# 功能一的设计文档——用户验证登录功能设计报告

## 1. 项目背景

本项目旨在开发一个基于Web的应用程序，使用Python语言和Django框架实现后端开发。该应用程序提供用户注册、登录、登出以及主页访问的功能，同时包括药物管理功能，如添加、修改、删除和查询药品信息。

## 2. 需求分析

### 2.1 用户角色和权限

* 普通用户：能够注册账号、登录、登出以及访问主页。
* 管理员：除了普通用户的功能外，还能进行药品信息的管理，以及用户账号的管理。

### 2.2 功能需求

* 支持现代浏览器（如Chrome、Firefox、Safari）。
* 支持跨平台操作（Windows、macOS、Linux）。
* 使用SQLite数据库进行开发。

## 3. 系统设计

### 3.1 架构设计

* 前端：使用HTML、CSS和JavaScript构建用户界面。
* 后端：使用Django框架进行后端逻辑处理。
* 数据库：使用SQLite数据库存储用户信息和药品信息。

### 3.2 数据库设计

用户数据存储在`User`模型中，包含以下字段：

* email：用户邮箱，作为唯一标识。
* is\_active：用户账号是否激活。
* is\_admin：用户是否为管理员。

### 3.3 接口设计

* 注册接口：`POST /register`，接收邮箱和密码，创建新用户。
* 登录接口：`POST /login`，接收邮箱和密码，验证用户身份。
* 登出接口：`POST /logout`，清除用户会话，实现登出。

### 3.4 安全性设计

* 使用Django的`authenticate`和`login`方法进行用户认证。
* 密码加密存储，使用Django的密码哈希功能。
* 使用会话管理用户登录状态。

## 4. 功能实现

### 4.1 前端实现

* 使用HTML表单收集用户输入。
* 使用CSS进行样式设计，提供友好的用户界面。
* 使用JavaScript处理表单提交和用户交互。

### 4.2 后端实现

* 注册：`register`函数处理用户注册，使用`User.objects.create\_user`创建用户。
* 登录：`user\_login`函数处理用户登录，使用`authenticate`和`login`进行身份验证。
* 登出：`user\_logout`函数处理用户登出，清除会话信息。

### 4.3 数据库交互

使用Django ORM进行数据库操作，包括用户创建、验证和查询。

## 5. 测试计划

* 单元测试：对注册、登录、登出功能进行单元测试。
* 集成测试：测试前端和后端的集成，确保数据正确流转。
* 性能测试：测试系统在高并发情况下的表现。

## 6. 部署和维护

* 使用Docker容器化部署，确保环境一致性。
* 定期更新依赖库，修复安全漏洞。

## 7. 用户文档和帮助

* 提供用户手册，说明如何注册、登录和登出。
* 提供在线帮助和FAQ，解决用户常见问题。

## 8. 风险评估

* 安全风险：通过加密和安全认证机制降低风险。
* 性能风险：通过性能测试和优化确保系统稳定运行。

## 9. 结语

感谢老师百忙之中阅读本报告。我们小组使用Python和Django框架实现了一个具有用户验证登录和药物管理功能的Web应用程序。我们希望这个系统能够满足教学和实践的需求。

## 10. 附录

* 代码片段：包括注册、登录、登出功能的代码实现。
* 配置文件：Django项目的配置文件。
* 架构图：系统的架构图，展示各组件之间的关系。

## 11. 参考文献

* Django官方文档
* HTML和CSS参考手册

## 附录

### A.源码

#### 1.路由

|  |
| --- |
| #一级路由  urlpatterns = [  path('admin/', admin.site.urls),  # 凡是 url 以 sales/ 开头的，  # 都根据 sales.urls 里面的 子路由表进行路由  path('sales/', include('sales.urls')),  # 凡是 url 以 api/mgr 开头的，  # 都根据 mgr.urls 里面的 子路由表进行路由  path('api/mgr/', include('mgr.urls')),  path('accounts/', include('accounts.urls')),  #二级路由  urlpatterns = [  path('register/', views.register, name='register'),  path('login/', views.user\_login, name='login'),  path('home/', views.home\_view, name='home'),  path('logout/', views.user\_logout, name='logout'), |

#### 功能一：登入登出

|  |
| --- |
| # 注册方法  def register(request):  if request.method == 'POST':  email = request.POST.get('email')  password = request.POST.get('password')  if email and password:  try:  User.objects.create\_user(email=email, password=password)  messages.success(request, 'Registration successful!')  return redirect('login')  except ValueError as e:  messages.error(request, str(e))  else:  messages.error(request, 'Email and password are required.')  return render(request, 'accounts/register.html')    # 登录方法  def user\_login(request):  if request.method == 'POST':  email = request.POST.get('email')  password = request.POST.get('password')  if email and password:  user = authenticate(request, email=email, password=password)  if user:  login(request, user)  messages.success(request, '登录成功！')  request.session['is\_logged\_in'] = True  return redirect('home')  else:  messages.error(request, '用户名或密码不正确。')  else:  messages.error(request, 'Email and password are required.')  return render(request, 'accounts/login.html')    # 登出方法  def user\_logout(request):  if request.session.get('is\_logged\_in'):  del request.session['is\_logged\_in']  return redirect('login')    # home 页面（包含 session 验证）  @login\_required  def home\_view(request):  #登录验证  if not request.session.get('is\_logged\_in'):  return redirect('login')  user = request.user  drugs = Drug.objects.all()    # 处理查询请求  search\_query = request.GET.get('search')  if search\_query:  drugs = drugs.filter(drug\_name\_\_icontains=search\_query)    return render(request, 'accounts/home.html', {'user': user, 'drugs': drugs})      这部分数据库创建的代码：  class User(AbstractBaseUser):  email = models.EmailField(unique=True)  is\_active = models.BooleanField(default=True)  is\_admin = models.BooleanField(default=False)  USERNAME\_FIELD = 'email'  REQUIRED\_FIELDS = []  objects = UserManager()  def \_\_str\_\_(self):  return self.email  @property  def is\_staff(self):  return self.is\_admin |

### 运行截图

|  |
| --- |
| ae8b74d0f9870ce462c2405886f7ee2 |
| 573b979bd54760dce697cac38f7e205 |
| 0911283956a32db01c786b6c31e8ee0 |