浙江大学

本科实验报告

课程名称: B/S 体系软件设计

姓　　名: Li Anjing

学　　院: 计算机科学与技术学院

　　系:

专　　业: 软件工程

学　　号: 3230104388

指导老师: 胡晓军

2025 年 11 月 21 日

目录

1 引言

1.1 编写目的

这个项目是 2025-2026 秋冬学期《B/S 体系软件设计》课程的大程，旨在开发一个成熟的图片管理网站。用户登录后，能够通过 PC 或手机浏览器上传、查询、检索图片，可以帮助用户更好地存储以及管理图片。这个网站需要提供友好的交互界面，并且要适配手机移动端，确保在手机浏览器/微信等应用内置的浏览器中都能够良好地展示。此外，项目还需要提供详尽的软件项目文档，以帮助开发者了解并掌握一整套 web 应用开发技术以及整个开发流程。

该文档是项目的中期系统设计文档，包含了系统的需求分析、总体架构设计与技术选型、数据库设计、系统接口以及界面设计等内容，介绍该系统的具体设计方案。

1.2 项目背景

在数字化时代，随着智能手机与数码相机的普及，个人与机构产生的图片数量正经历爆炸式增长。海量图片的高效存储、组织管理与快速检索，已成为用户面临的普遍性痛点。传统的本地文件夹存储方式，不仅难以跨设备访问，更在图片分类与搜索上效率低下，大量珍贵影像资料因此被埋没。

与此同时，Web 技术、云计算与人工智能的迅猛发展，为在线图片管理提供了全新的解决方案。通过利用浏览器即可访问的 B/S 架构应用，用户能够突破设备与地域的限制，随时随地管理个人图库。现代技术如 EXIF 信息自动提取、AI 图像识别等，更能实现图片属性的自动化标注与智能化分类，从根本上提升管理效率。

在此背景下，本项目旨在开发一个基于 B/S 架构的智能图片管理网站。它不仅是简单的网络相册，更是一个集安全存储、智能分类、多条件检索、便捷编辑与友好展示于一体的综合性平台，以期满足现代用户对高效、智能与个性化图片管理的迫切需求。

2 系统需求分析

2.1 实验要求

需要实现的基本功能如下：

1. 实现用户注册、登录功能，用户注册时需要填写必要的信息并验证，如用户名、密码要求在 6 字节以上，email 的格式验证，并保证用户名和 email 在系统中唯一，用户登录后可以进行以下操作。
2. 通过 PC 或手机浏览器将照片或其他类型的图片上传到网站进行存储。
3. 能够通过照片的 exif 信息自动创建图片分类标签及其他辅助信息，如时间、地点、图片分辨率等。
4. 可以给图片增加自定义分类标签，方便检索。
5. 生产缩略图方便后续显示。
6. 图片信息保存在数据库中，方便后续查询。
7. 提供查询界面能根据各种条件查找图片。

8. 提供友好的展示界面，如选择一定的图片进行轮播显示等。
9. 对选定的图片提供简单的编辑功能，如裁剪、修改色调等。
10. 提供删除功能。
11. 样式适配手机，开发手机 App 或能够在手机浏览器/微信等应用内置的浏览器中友好显示。

