作业 3——登录注册说明文档

191250024　丁炳智　软件学院

一、文件结构说明

此次作业涉及的文件主要包括：

1) 静态资源文件：/public/* (包含图片、js/html/css 文件)

2) MySQL 数据库文件：/sql/web3.sql

3) Node.js 服务端代码：/App.js

二、作业说明

0、主页：启动 App.js 后，在浏览器中输入 http://127.0.0.1:3000 即可访问

1、数据库表设计

数据库分为两张表：users、user_auth.

(1)users 表

包含注册用户的基本信息，以 uid 为自增主键，包含用户注册的邮箱 email、昵称 display_name、注册时间 signup_time、用户身份信息 role。其中用户身份 role 为预留维度，有 admin 和 user 两种。

(2)user_auth 表

包含用户的安全信息，以 auth_id 为自增主键，以 users.uid 为外键，包含由原密码“加盐”并哈希转换后的加密密码 passwd，以及该用户密码对应的“盐值”salt。由于转换密文时需要对每个用户注册的密码进行加盐，以便加强密文强度和后续密码验证，因此需要在 user_auth 表中以 salt 字段存放每个用户自己的“盐值”。

2、注册功能

(1)注册界面

(2)功能实现

①注册界面中含有六个字段：邮箱(用于今后的登录)、用户昵称、密码、重复密码、验证码和同意许可勾选框。每一个字段都实现了实时验证，即字段输入框失去焦点后，使用醒目红字提示用户：

②邮箱需要检查是否填写并验证格式，采用正则表达式进行验证，验证逻辑为：邮箱的字符数需在 6-32 之间，在“@”符号之前需要有至少两个字符，“@”符号之后可以有多个后缀：

```
/(?=([a-zA-Z0-9._\-]{2,}@[a-zA-Z0-9_\-]{2,}(\.[a-zA-Z0-9_\-]+)+))^\S{6,32}$/
```

③用户昵称需要检查是否填写并验证格式，采用正则表达式进行验证，验证逻辑为：昵称的字符数需在 4-20 之间，由字母、数字、汉字或下划线组成：

```
/((?=.*\d)|(?=.*[a-zA-Z])|(?=.*[\u4E00-\u9FA5])|(?=.*_))^\S{4,20}$/
```

④(**重点**)用户密码需要检查是否填写并验证格式和强度，采用正则表达式进行验证。其中：

I.用户密码格式验证逻辑为：字母、数字、符号两种及以上组合，8-20 个字符：

```
/((?=.*\d)(?=.*\D)|(?=.*[a-zA-Z])(?=.*[^a-zA-Z]))(?!^.*[\u4E00-\u9FA5].*$)^\S{8,20}$/
```

II.用户密码强度验证逻辑为：

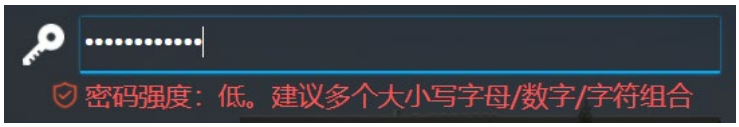
密码规范	得分
含十个及以上字符	+1
采用大小写字母混合	+1
含有数字	+1
含有特殊字符	+1

密码强度得分规则

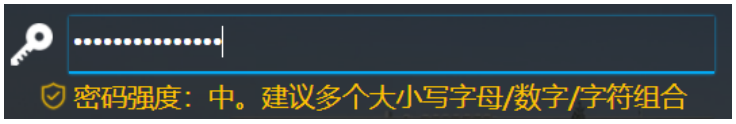
得分	密码强度判定
1/2	低
3	中
4	高

密码强度与分数对照表

对于得分规则的判断，均采用正则表达式进行，由于判断逻辑较为简单，代码不作展示。下面展示对于不同强度的密码，界面显示的实时强度提示：



输入 qiangdudi123：密码强度低



输入 Qiangduzhong123：密码强度中

输入 Qiangdugao16050?!：密码强度高

⑤对于重复密码字段，需要检查是否填写并检查是否与原密码输入一致，提示如下：

⑥(**重点**)对于验证码字段，需要进行验证码绘制、检查是否填写及是否填写正确。下面进行说明：

I.验证码绘制

采用 canvas 进行绘制，在 html 文件中先创建一个空 canvas 画布元素，然后对画布进行绘制。绘制思路为：给定 0~9/a~z/A~Z 字符数组，每次绘制时从中随机抽取 4 个字符；为它们随机分配四个较深颜色、随机旋转一定锐角；然后在画布上随机选取 4 个位置，将 4 个字符分别放置其中；对画布选择一个较浅背景色；在画布上随机绘制一些干扰线条和点。效果如下：

