# Web数据库大作业 用户说明文档

## 项目简述

我们的Web应用项目模仿知乎和简书，希望做一个知识分享和讨论平台。网站功能描述以及设计思路如下：

1 登录注册

2 动态下拉刷新信息

3 每个信息模块包含的功能:

follow 作者

留言评论

点赞

收藏

4 个人账户界面包含功能

基本信息的编辑

收藏文章的整理(分组与删除)

好友列表的编辑(可点击头像进去其主页)

5 他人信息界面包含的功能

基本信息(仅查看)

发私信

而我们决定做这么一个模仿知乎与简书的Web应用的理由如下：

1. 需求是清晰的，不存在不知道该开发什么的问题。

2. 具有一定的复杂性，对我们来说是个机会和挑战。

3. 这一类网站基本上涵盖了我们日常生活中大多数访问的网站的组织形式。不但能够学习到成熟网站的信息架构和交互设计框架，而且能够了解前端开发最常涉及到的哪些最基本的组件，它们是怎么实现 与应用到具体的上下文中的，以及后端常见的数据组织形式和处理方式。

4. 我们全程都尽力按照现实开发的过程来，希望提高效率，提高和他人配合的能力和意识。

## 项目方案

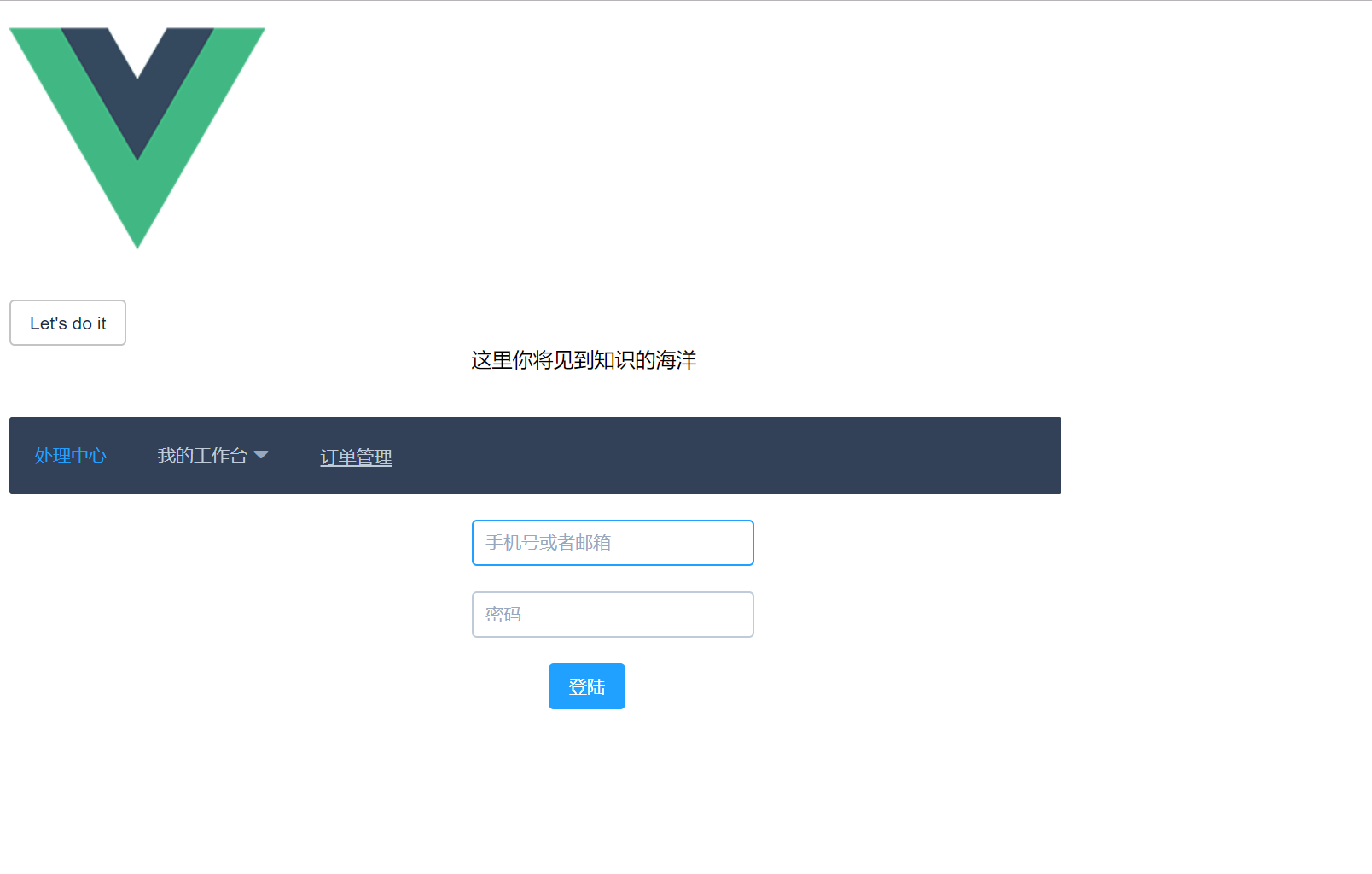
本项目是一个Web应用，主要分为前端与后端两个板块，我们希望采用前后端完全分离的结构来实现本项目。即前端负责显示页面，各个页面的跳转等等，而后端负责相应请求，并提供存储在数据库的各种数据的操作。前端我们主要使用HTML、CSS、Vue.js框架以及基于Vue.js的ElementUI组件库来进行开发，后端则是采用Python的框架Django。