增强功能：

1. 调用 AI 模型分析图片，提供多类型的标签，如风景、人物、动物等。
2. 提供 mcp 接口，能通过大模型对话方式检索网站上的图片。

2.2 功能性需求分析

本项目的功能性需求围绕图片的全生命周期管理进行设计，从用户身份验证到图片的上传、处理、管理、检索和展示。根据实验要求，我们将核心功能划分为以下六大模块。

2.2.1 用户认证与管理模块功能需求分析

- 用户注册
  - 用户可填写用户名、密码、邮箱等信息进行注册。
  - 系统需对输入进行有效性验证：用户名和密码长度均需大于 6 字节；邮箱格式必须符合规范。
  - 系统需保证用户名和邮箱在全系统内的唯一性，注册失败需给出明确提示。
  - 验证失败时，需在界面上给出明确、具体的错误提示（如“用户名已存在”、“邮箱格式不正确”）。
  - 验证通过后，系统需将完整的注册信息存入数据库的用户表中。
  - 注册成功自动跳转至用户登录主界面。
- 用户登录
  - 已注册用户可使用用户名/邮箱和密码登录系统。
  - 系统根据输入的用户名/邮箱在数据库中查找对应用户，并验证加密后的密码是否匹配。
  - 验证失败后，系统需输出错误信息（如“用户名或密码错误”），并保持在登录界面。
  - 验证成功后，系统建立并维护用户会话，跳转至系统主界面。
- 用户信息修改
  - 用户登录后，可进入个人主页或设置页面，对个人信息进行修改，包括用户名、密码和邮箱等。
  - 修改信息时，需再次进行身份验证（如输入当前密码）并重新通过所有格式和唯一性验证规则。
  - 若修改密码，新密码同样需进行加密存储。
  - 验证通过后，将更新后的信息保存至数据库，并反馈修改成功提示。

- 退出登录
  - 用户点击“退出登录”按钮后，系统将立即清除服务器端的用户会话信息及客户端的相关凭证。
  - 退出成功后，页面自动跳转回用户登录主界面。
- 用户注销
  - 用户可在登录后，于个人主页可进行注销。
  - 执行注销操作前，系统应进行二次确认，提醒用户此操作不可逆。
  - 确认后，系统将永久删除该用户在数据库中的所有相关数据（包括用户基本信息及其上传的图片、标签等记录）。
  - 注销完成后，系统自动退出当前登录状态，并跳转至用户登录主界面。

2.2.2 图像上传与处理模块功能需求分析

- 图片上传
  - 用户可通过 PC 或手机浏览器，以表单形式上传单个或多个图片文件。
  - 系统需支持常见图片格式（如 JPEG, PNG, GIF 等）。
  - 上传的图片需与登录用户关联存储。
- EXIF 信息提取
  - 系统需自动解析上传图片的 EXIF 元数据。
  - 提取的关键信息包括：拍摄时间、拍摄地点（GPS 坐标，需转换为文字地名）、相机型号、图片分辨率等，并存入数据库。
- 缩略图生成
  - 系统需在上传后自动为原图生成一个或多个固定尺寸的缩略图，用于后续列表和画廊展示，以提升页面加载性能。
- AI 智能标签
  - 调用预训练的 AI 图像识别模型（如 CNN 模型或云端 API）对图片内容进行分析。
  - 自动生成描述性标签，如“风景”、“人物”、“动物”、“建筑”、“夜晚”等，丰富图片的检索维度。

2.2.3 图片管理与标签模块功能需求分析

- 自定义标签管理
  - 用户可为单张或批量图片添加、编辑或删除自定义文本标签。
- 图片信息储存
  - 将所有图片信息（原图/缩略图存储路径、上传时间、EXIF 信息、AI 标签、自定义标签等）结构化地存入数据库。

- 图片删除
  - 用户可删除自己上传的图片。删除操作需同时移除数据库中的记录和服务器上的物理文件（包括缩略图）。

2.2.4 图片查询与检索模块功能需求分析

- 多条件组合查询
  - 提供搜索界面，支持用户根据一个或多个条件进行查询。
- 结果展示
  - 搜索结果以缩略图列表形式呈现，并附带关键信息（如标题、主要标签）。

2.2.5 图片展示与编辑模块功能需求分析

- 友好界面展示
  - 图片详情页: 点击缩略图后可查看大图及所有详细信息（EXIF、标签等）。
  - 轮播展示: 用户可选择多张图片进入专门的轮播播放模式，实现自动或手动幻灯片放映。
- 响应式设计
  - 整个网站前端需采用响应式布局，能够自动适配 PC、手机浏览器及微信内置浏览器等不同屏幕尺寸的设备。
- 简易在线编辑
  - 在图片详情页提供简单的编辑功能，如裁剪和修改色调，编辑后可生成新版本图片保存。

