浙江大学

本科实验报告

课程名称：B/S体系软件设计

姓　　名：周鹏博

学　　院：计算机科学与技术学院

系：信息安全系

专　　业：信息安全

学　　号：3230106024

指导教师：胡晓军

2026年1月5日

1. 实验概述

1.1 实验目的

本实验旨在设计并实现一个功能完善的图片管理网站，通过实践掌握现代 Web 全栈开发技术，包括前端框架 Vue 3、后端框架 Django、异步任务处理、AI 集成等技术。

1.2 实验环境

项目	配置
操作系统	Linux (Ubuntu/WSL2)
前端框架	Vue 3 + Vite + Element Plus
后端框架	Django 3.2 + Django REST Framework
数据库	SQLite 3
异步任务	Celery + Redis
AI 服务	SiliconFlow API (deepseek-vl2)

---

2. 功能完成情况

2.1 基本功能（11 项）

序号	功能
1	用户注册/登录（用户名、密码、邮箱验证）
2	PC/手机浏览器上传图片
3	EXIF 自动解析与标签生成
4	自定义标签管理
5	缩略图自动生成
6	数据库持久化存储
7	多条件查询检索
8	友好展示界面（轮播、画廊）
9	图片编辑（裁剪、旋转、调色）
10	删除功能（单个/批量）
11	移动端适配

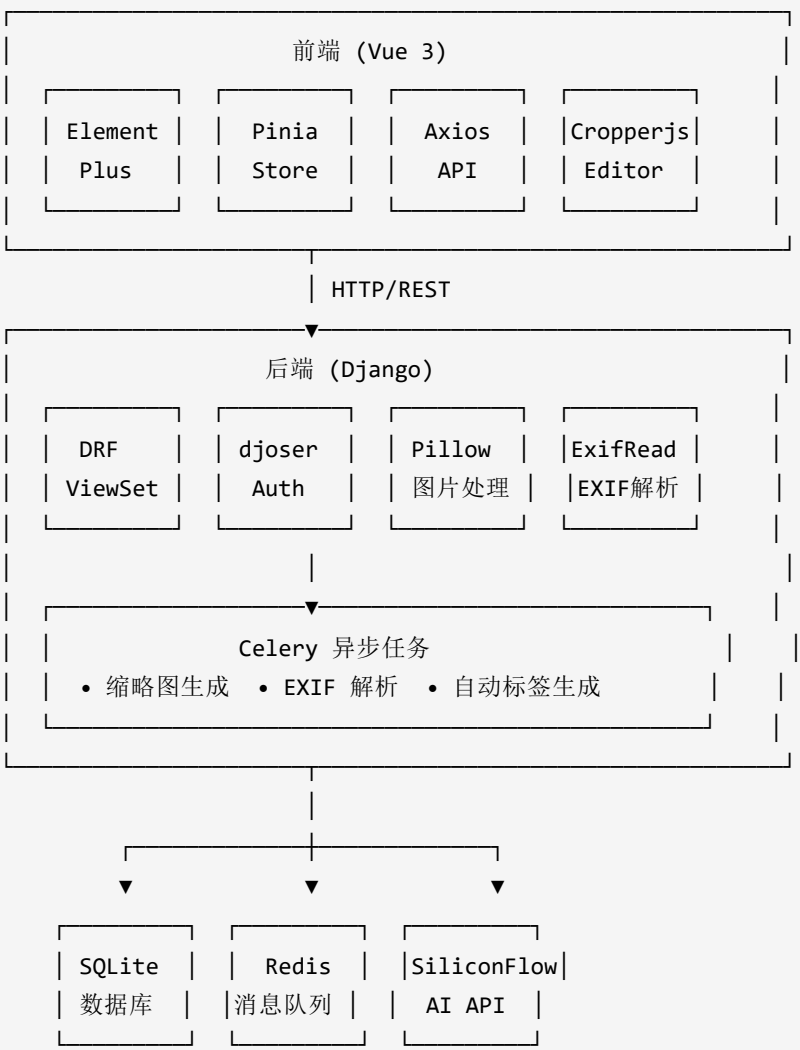
2.2 增强功能

序号	功能
1	AI 模型分析图片生成描述

---

3. 系统架构

3.1 整体架构图



3.2 技术栈说明

层级	技术	作用
前端框架	Vue 3 + Composition API	构建响应式用户界面
UI 组件	Element Plus	提供丰富的 UI 组件
状态管理	Pinia	管理全局状态