同时，我们小组分工明确，采用真正团队开发的模式，辅以Github、工作进度表以及定期的小组讨论会议来推进项目进展，借此来提高我们的项目开发能力。

## 已完成页面

下面的是我们已完成的部分页面：

**登录注册页面：**



目前实现了一个基本登录注册页面的基本布局和样式，细节有待调整。

**个人主页：**



基本的网页布局已完成，有待接入其它网页以及加入各种按钮、功能的处理逻辑。

**评论页面：**



已布局好基本的评论内容。

## 后端简述：

**后端数据库表的设计：**

后端采用流行的Python框架Django，用来处理各种请求的响应，URL的匹配以及后台数据库的相关增删查改工作。

1. **user\_data 表——对应User类**

该表用于存储用户数据，包括用户账号、密码等，在后面的开发中我们会逐步加入用户账号的其它个人信息，完善相关字段，以满足网站的需求。

每个用户对表中一条记录，而这一条记录在Django的封装接口下以一个基于Python类创建得到的对象呈现，因此我们可以直接对该User对象访问来实现对数据库和表中数据的操作。

下面是目前User的设计要点：

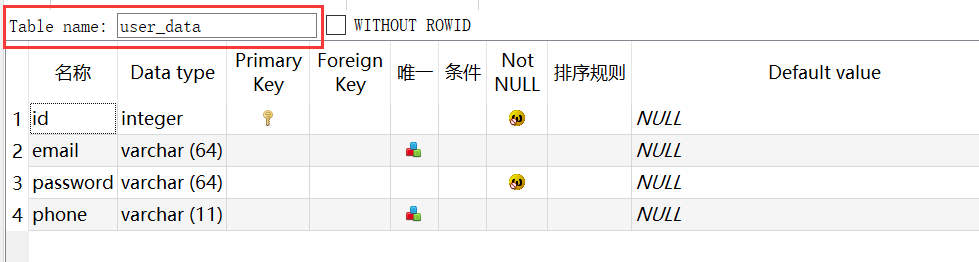
**字段：**

(ID)：隐式生成的字段；自增类型的整数；该表的主键。由phone和email的联合主键选项自动生成。

phone：用户的手机号码。最大长度为11位的字符串；Unique属性为真，意味着整个表中只能有唯一的手机号码，不允许重复；默认缺省值为null。

email：用户的邮箱地址。对大长度为64的字符串，并采用Django的EmailField限制；Unique值为真，不允许重复；默认缺省值为null。

password：用户账号的密码。最大长度为64的字符串。



**约束：**

联合主键：phone 与 email联合唯一，并且由此产生一个ID字段，以ID作为表的主键。意味着用户可以使用手机号和邮箱地址的其中一个作为注册账号，但后面补充信息时不能填写已存在的手机号或者邮箱地址。

