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.如果网页有向服务器传送数据【request.method == 'POST'】，则跳转到II,否则结束**(这里的结束指的是异常处理，针对不同的异常有不同的return值)**

II.如果网页不存在用户名栏和密码栏或用户名栏和密码栏里无数据【username = request.POST.get('username', None) ; if username is None ; if username == ""】，对应不同的结束，否则跳转到III

**(用户名栏里的数据为username，密码栏里的数据为pwd)**

III.查找用户名里是否有'@'【pos\_at = True if ("@" in username) else False】，跳转到IV

IV.判断用户名是否符合邮箱或手机的格式，如果不符合，结束；否则跳转到V

V. 判断密码是否符合格式，如果不符合，结束；否则跳转到VI

VI.判断用户名(用户或邮箱)是否在数据库里

【 if not pos\_at and User.objects.filter(phone=username) \

or pos\_at and User.objects.filter(email=username):】，如果是则结束；否则跳转到VII

VII.增添用户数据【User.objects.create(email=username, password=pwd)】，跳转到VIII

VIII.把用户名存储到cookie里【 response = HttpResponseRedirect(reverse("main\_page"))

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

return response】

【cookie：为了辨别用户身份、进行 session 跟踪而储存在用户本地终端上的数据（通常经过加密）】，网页跳转到main\_page，从cookie获取用户名【return render(request, "main\_page.html", {"username": request.COOKIES["username"]})】

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【 if not pos\_at and User.objects.filter(phone=username, password=pwd) \

or pos\_at and User.objects.filter(email=username, password=pwd):】，如果不是则结束；否则跳转到VII

VII变为：如果用户名是admin账号，跳转到对应网页，否则跳转到VIII

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，如果不是则结束；否则验证信息发送到phone/email，等待核实。如果核实成功，跳转到VII，否则结束

VII变为：对该用户的密码进行修改【 if pos\_at:

User.objects.filter(email=username).update(password=pwd)

else:

User.objects.filter(phone=username).update(password=pwd)】，跳转到VII

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