层级	技术	作用
HTTP 客户端	Axios	API 请求封装
后端框架	Django + DRF	RESTful API 开发
用户认证	djoser	用户注册、登录、Token 管理
图片处理	Pillow	缩略图生成、图片编辑
EXIF 解析	ExifRead	提取图片元数据
异步任务	Celery + Redis	处理耗时操作
AI 服务	SiliconFlow	图片描述生成

3.3 项目代码结构

```
photo-manager/
├── backend/                          # Django 后端
│   ├── manage.py                     # Django 管理脚本
│   ├── requirements.txt              # Python 依赖列表
│   ├── Dockerfile                    # 后端 Docker 镜像配置
│   ├── config/                       # 项目配置目录
│   │   ├── settings.py               # Django 全局配置
│   │   ├── urls.py                   # 根 URL 路由配置
│   │   ├── celery.py                 # Celery 异步任务配置
│   │   └── wsgi.py                   # WSGI 部署入口
│   └── apps/                         # 应用模块目录
│       ├── users/                    # 用户管理模块
│       │   ├── models.py             # 用户模型（UserProfile）
│       │   ├── views.py              # 用户视图（注册、登录、资料）
│       │   └── serializers.py        # 用户序列化器
│       ├── images/                   # 图片管理模块
│       │   ├── models.py             # 图片模型（Image）
│       │   ├── views.py              # 图片视图（CRUD、AI生成）
│       │   ├── serializers.py        # 图片序列化器
│       │   ├── tasks.py              # Celery 异步任务
│       │   ├── exif_utils.py         # EXIF 解析工具
│       │   └── image_editor.py       # 图片编辑工具
│       └── tags/                     # 标签管理模块
│           ├── models.py             # 标签模型（Tag）
│           ├── views.py              # 标签视图
│           └── serializers.py        # 标签序列化器
│
├── frontend/                         # Vue 3 前端
│   ├── index.html                    # 入口 HTML
│   ├── package.json                  # Node.js 依赖配置
│   ├── vite.config.js                # Vite 构建配置
│   ├── Dockerfile                    # 前端 Docker 镜像配置
│   ├── nginx.conf                    # Nginx 静态文件服务配置
│   └── src/                          # 源代码目录
│       ├── main.js                   # Vue 应用入口
│       ├── App.vue                   # 根组件（布局、导航）
│       ├── router/
│       │   └── index.js              # Vue Router 路由配置
│       ├── store/
│       │   └── userStore.js          # Pinia 用户状态管理
│       ├── utils/                    # 工具函数
│       │   ├── api.js                # Axios 实例配置
│       │   ├── authApi.js            # 认证相关 API
│       │   ├── imageApi.js           # 图片相关 API
│       │   ├── tagApi.js             # 标签相关 API
│       │   └── adminApi.js           # 管理员 API
│       ├── views/                    # 页面组件
│       │   ├── Home.vue              # 首页（轮播展示）
│       │   ├── Gallery.vue           # 图片库（公共画廊）
│       │   ├── MyImages.vue          # 我的图片（个人管理）
```

```
│       │   ├── Upload.vue            # 图片上传页面
│       │   ├── ImageDetail.vue       # 图片详情页
│       │   ├── Login.vue             # 登录页面
│       │   ├── Register.vue          # 注册页面
│       │   ├── Profile.vue           # 个人资料页
│       │   └── Admin.vue             # 管理员后台
│       ├── components/
│       │   └── ImageEditor.vue       # 图片编辑组件
│       └── assets/
│           └── mobile.css            # 移动端适配样式
│
├── docker/                           # Docker 配置
│   └── nginx/
│       └── nginx.conf                # 生产环境反向代理配置
│
├── docs/                             # 项目文档
│   ├── 设计文档.md
│   ├── 使用手册.md
│   ├── 项目搭建.md
│   └── 实验报告.md
│
├── docker-compose.yml                # Docker Compose 编排配置
├── start.sh                          # 本地开发启动脚本
└── .gitignore                        # Git 忽略规则
```

3.4 模块功能分析

后端模块

模块	文件	功能说明
配置模块	`config/`	
	`settings.py`	Django 全局配置：数据库、中间件、CORS、REST Framework、Celery 等
	`urls.py`	根 URL 路由，整合各 app 的路由并注册 djoser 认证路由
	`celery.py`	Celery 应用配置，连接 Redis 消息队列，自动发现异步任务
	`wsgi.py`	WSGI 部署入口，供 Gunicorn 等服务器调用
用户模块	`apps/users/`	
	`models.py`	`UserProfile` 模型：扩展 Django User，添加头像、加密存储的 API Key
	`views.py`	`UserProfileViewSet`：个人资料 CRUD、头像上传、API Key 设置
	`serializers.py`	用户数据序列化：密码加密处理、API Key 解密返回（脱敏）
	`admin.py`	Django Admin 后台注册用户模型
图片模块	`apps/images/`	