唯一：phone、email字段都有唯一限制，不允许重复出现的值。

1. **comment\_data 表——对应Comment类**

该表用于存储在文章中的评论数据，包括评论内容、作者、发布时间等。

**字段：**

(ID)：隐式生成的字段；自增类型的整数；该表的主键。由author和content的联合主键选项自动生成。

content：该条评论的内容。采用Django提供的TextField限制域，本质上是一个字符串。

author：该条评论的作者。此处使用外键属性连接到上面的User中。

time：评论的发布时间。使用Django提供的DateTimeField限制域；使用auto\_now\_add选项，自动提供当前系统时间为默认缺省值。



**特殊关系映射：**

外键：将User链接到author作为外键。意味着每条记录中的author都对应user\_data

中的一条记录，是一种特殊的多对一关系（多条评论对应一个用户）。

隐含的多对多关系：后面将会提到，一个评论被对应到一篇文章，而一篇文章可以对应很多条评论记录。借助这个隐含的多对多关系，我们可以实现反向查询，通过评论找到它属于的文章。

**约束**：

联合主键：author 与 content 联合唯一，并且由此产生一个ID字段，以ID作为表的主键。意味着同一个作者发布的两个评论不能含有一模一样的内容。

1. **article\_data 表——对应Article类**

该表用于存储文章数据，包括文章的标题、作者、内容、评论、发布时间、浏览数和点赞数等等。

**字段：**

(ID)：隐式生成的字段；自增类型的整数；该表的主键。由author和title的联合主键选项自动生成。

title：文章的标题。最大长度为64的的字符串。

author：文章的作者。使用外键链接到User。

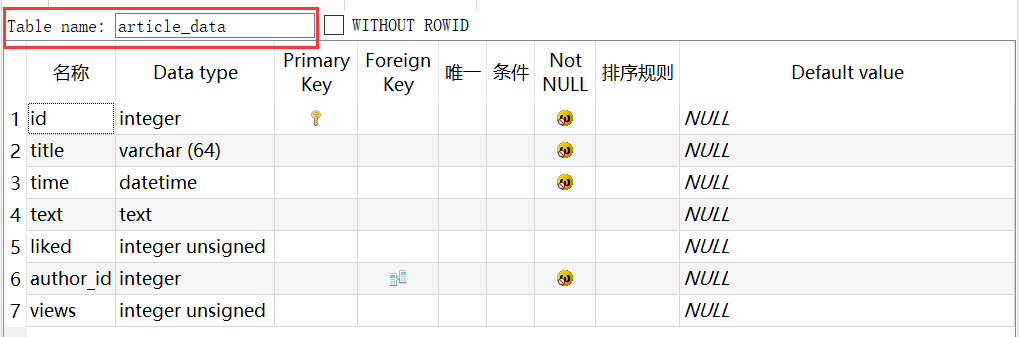
time：文章的发布时间。使用DateTimeField限制域；使用auto\_now\_add选项，自动提供当前系统时间为默认缺省值。

text：文章的主体内容。使用TextField限制域；默认缺省值为null。

comments：文章对应的评论。使用ManyToManyField 链接到Comment，产生多对多关系。

views：文章的阅读量。非负整数；默认初始值为0。

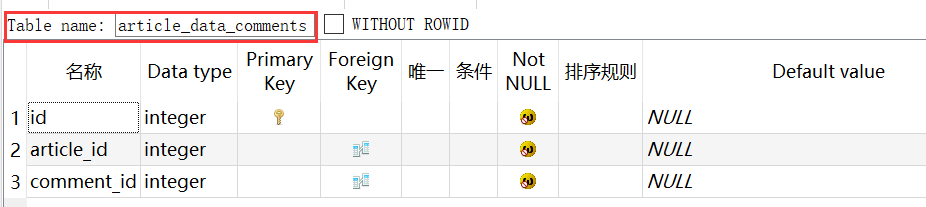
liked：文章的点赞数。非负整数；默认初始值为0。



**特殊关系映射：**

外键：将User链接到author作为外键。实现一篇文章对应外部表中一个用户记录。

多对多关系：借助ManyToMany关系将comment\_data表中的记录链接到comments字段。在Python中comments成员对应的是一个集合，该集合中包含该篇文章对应的所有评论。意味着定义了多对多关系后，我们可以通过文章记录直接得到其下面的评论，无需进行二次条件筛选查询。实际上这种多对多关系的映射是通过另外一个中间表存储各自的主键来实现表A中的一条记录与表B中一条记录的关联关系的存储。