2.2.6 集成接口模块功能需求分析

- MCP 接口
  - 提供基于 MCP（Model Context Protocol）的接口，允许大型语言模型（如 ChatGPT）通过自然语言对话的方式理解用户意图，并调用本系统的检索功能来查找图片。例如，用户可以说 “帮我找出上个月拍的所有包含狗的风景照”，模型可通过此接口将指令转换为系统可执行的查询条件。

2.3 非功能性需求分析

2.3.1 性能需求

- 响应时间
  - 常规页面（如登录页、查询页）加载时间应在 3 秒内。
  - 图片上传操作的响应时间（包括处理）应在 5 秒内（受网络和图片大小影响）。
  - 图片搜索、筛选等查询操作的响应时间应在 2 秒内。
- 吞吐量与并发量

  - 系统应能支持至少 50 名用户同时进行基本操作（如浏览、查询）。
  - 系统应能支持至少 20 名用户同时进行图片上传操作。

- 资源利用率
  - 在预期负载下，系统关键组件（如应用服务器、数据库服务器）的 CPU 和内存平均利用率不应超过 70%，峰值不应超过 90%，以避免系统过载。

2.3.2 可用性需求

- 易用性
  - 用户界面应简洁直观，无需专门培训即可上手使用。
  - 提供明确的操作指引和反馈（如成功/错误提示）。
  - 关键操作（如删除、注销）需有二次确认机制，防止误操作。
- 可访问性
  - 界面应遵循响应式设计，确保在主流 PC 浏览器、手机浏览器及微信内置浏览器中均能正常、友好地显示和交互。

2.3.3 可靠性需求

- 成熟度
  - 系统在发布后，核心功能（上传、存储、检索）的故障率应低于 1%。
- 可恢复度
  - 在发生一般性软件故障（如服务意外终止）后，系统应能在 5 分钟内自动或手动恢复。
  - 用户数据（如图片文件、元数据）必须定期备份，在数据丢失或损坏时，能够恢复到 24 小时内的状态。

2.3.4 安全性需求

- 数据保密性
  - 用户密码在数据库中必须加密存储（如 Bcrypt 等强哈希算法），严禁明文保存。
  - 所有用户会话（Session）需使用安全令牌管理，防止会话劫持。
- 访问控制
  - 严格实行身份验证，除注册和登录页面外，所有功能页面均需验证用户登录状态。
  - 用户只能访问、操作（查看、编辑、删除）其本人上传的图片和数据，确保数据的隔离性。
- 输入验证
  - 对所有用户输入（如表单字段、文件上传）进行严格的后端验证和过滤，防止 SQL 注入、XSS 跨站脚本等常见 Web 攻击。
- 文件上传安全
  - 必须对上传文件的类型、大小进行严格限制和检查，防止恶意文件上传。

2.3.5 可维护性需求

- 模块化
  - 系统应采用分层架构（如控制器-服务-数据访问层），代码模块化程度高，便于后续功能扩展和修改。
- 可管理性
  - 应提供清晰的日志记录功能，记录用户关键操作（登录、上传、删除）和系统错误，便于问题排查和审计。
- 数据可维护性
  - 数据库结构设计应规范，并提供完整的建库建表脚本，便于数据库的初始化、迁移和维护。

2.3.6 兼容性需求

- 浏览器兼容性
  - 系统应兼容主流版本的 Chrome, Firefox, Safari，及 Edge 浏览器。
- 移动端兼容性
  - 在 iOS 和 Android 平台的常用浏览器及微信内置浏览器中，功能与显示均需正常。

3 系统总体架构设计与技术选型

3.1 系统功能模块设计

3.1.1 用户认证与管理模块

- 用户注册子模块：处理新用户注册，包括信息验证、密码加密和数据库记录。
- 用户登录子模块：验证用户凭证，创建并管理用户会话。
- 用户信息管理子模块：允许一登陆的用户查看和修改个人信息（如用户名、密码、邮箱）。
- 会话控制子模块：处理用户的退出登录和注销流程。