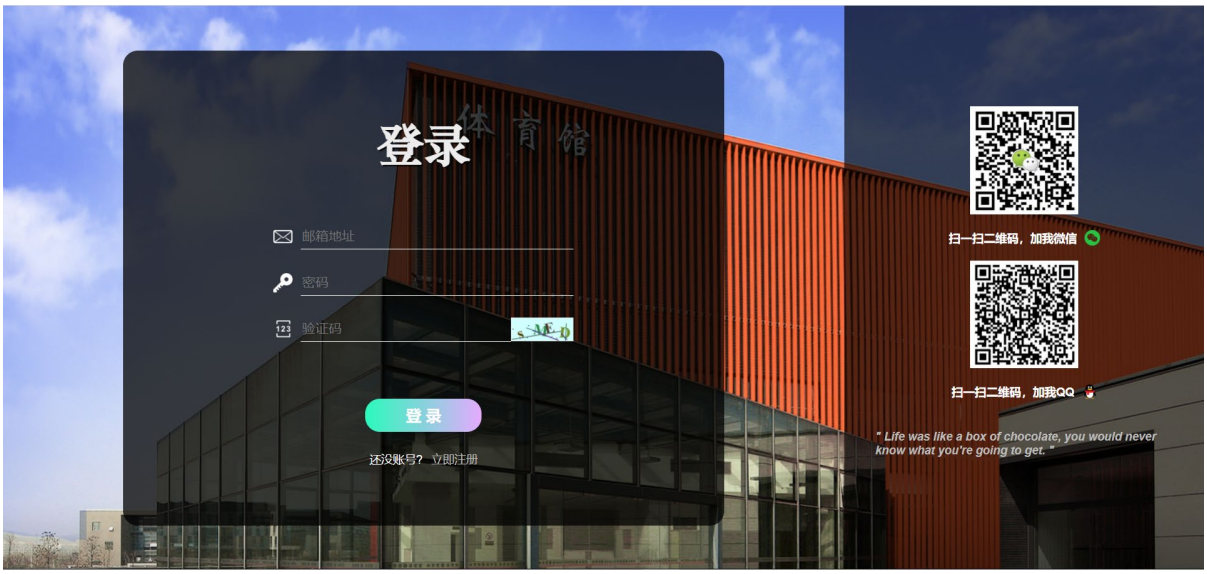
II.检查是否填写正确

用一个数组记录下绘制验证码时选取的 4 个字符，并与用户的输入进行比对即可。在比较时将这些字符都转为小写。

⑦对于勾选同意用户许可字段，判断逻辑较为简单，不再赘述。

3、登录功能

(1)登录界面



(2)功能实现

登录功能部分的实现相当于注册功能的子集，因此相关功能不再赘述。

4、(重点)密码加密

从安全性的角度出发，在客户端和服务端进行通信时，若使用明文传输密码，并以明文形式在数据库中存储密码，则存在极大的安全隐患，因此无论在前后端通信还是在存储重要数据时，都应该进行加密处理。常见的散列算法存在被彩虹表、碰撞攻击的危险，因此本次作业中采取了明文加盐、多次哈希迭代的方式进行加密，下面进行说明：

(1)哈希算法选取

传统哈希函数如 MD5、SHA-1 等，均存在易受碰撞攻击、可用彩虹表进行暴力破解的缺陷。本次作业中采用 SHA-2(Secure Hash Algorithm 2 — 安全散列算法 2)系列算法中的 SHA-256 算法，对于不同长度的输入串，其将会生成一个长度为 64 的加密字符串。

具体函数详见 /public/js/util/passwdHash.js

(2)加密完善——加盐

对于较为简单的密码，如“password”/“123456”等，很容易保留其 SHA-256 加密后的散列值，对于常见的密码制作一份表格，存放它们对应不同加密算法后的值，就可以将密码破解，此即“彩虹表”。为了化解这个隐患，可以采用“加盐”的方式——即向要加密的串前端添加一段完全随机的字符串。