模块	文件	功能说明
	`models.py`	`Image` 模型：图片文件、缩略图、EXIF 字段（相机、时间、GPS）、处理状态
	`views.py`	`ImageViewSet`：图片上传、列表查询、详情、编辑、删除、批量操作、AI 生成描述
	`serializers.py`	图片数据序列化：嵌套标签、格式化 EXIF、计算文件大小
	`tasks.py`	Celery 异步任务：`process_image` 处理缩略图生成、EXIF 解析、自动标签
	`exif_utils.py`	`extract_exif()` 函数：使用 ExifRead 库解析图片 EXIF 元数据
	`image_editor.py`	图片编辑函数：`crop_image`、`rotate_image`、`adjust_image`（亮度/对比度）
	`urls.py`	图片模块路由配置，注册 ViewSet 到 Router
标签模块	`apps/tags/`	
	`models.py`	`Tag` 模型：标签名称、类型（系统/用户）、关联图片（多对多）
	`views.py`	`TagViewSet`：标签 CRUD、热门标签统计、按标签筛选图片
	`serializers.py`	标签数据序列化：包含关联图片数量统计
	`urls.py`	标签模块路由配置

前端模块

模块	文件	功能说明
入口配置		
	`main.js`	Vue 应用入口：挂载 Pinia、Router、Element Plus
	`App.vue`	根组件：响应式布局、顶部导航栏、移动端抽屉菜单
	`vite.config.js`	Vite 配置：开发服务器代理、构建优化
路由模块	`router/`	
	`index.js`	定义所有页面路由、配置导航守卫（登录检查、权限控制）
状态管理	`store/`	
	`userStore.js`	Pinia Store：用户登录状态、Token 持久化、用户信息缓存
API 封装	`utils/`	
	`api.js`	Axios 实例：baseURL 配置、请求拦截器（添加 Token）、响应拦截器（错误处理）
	`authApi.js`	认证 API：`login`、`register`、`logout`、`getCurrentUser`

模块	文件	功能说明
	`imageApi.js`	图片 API：`uploadImage`、`getImages`、`updateImage`、`deleteImage`、`generateAI`
	`tagApi.js`	标签 API：`getTags`、`createTag`、`getPopularTags`
	`adminApi.js`	管理 API：`getUsers`、`getAllImages`、`deleteUser`
页面组件	`views/`	
	`Home.vue`	首页：图片轮播展示、最新上传、热门标签
	`Gallery.vue`	图片库：公共图片画廊、瀑布流/网格布局、标签筛选
	`MyImages.vue`	我的图片：个人图片管理、列表/网格视图切换、批量选择删除
	`Upload.vue`	上传页面：拖拽上传、多文件上传、上传进度、格式校验
	`ImageDetail.vue`	图片详情：大图预览、EXIF 信息展示、标签编辑、AI 描述生成
	`Login.vue`	登录页面：表单验证、记住登录、跳转注册
	`Register.vue`	注册页面：用户名/邮箱/密码验证、注册成功跳转
	`Profile.vue`	个人资料：头像上传、密码修改、AI API Key 配置
	`Admin.vue`	管理后台：用户管理、图片审核、系统统计
功能组件	`components/`	
	`ImageEditor.vue`	图片编辑器：基于 Cropper.js，支持裁剪、旋转、翻转、亮度/对比度调节
样式资源	`assets/`	
	`mobile.css`	移动端适配样式：响应式断点、触摸优化、汉堡菜单

3.5 项目功能数据流

以下针对项目中实现的主要功能，展示其数据流：

基本功能数据流

功能 1：用户注册/登录

```
用户输入                前端处理                          后端处理
──────────────────────────────────────────────────────────────────

填写表单  →  Register.vue 验证  →  authApi.register()
                                          ↓
                                  POST /api/auth/users/
                                          ↓
                                  djoser 创建用户 → 返回用户信息
                                          ↓
登录表单  →  Login.vue 验证     →  authApi.login()
                                          ↓
                                  POST /api/auth/token/login/
                                          ↓
                                  验证成功 → 返回 Token → userStore 存储
```

功能 2：PC/手机上传图片

```
选择文件                前端处理                          后端处理
──────────────────────────────────────────────────────────────────

拖拽/选择  →  Upload.vue 校验格式大小  →  imageApi.uploadImage()
                                                  ↓
                                          POST /api/images/ (multipart)
                                                  ↓
                                          ImageViewSet.create() 保存文件
                                                  ↓
                                          process_image.delay() 异步处理
```