3.1.2 图片上传与处理模块

- 图片接收子模块：接收来自前端的上传请求，验证文件类型和大小。
- EXIF 信息提取子模块：解析图片的元数据，提取拍摄时间、地点、设备等信息。
- 缩略图生成子模块：根据配置的尺寸，自动为原图生成缩略图。
- AI 智能标签子模块（增强功能子模块）：调用 AI 模型接口，为图片自动生成内容描述标签。

3.1.3 图片数据与管理模块

- 数据持久化子模块：将图片的元数据（路径、EXIF 信息、AI 标签等）存入数据库。
- 标签管理子模块：提供对自定义标签的增加、删除、修改、查询接口。
- 图片删除子模块：处理图片删除请求，同步一处数据库记录和服务器上的物理文件。

3.1.4 图片查询与检索模块

- 查询接口子模块：提供后端 API，接受并解析前端的查询条件（如标签、时间范围等）。
- 数据检索子模块：构建并执行数据库查询语句，返回符合条件的图片数据列表。

3.1.5 图片展示与交互模块

- 响应式布局子模块：确保网站在 PC、手机等不通设备上均有良好的显示效果。
- 图片画廊子模块：以列表、网格等形式展示图片缩略图。
- 图片详情与轮播子模块：展示单张图片大图及所有详细信息，并提供多图轮播功能。
- 在线图片编辑子模块：在浏览器端实现简单的图片编辑功能，如裁剪与色调调整。

3.1.6 系统接口模块（增强功能模块）

- MCP 服务接口子模块：实现 MCP 服务器，允许大型语言模型通过自然语言对话来检索系统内的图片。

3.2 系统总体架构设计

本项目采用基于分层模型的 B/S 架构，并结合模块化设计思想，将系统划分为表示层、应用层、数据层和外部服务层。这种设计确保了系统的可扩展性、可维护性和松耦合性。各层之间通过定义良好的 API 接口进行通信，下层为上层提供服务，上层无需关心下层的具体实现细节。

3.2.1 表示层

作为用户与系统交互的界面，负责渲染页面、接收用户输入并将请求发送至后端，同时展示后端返回的数据。

基于现代 Web 技术开发，采用响应式框架，确保在 PC 和移动端浏览器上均有良好的用户体验。该层不包含任何业务逻辑，仅通过 AJAX/Fetch API 与后端的 RESTful API 进行数据交互。

3.2.2 应用层

这是系统的核心，承载了所有的业务逻辑，我们将其部署在一个 Web 服务器上运行的后端应用中。该层内部可进一步细分为：

- 路由与控制器：接收来自前端的 HTTP 请求，进行参数解析、基本验证，并将请求路由至对应的业务逻辑模块进行处理，最后将处理结果封装成 JSON 或视图返回给前端。
- 业务逻辑模块：这是架构图中的核心模块，根据功能划分为：

  - 用户认证与管理模块：处理注册、登录、信息修改等逻辑。
  - 图片上传与处理模块：协调文件接收、EXIF 解析、缩略图生成等流程。
  - 图片数据与管理模块：负责标签管理和图片元数据的 CRUD 操作。
  - 图片查询与检索模块：执行复杂的多条件搜索逻辑。
  - 图片展示与交互模块：为前端提供图片列表、详情等数据。
- 数据访问层：封装所有对数据层的操作，使用 ORM 或自定义的 Repository 模式来访问数据库，使业务逻辑与具体的数据存储技术解耦。