**约束**：

联合主键：author 与 title联合唯一，并且由此产生一个ID字段，以ID作为表的主键。意味着同一个作者发布的两篇文章不能有一样的标题。

**后端处理工作：**

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

* 识别网页是否向服务器传送数据
* 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
* 对用户名进行匹配('@')，判断用户名是邮箱还是手机
* 判断用户名是否符合邮箱或手机的格式
* 判断密码是否符合格式
* 判断用户名(用户或邮箱)是否在数据库里
* 增添用户数据
* 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：

* 变为：判断用户名和密码(一个整体)是否在数据库里
* 变为：判断用户名是否admin账号

3.忘记密码 ——与登录的区别：

* 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
* 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"]})

## 个人感悟与收获

**郑川：**

1. 使用前端框架进行页面开发的时候，最重要的是搞清楚它的布局规则，

先设计页面，然后利用布局规则进行页面的划分，最后编码。

1. 要在一开始就考虑清楚这个页面是否需要响应式，这个在中途更换很费时费力
2. 页面布局不是单纯框架布局规则的结果，它是多方面因素的综合结果，要搞清楚这个结果是为什么，尤其它出乎你的意料的时候
3. 如果能力允许，还是要尽早进入源码学习的阶段，仅仅掌握一个UI框架没有多大的价值，思考它背后的设计思想在是最为重要的，也是你区别于其他人的所在。读懂ElementUI 的源码相当于读懂了他们的开发团队如何应用Vue.js。然后就可以尝试读Vue.js的源码了
4. 搭建前端页面不是一个难事，也不应该被当做一个难事，如果你觉得比较困难，而且长期如此感觉，说明你的进步不够快。快速搭建页面布局是检验你理解这个框架如何的好办法
5. HTML和CSS不应该像读课本那样去学习，它们确实有非常好的完备性，能实现各种需求，但是如果我们接触不到这些需求，那学习相关知识是没有价值的。只有拿到页面布局需求，从而丰富自己的知识结构，才是最正确的做法。我以前想看文案一样学这些琐碎的知识，根本记不住，虽然容易理解，但是没有实际使用最终都会忘记的。这是个方法论的问题，大家一定不要搞错，否则浪费很多时间。
6. 开会就是个清晰的表达自己的想法的宝贵机会，希望大家都能充分利用，组织好语言，然后表述清楚，在必要的时候对一些可能被听者理解错的概念进行区分和辨析，同时对比较难以理解的东西不妨作类比，举例子。

**陈冠斌：**

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。

**杨万里：**

1.前端和后端的信息交流，学会前端接收信息并传送到后端，然后后端处理信息交由前段显示。

2.背景特效的处理.（这个可搁置）

4.手机注册和邮箱注册的前端合并。

5.核心问题：文章主体和文章评论区。（后端表的设计+前端页面布局）

6.紧急问题：打通前后端（把django和vue.js组合起来。完成快的话可以加快点进度）。

Ps：template模板尽量用vue.js和element.ui处理，先确定布局。

7.vue+django整合:首先不需要自己构建数据库程序，django内部已经全部封装好了，我们只需要将全部的数据库内容添加到路由中。

交由数据库去实现相应的表。

前端:

创建完前端内容需要将前端内容全部映射到后端：

现在前后端已经全部创建完毕，但是问题是他们是分离的，运行在不同的服务器上，所以需要将二者绑定，采用的方法是在django上修改templateview。

接下来需要修改文件的模板，配置文件的搜索路径，最后运行程序。

8.前端代码疑问：对于组件还是有以下疑问:在修改完图标风格后，发现这里还是有很多问题，比如图标和字符串还是不能很好的排版，最好的方法还是修改图标的像素大小，使之尽量和字符串匹配。这个需要好好修改下。

9.这一周基本页面基本确定，但是与较为成熟的页面相比还是有很多不足，最严重的是布局问题，一个好的网页效果与布局是密不可分的

**姚乔日：**

暂时没写