功能 3：EXIF 自动解析与标签生成

```
图片上传后              Celery 异步任务
──────────────────────────────────────────────────────────────────

process_image.delay(image_id)
        ↓
exif_utils.extract_exif(image_path)  →  解析相机型号、拍摄时间、GPS
        ↓
Image.objects.update(exif_camera_make=..., exif_datetime=...)
        ↓
根据 EXIF 生成标签  →  Tag.objects.get_or_create(name="Canon")
```

功能 4：自定义标签管理

```
用户操作                前端处理                          后端处理
──────────────────────────────────────────────────────────────
添加标签  →  ImageDetail.vue  →  tagApi.createTag()
                                        ↓
                                POST /api/tags/ {name, image_id}
                                        ↓
                                TagViewSet.create() → Tag 关联 Image
                                        ↓
删除标签  →  点击标签 ×      →  tagApi.removeTag()
                                        ↓
                                DELETE /api/images/{id}/tags/{tag_id}/
```

功能 5：缩略图自动生成

```
图片上传后              Celery 异步任务
──────────────────────────────────────────────────────────────
process_image.delay(image_id)
        ↓
Pillow 打开原图  →  thumbnail((300, 300))  →  保存到 thumbnails/
        ↓
Image.objects.update(thumbnail=thumbnail_path)
        ↓
前端列表使用 thumbnail 字段显示缩略图
```

功能 6：数据库持久化存储

```
数据操作                Django ORM                    SQLite
──────────────────────────────────────────────────────────────
Image.objects.create(...)  →  INSERT INTO images_image ...
Image.objects.filter(...)  →  SELECT * FROM images_image WHERE ...
image.save()               →  UPDATE images_image SET ...
image.delete()             →  DELETE FROM images_image WHERE id=...
```

功能 7：多条件查询检索

```
用户筛选                前端处理                          后端处理
──────────────────────────────────────────────────────────────
选择标签/日期  →  MyImages.vue 构建查询参数
                        ↓
              imageApi.getImages({tag: 'nature', date: '2026-01'})
                        ↓
              GET /api/images/?tag=nature&date=2026-01
                        ↓
              ImageViewSet.list() → filter(tags__name=tag, created__month=1)
```

功能 8：友好展示界面（轮播、画廊）

```
页面加载                前端处理                          后端处理
──────────────────────────────────────────────────────────────
Home.vue mounted  →  imageApi.getLatestImages()
                          ↓
               GET /api/images/?ordering=-created_at&limit=10
                          ↓
               el-carousel 组件渲染轮播图
                          ↓
Gallery.vue       →  瀑布流布局 + el-image 懒加载
```

功能 9：图片编辑（裁剪、旋转、调色）

```
用户操作                前端处理                          后端处理
──────────────────────────────────────────────────────────────
打开编辑器  →  ImageEditor.vue 初始化 Cropper.js
                          ↓
裁剪/旋转   →  cropper.getCroppedCanvas() 获取编辑后数据
                          ↓
保存编辑   →  imageApi.editImage(id, {action: 'crop', data: ...})
                          ↓
               POST /api/images/{id}/edit/
                          ↓
               image_editor.crop_image() → Pillow 处理 → 保存新文件
```

功能 10：删除功能（单个/批量）

```
用户操作                前端处理                          后端处理
──────────────────────────────────────────────────────────────
单个删除   →  ImageDetail.vue  →  imageApi.deleteImage(id)
                                          ↓
                                DELETE /api/images/{id}/
                                          ↓
批量删除   →  MyImages.vue 选择多张  →  imageApi.batchDelete(ids)
                                          ↓
                                POST /api/images/batch_delete/ {ids: [...]}
                                          ↓
                                Image.objects.filter(id__in=ids).delete()
```

功能 11：移动端适配

```
设备检测                前端处理
──────────────────────────────────────────────────────────────
window.innerWidth < 768px
        ↓
App.vue 切换为移动端布局  →  隐藏桌面导航，显示汉堡菜单
        ↓
mobile.css 媒体查询生效  →  调整字体、间距、按钮大小
        ↓
el-drawer 侧边抽屉菜单  →  触摸滑动关闭
```

