后端工作(使用django+sqlite3)：

A.注册、登录、忘记密码,admin状态下的操作(显示所有用户名和密码)：

(1) 函数的功能实现

(2) 异常的处理(如用户名和密码不匹配，不存在用户名，用户名格式不对等)

(3) sqlite3数据库在django下的增加、查询、修改操作

(4) 对应的测试网页

1.注册

I.识别网页是否向服务器传送数据

II.判断是否有用户名栏和密码栏 和 用户名栏和密码栏里是否有数据

III.对用户名进行匹配('@')，判断用户名是邮箱还是手机

IV.判断用户名是否符合邮箱或手机的格式

V. 判断密码是否符合格式

VI.判断用户名(用户或邮箱)是否在数据库里

VII.增添用户数据

VIII.把用户名存储到cookie里，返回网页(通过cookie获得用户名，在网页对应地方显示)

2.注册 ——与登录的区别：

VI变为：判断用户名和密码(一个整体)是否在数据库里

VII变为：判断用户名是否admin账号

3.忘记密码 ——与登录的区别：

VI变为：判断用户名(用户或邮箱)是否在数据库里，如果在，则把验证信息发送到phone/email，等待核实。如果核实成功，则进行后续操作

VII变为：对用户的密码进行修改

语句：

I.获得数据：username = request.POST.get('username', None)

II.字符串匹配：pos\_at = True if ("@" in username) else False

III. 数据库查询数据：User.objects.filter(phone=username)

IV.数据库增添数据：User.objects.create(email=username, password=pwd)

V. 数据库修改数据：User.objects.filter(email=username).update(password=pwd)

VI.cookie操作：

response = HttpResponseRedirect(reverse("main\_page"))

response.set\_cookie("username", username)

return response

return render(request, "main\_page.html", {"username": request.COOKIES["username"]})

个人总结：

一开始写的过程，没有实现模块化处理，导致request处理函数几乎是if,else的嵌套，不仅代码可读性差，而且修改难度大，与request处理函数(register,login,password\_forget)间没有形成关联。其实这三个request处理函数的内容方式差不多，把函数分成几个模块，三个函数中有几个模块是相同的，有几个模块是相似的。把函数模块化，不仅思路清晰，而且代码实现难度降低[写完一个函数，另外两个函数在原有函数上进行修改]，当代码需要修改时，需要修改的地方容易找到，不容易改错。

II.文章、评论

1.文章和所属评论的显示

2.用户对于文章、评论的增加、修改、操作

3.用户对于自己所写的文章、评论的显示

4.对应的数据库操作