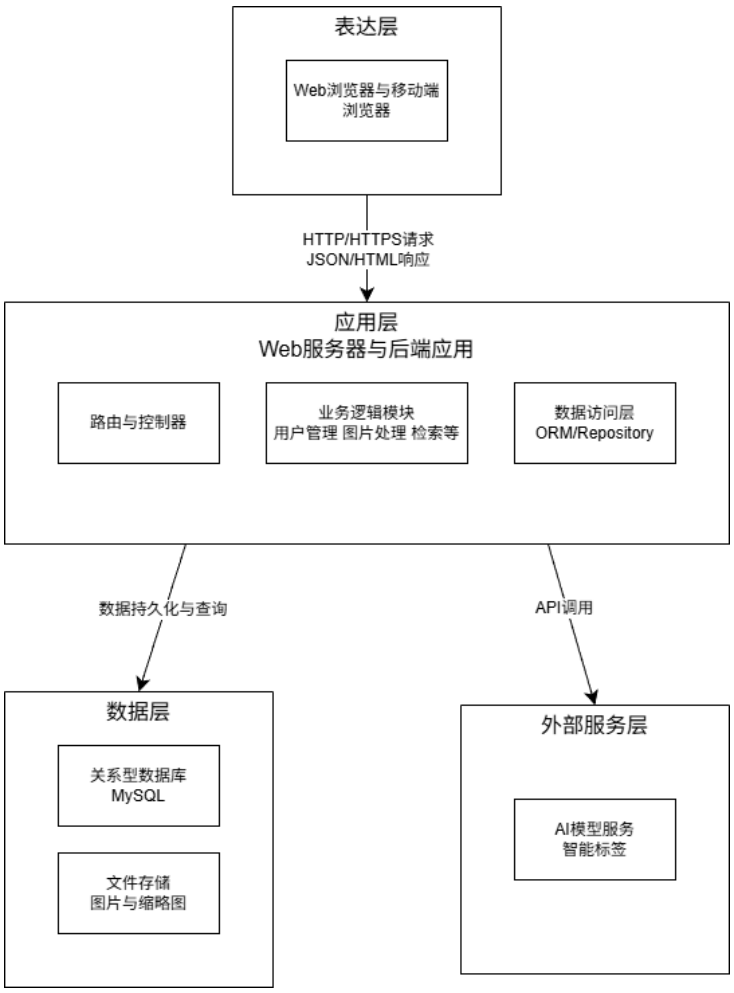
3.2.3 数据层

负责数据的持久化存储。关系型数据库用于存储结构化数据，包括用户信息、图片元数据（EXIF 信息、自定义标签、AI 标签等）；文件存储使用服务器的磁盘系统或对象存储服务，用于存储用户上传的原始图片文件和系统生成的缩略图文件。

3.2.4 外部服务层

集成第三方服务以扩展系统能力，实现增强功能。AI 模型服务通过 HTTP API 调用云端或本地部署的图像识别服务，为图片生成智能标签。

3.2.5 架构设计简单图示及流程



架构设计图如上，其流程大致为：

1. 用户在表示层进行操作（如登录、上传图片）。
2. 请求通过 HTTP/HTTPS 协议发送到应用层的对应控制器。
3. 控制器调用相应的业务逻辑模块进行处理。在处理过程中，业务模块可能通过数据访问层与数据层交互，也可能调用外部服务层。
4. 业务逻辑处理完毕后，结果经由控制器返回给表示层。
5. 表示层根据返回的数据更新界面，完成一次交互。

3.3 技术栈选择

- 前端技术栈
  - Vue 3 + Vite：Vue 3 用于构建用户界面的组件化架构，实现数据驱动视图的高效更新。Vite 提供极速的前端开发服务器和优化的生产构建，提升开发体验和页面加载性能。
  - Element Plus (PC) + Vant 4 (Mobile):Element Plus 提供丰富的桌面端 UI 组件，快速搭建图片管理后台的各类表单、列表和弹窗。Vant 4 提供专为移动端设计的交互组件，确保在手机浏览器上的原生体验和友好显示。
  - Pinia: 集中管理跨组件共享的应用程序状态，如用户登录会话、全局的图片查询条件等，保证数据流清晰可控。
- 后端技术栈
  - Java: 用于构建稳定、高性能的后端服务逻辑，处理复杂的业务规则、用户认证、数据校验和 API 路由。
  - Spring Boot: 作为项目的核心骨架，快速创建独立运行的后端应用，集成 Web 服务器、依赖注入和各种 Starter 依赖，极大简化了项目配置和部署流程。
  - MyBatis: 负责与数据库交互，MyBatis 精准映射 SQL 语句至 Java 对象，高效完成图片元数据及用户信息的持久化与复杂条件检索。
  - Maven: 统一管理项目所需的第三方库（JAR 包），并自动化完成项目的编译、测试和打包过程。
- 数据库技术栈
  - MySQL: 可靠地存储所有结构化数据，包括用户账户、图片的 EXIF 信息、自定义标签、AI 标签以及它们之间的关联关系。
- 外部服务与工具库
  - Thumbnailator: 在服务器端执行图片的缩放与裁剪操作，核心用于在图片上传后自动生成统一规格的缩略图，以供前端列表和画廊快速展示。
  - Apache Commons Imaging: 自动解析上传图片文件的元数据，精准获取拍摄时间、相机型号、GPS 定位等关键信息，为图片的自动分类和高级检索提供数据基础。
  - 云端图像识别 API: 为系统注入智能化能力，通过调用 API 对图片内容进行分析，自动生成“风景”、“人物”等描述性标签，极大丰富图片的检索维度和管理粒度。