增强功能数据流

功能 1：AI 模型分析图片生成描述

```
用户操作                前端处理                        后端处理
──────────────────────────────────────────────────────────────
点击 AI 生成  →  ImageDetail.vue  →  imageApi.generateAI(id)
                                          ↓
                                POST /api/images/{id}/generate_ai_description/
                                          ↓
                                ImageViewSet.generate_ai_description()
                                          ↓
                                获取用户 API Key（解密）
                                          ↓
                                调用 SiliconFlow API (deepseek-vl2)
                                发送图片 Base64 + prompt
                                          ↓
                                AI 返回描述文本
                                          ↓
                                Image.objects.update(ai_description=...)
                                          ↓
                                返回描述 → 前端显示
```

---

4. 技术要点

4.1 前端技术要点

Vue 3 Composition API

相比 Options API 的优势：

- 逻辑复用更灵活，可抽取为 composable 函数
- TypeScript 支持更好
- 相关逻辑可以组织在一起

Pinia 状态管理

比 Vuex 更简洁，不需要 mutations：

```
// Pinia - 直接修改
userStore.token = newToken

// Vuex - 需要 mutation
commit('SET_TOKEN', newToken)
```

4.2 后端技术要点

Celery 异步任务

对于耗时操作使用异步处理：

```
# 同步处理 - 用户需要等待
process_image(image)  # 可能需要几秒

# 异步处理 - 立即返回
process_image.delay(image.id)  # 毫秒级返回
```

API Key 加密存储

使用 Fernet 对称加密：

```
from cryptography.fernet import Fernet

# 使用 Django SECRET_KEY 派生加密密钥
key = base64.urlsafe_b64encode(hashlib.sha256(SECRET_KEY.encode()).digest())
fernet = Fernet(key)

# 加密存储
encrypted = fernet.encrypt(api_key.encode())
```

---

5. 项目亮点

5.1 技术亮点

1. **完整的前后端分离架构**: Vue 3 + Django REST Framework
2. **异步任务处理**: Celery + Redis 处理耗时操作，提升用户体验
3. **AI 集成**: 接入视觉大模型生成图片描述
4. **安全设计**: API Key 加密存储、权限控制、文件校验

5.2 功能亮点

1. **智能 EXIF 解析**: 自动提取拍摄信息并生成标签
2. **在线图片编辑**: 支持裁剪、旋转、调色等操作
3. **批量操作**: 高效管理大量图片
4. **响应式设计**: 适配 PC、平板、手机多端

---

6. 开发过程中的问题与解决

6.1 遇到的部分 Bug

Bug	问题描述	原因分析	解决方案
#1	头像更换后页面不刷新	前端未清除缓存	添加页面刷新逻辑
#2	网格选择无勾选标记	自定义复选框样式问题	重写复选框组件样式
#3	移动端按钮图标消失	CSS 样式覆盖	使用 el-icon 组件
#4	AI 生成返回 404	action 放错 ViewSet	移动到正确的 ViewSet
#5	批量操作按钮不显示	watch 未深度监听	添加 `{ deep: true }`

6.2 部分问题分析

Bug #4: AI 生成 404 错误

问题表现: 点击 AI 生成描述按钮时，API 返回 404 错误。

原因分析: Django REST Framework 的 `@action` 装饰器会根据所在 ViewSet 生成 URL。`generate_ai_description` 方法被错误地放在了 `AdminImageViewSet` 中，导致 URL 变成了 `/api/admin/images/{id}/generate_ai_description/`，而前端请求的是 `/api/images/{id}/generate_ai_description/`。

解决方案: 将方法移动到 `ImageViewSet` 中。

经验教训: DRF 的 action 会根据所在 ViewSet 生成 URL，放错位置会导致 404。

Bug #5: 批量操作按钮不显示

问题表现: 在网格视图中选择图片后，底部的批量操作按钮没有出现。

原因分析: Vue 3 的 `watch` 默认不会深度监听数组的变化（如 push、splice）。

解决方案:

```
watch(selectedImageIds, (ids) => {
  selectedImages.value = images.value.filter(img => ids.includes(img.id))
}, { deep: true })  // 添加 deep: true
```

---

7. 文档清单

文档	说明
设计文档.md	系统架构、数据库设计、API 设计

文档	说明
使用手册.md	用户操作指南
项目搭建.md	环境配置、项目启动指南
实验报告.md	功能完成情况、问题解决、总结

---

8. 总结与收获

8.1 技术能力提升

- 掌握了 Vue 3 Composition API 的使用
- 深入理解了 Django REST Framework 的 ViewSet 和 Router 机制
- 学会了 Celery 异步任务处理
- 了解了 AI API 的集成方式
- 掌握了 Docker 容器化部署

8.2 工程能力提升

- 学会了规范的项目结构组织
- 理解了前后端分离的开发模式
- 提高了调试和解决问题的能力
- 养成了编写可维护代码的习惯