在本次作业中，采用随机加盐的方式，即从 0~9/a~z/A~Z 中随机选出 32 个字符(可重复)组成字符串，作为用户密码对应的盐，将此值拼接到用户密码前端，并将拼接后的字符串进行 100 次哈希函数 SHA-256 迭代，最终得到加密后的密码，并将其和盐值一同存放在 user_auth 表中。

```
function genSalt(length) {
  // 生成盐
  // 由于盐生成的过程是随机的，因此没必要向攻击者保密盐
  const chars = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
  let result = ''
  for (let i = length; i > 0; --i) result += chars[Math.floor( x: Math.random() * chars.length)]
  return result
}

function genHashedPasswd(plainPasswd, salt) {
  // 盐加到原密码前面效果较好
  // 慢哈希
  let res = sha256( ascii: salt + plainPasswd)
  for (let i = 0; i < 100; i++) {
    res = sha256(res)
  }
  return res
}
```

在这里对第一页中的数据库表设计进行一下补充说明，之所以要分为 users 和 user_auth 两个表进行存储，就是为了方便管理用户的密码和盐值，由于盐值每一次都是完全随机生成的，因此需要对每一个用户的盐值进行保存，该值是随机且用户唯一的。

对于盐值，由于其随机性，无需对盐值进行保密，攻击者事先未知盐值，也就无法制作相应的彩虹表进行密码破解，在登录验证、修改密码的过程中，都需要获知用户密码的盐值，以进行密码验证。

5、其他功能

除作业要求的基本功能外，本次作业还额外实现了一些其他功能，下面对此进行说明：

(1)识别登录状态

用户登录后，在 window.localStorage 中预设一些项(如 uid 等)，用户退出登录后，将这些项移除。若用户未登录，则其访问主页(/main_page)、修改密码页(/change_password_page)、水印展示页(/watermark)等需要登录后才能访问的页面时，会自动跳到登录页面(/login_page)

用户在登录后，若访问登录页面(/login_page)，则会自动跳转到主页(/main_page)

用户登录来到主页后，主页会展示用户的昵称、邮箱等信息。

(2)修改密码

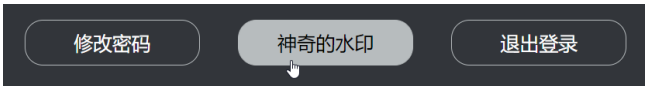
在主页中，用户可以点击“修改密码”进行密码修改，密码修改页面中同样有密码强度提示、验证码输入等交互信息。

(3)水印展示

将第二次作业的水印页面整合到了此次作业中，作为主页的一个子页面，点击“神奇的水印”按钮即可查看效果。

(4)较完善的交互细节

对于输入框，会根据输入内容实时向用户反馈检验结果，以醒目文字或元素抖动动画进行提示；对于按钮，当鼠标移至其上时，有明显的样式变换(如文本、按钮背景颜色渐变、指针变为手型等)，符合人机交互原则。



6、数据库表示例

下面进行注册后 users 和 user_auth 表的数据展示。

(1)users 表

uid	email	display_name	signup_time	role
1	191250024@smail.nju.edu.cn	Bingzhi_Ding	2021-12-11 18:51:34	user
2	1048927295@qq.com	丁炳智0123	2021-12-11 19:00:34	user
3	ding191250024@163.com	191250024丁炳智	2021-12-11 19:01:00	user
4	web3.homework@software.nju.cn	Web第三次作业示例	2021-12-11 19:01:41	user

(2)user_auth 表

auth_id	uid	passwd	salt
1	1	98f97fab3184cb921db3dc3438d0165fcb3201f868da546c31f21f0589bd8cac	MHcFijhpMnQGHQZjt9odgC0j2njKweLw
2	2	85634fddfd8c11c7313825dcc399e47b4a97fe9602d2c35e305a49f2f726c76b	XrTPyP87qrlFSOoVxXvTGYV9r6JvBJ0C
3	3	1bf4108c552fda05dc091fd00aa8d298424c50b28eb203f08f9b8bc47e046681	k77stwmFsB73UuB12LmWsLfRRadi9NgU
4	4	59fa716c8c5ff8d02e3a3e543498d5b970dca51eb8a39819231dd62912a2005e	XNfp47glR6RMqBBWKdRKbWA7VSxu41Xc

可以看出，对应于 users 表中的每一个用户，在 user_auth 表中都存放了其密码加盐并迭代 100 次后生成的密文 passwd，以及每个用户随机且唯一的盐值 salt。