  - BCrypt: 在用户注册和登录环节，对用户密码进行强哈希加密并验证，确保用户凭证在数据库中的存储安全，有效防止信息泄露。

4 数据处理

4.1 数据存储

4.1.1 结构化数据

本项目使用 MySQL 数据库存储数据，进行数据的管理。具体数据表设计见4.3。

4.1.2 非结构化数据

图片文件本身作为二进制大对象，采用文件系统进行存储。

- 存储策略：在服务器上建立专门的存储目录，并根据上传日期或用户 ID 生成子目录进行组织，避免单目录文件过多。
- 路径映射：数据库中仅存储图片和缩略图的相对路径或文件名。当需要访问图片时，后端应用根据路径规则定位文件并提供给前端。

这种存储方式管理简单、访问直接，并且与数据库分离，降低了数据库的负载，同时便于未来扩展至对象存储服务。

4.2 数据安全

本系统的设计从多个层面保障用户数据的机密性、完整性和可用性。

4.2.1 敏感信息加密

- 用户密码：使用 BCrypt 强哈希算法进行单向加密存储。即使数据库泄露，攻击者也无法直接获取用户明文密码。
- 传输加密：全程使用 HTTPS（TLS）协议对浏览器与服务器之间的所有通信数据进行加密，防止数据在传输过程中被窃听或篡改。

4.2.2 访问控制与权限验证

- 身份认证：使用基于 Token 或 Session 的机制来管理用户登录状态。任何非公开 API 请求都必须携带有效的认证令牌。
- 权限校验：在执行所有数据操作前，后端均会进行严格的权限验证。例如，在删除图片前，会校验当前登录用户的 ID 是否与该图片的上传者 ID 匹配，确保用户只能操作自己创建的数据，实现严格的数据隔离。

4.2.3 输入验证与攻击防护

- SQL 注入防护：通过使用 MyBatis 的 #{} 参数绑定方式，从根本上杜绝 SQL 注入攻击。
- XSS 跨站脚本防护：对用户提交的自定义标签等内容进行严格的输入过滤和转义处理，并在前端渲染时进行相应处理，防止恶意脚本的执行。

4.2.4 文件上传安全

- 类型白名单验证：在后端严格校验上传文件的类型格式，只允许 jpg, png 等安全的图片格式。
- 大小限制：对上传文件的大小进行限制，防止磁盘空间耗尽攻击。
- 文件重命名：对上传的文件进行随机化重命名，避免因文件名冲突或被恶意猜测路径而导致的安全风险。

4.2.5 数据备份与可恢复性

制定定期备份策略，同时对 MySQL 数据库和上传文件目录进行备份，确保在发生系统故障或数据误删时，能够将数据恢复到最近的备份点，保障数据的可用性。

4.3 数据库设计与建表

本项目采用关系型数据库模型，共包含 5 张核心数据表。这些表围绕“用户”、“图片”和“标签”三个核心实体构建，通过外键约束建立关联，形成一个完整的数据闭环，其 E-R 核心思想如下：

- 用户与图片是一对多关系：一个用户可以上传多张图片，但一张图片只属于一个用户。
- 图片与标签是多对多关系：一张图片可以拥有多个标签，一个标签也可以被标记在多张图片上。

