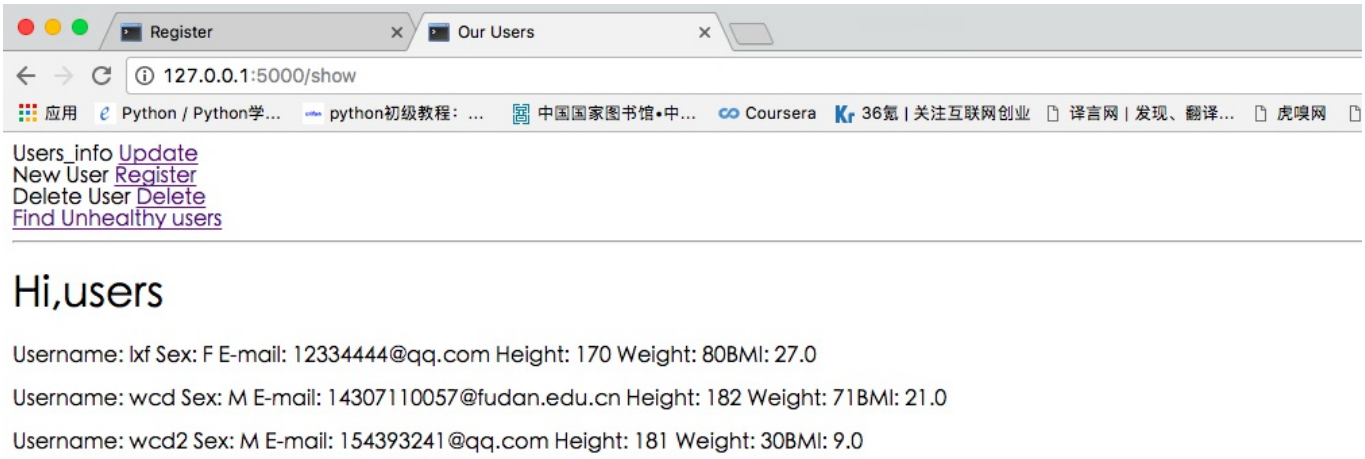
在command line里如上操作。



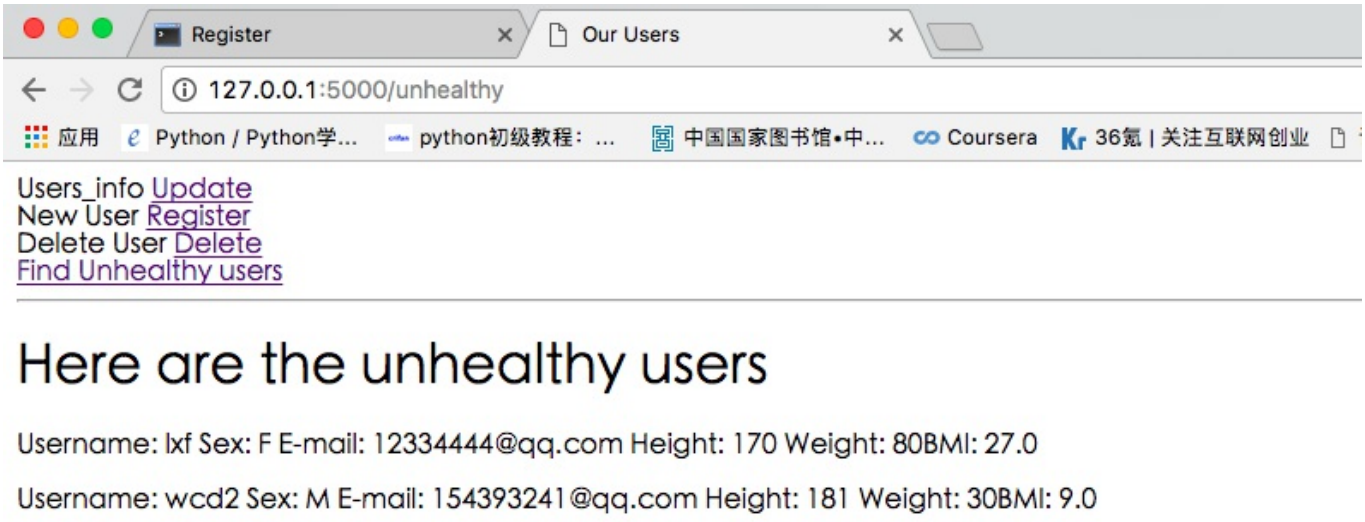
注册界面



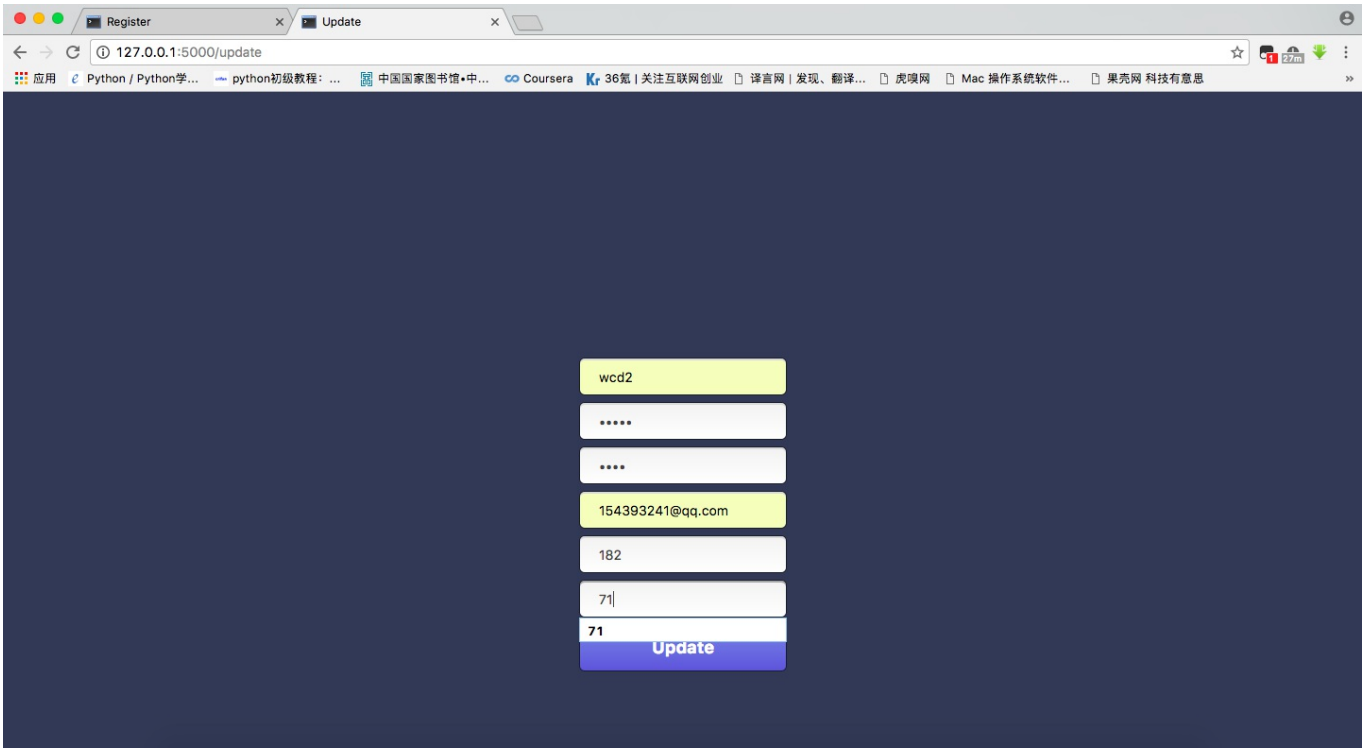
登录界面



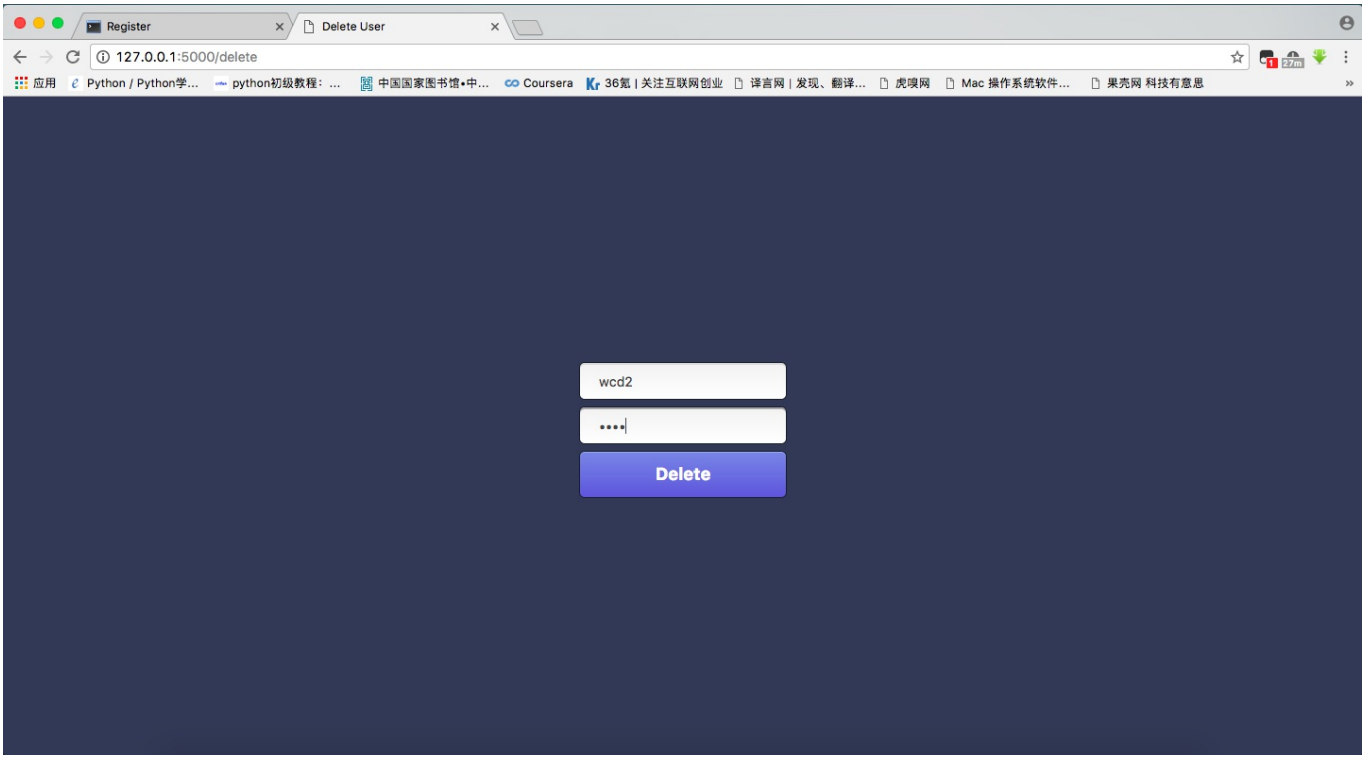
显示用户信息界面



查询不健康用户界面



更新用户信息界面



删除用户界面

实验总结

系统不足：

1、显示界面不够友好，可以选择更好的界面模板。

2、对于已经存在的用户再次注册会出现报错而非要求用户另取用户名

3、为了方便调试，每次运行网站都会重新建立数据库，从而调试时每次都需要输入一遍用户信息。

4、对用户的健康评估仅仅基于BMI指数，科学性有待加强。

未来计划：

1、注册域名，并将该网站连入互联网，目前网站仅仅建立在自己笔记本的网路内，但未来有可行性。

2、建立更全面的信息收集以及健康评估体系，对用户负责。

3、建立更好的登录体系与安全体系，防止用户信息泄露。

实验感想：

经过本实验我将本学期学到的数据库的基本概念以及sql语句应用到了实际的项目中，并结合python的功能与html强大的可视化能力，完成了数据库的project，也使自己的代码能力得到历练，对项目的统筹能力得到提升，收获颇丰！