为实现这种多对多关系，引入了图片-标签关联表作为中间表。此外，为满足范式要求并将动态元数据与核心信息分离，专门设立了图片元数据表，与图片表形成一对一关系。

4.3.1 用户表（Users）

字段名	类型	约束	描述
user_id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	系统内部唯一标识用户的 ID，自增。
username	VARCHAR(50)	UNIQUE,NOT NULL	用于用户登录和前端显示，系统强制保证唯一性。
email	VARCHAR(100)	UNIQUE,NOT NULL	可作为替代登录名，并用于接收系统通知，强制唯一。
password_hash	VARCHAR(255)	NOT NULL	存储经过 BCrypt 算法加密后的密码密文，确保即使数据库泄露，明文密码也不会被盗取。
create_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	记录用户注册的具体时间，用于审计和分析。

表 1: 用户表（Users）

4.3.2 图片表 (Images)

字段名	类型	约束	描述
image_id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	系统内部唯一标识用户的 ID，自增。
user_id	BIGINT	FOREIGN KEY,NOT NULL	标识此图片的上传者，直接关联到 Users 表。这是实现数据隔离的关键，确保用户只能访问自己的图片。
origin_filename	VARCHAR(255)	NOT NULL	用户上传时文件的原始名称，用于下载或友好显示。
storage_path	VARCHAR(500)	NOT NULL	在服务器文件系统或对象存储中的路径，用于定位和访问原图文件。
thumbnail_path	VARCHAR(500)	NOT NULL	系统生成的缩略图文件的路径，用于列表、画廊等需要快速加载的场景。
file_size	BIGINT		文件大小。以字节为单位，用于管理和统计。
uploaded_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	记录图片进入系统的时间。

表 2: 图片表（Images）

4.3.3 标签表（Tags）

字段名	类型	约束	描述
tag_id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	主键。
tag_name	VARCHAR(50)	NOT NULL	标签内容。
tag_type	ENUM('system', 'ai', 'custom')	NOT NULL	标签类型。用于区分标签来源：system（系统从 EXIF 提取，如相机型号）、ai（AI 模型自动生成）、custom（用户手动添加）。

表 3: 标签表（Tags）

4.3.4 图片元数据表（Image_Metadata）

字段名	类型	约束	描述
id	BIGINT	PRIMARY KEY, AUTO_-INCREMENT	主键。
image_id	BIGINT	FOREIGN KEY,UNIQUE,NOT NULL	与 Images 表形成一对一关系，确保每张图片只有一份元数据记录。
taken_time	DTAETIME		拍摄时间。从 EXIF 中提取，是按时间检索和分类的重要依据。
camera_model	VARCHAR(100)		相机/设备型号。
gps_latitude	DOUBLE		GPS 纬度。用于实现按地理位置检索和在地图上展示。
gps_longitude	DOUBLE		GPS 经度。用于实现按地理位置检索和在地图上展示。
width	INT		图片宽度（像素）。用于按分辨率筛选。
height	INT		图片高度（像素）。用于按分辨率筛选。

表 4: 图片元数据表（Image_Metadata）

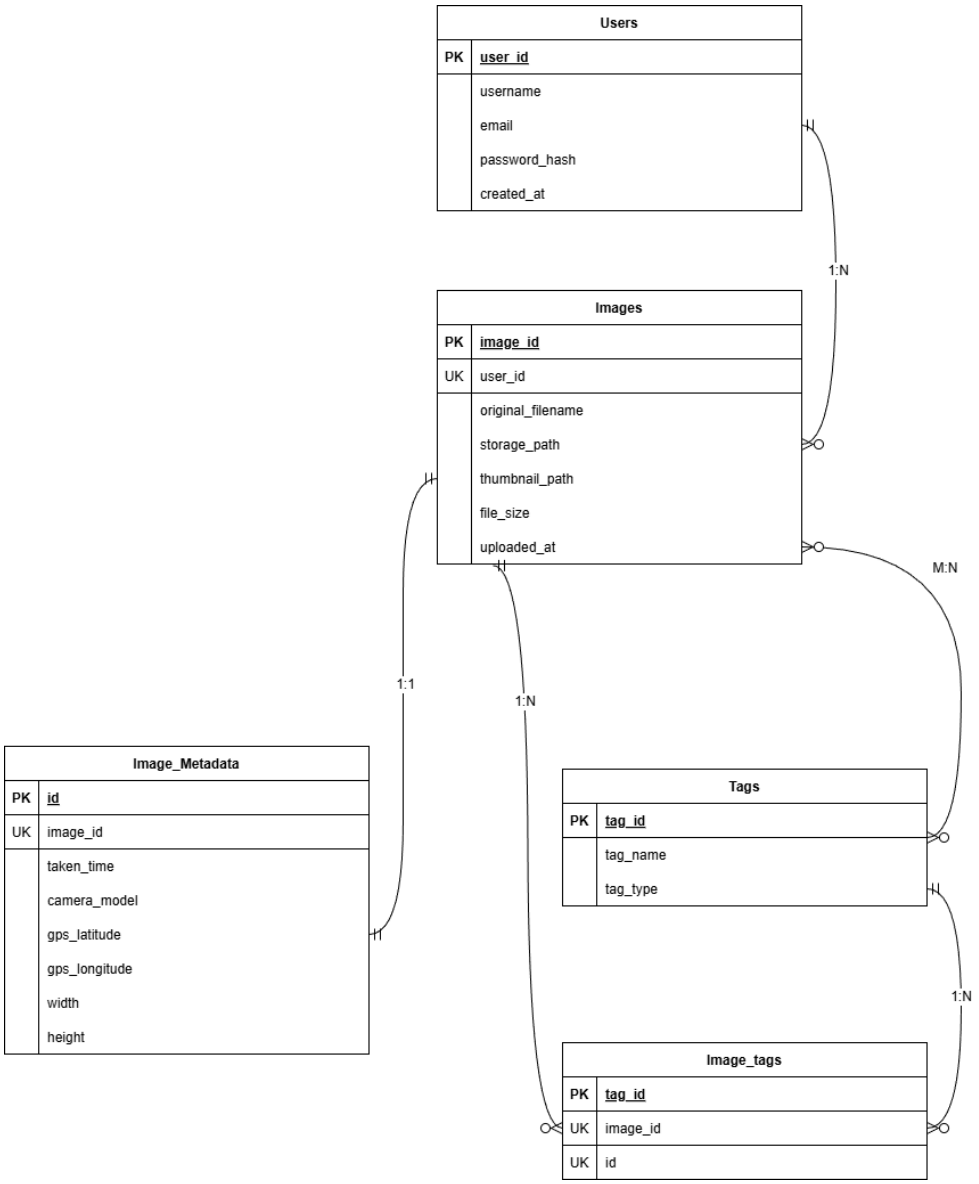
4.3.5 图片-标签关联表（Image_tags）

字段名	类型	约束	描述
id	BIGINT	PRIMARY KEY, AUTO_-INCREMENT	主键。
image_id	BIGINT	FOREIGN KEY,NOT NULL	外键，关联到图片。
tag_id	BIGINT	FOREIGN KEY,NOT NULL	外键，关联到标签。
		UNIQUE(image_id, tag_-id)	联合唯一约束。防止同一张图片被重复标记相同的标签，保证数据的完整性。

表 5: 图片-标签关联表（Image_tags）

4.3.6 E-R 图

该项目各表之间关系的 E-R 图如下：



5 系统接口设计

5.1 用户认证接口

- 用户注册
  - 函数名：UserRegister
  - POST
  - 功能说明：创建新用户账户
  - 主要输入参数：用户名、邮箱、密码
  - 返回数据：用户 ID、用户名
  - 数据修改：在 Users 中插入新用户记录
- 用户登录
  - 函数名：UserLogin

  - 请求方式：POST
  - 功能说明：验证用户身份并创建会话
  - 主要输入参数：用户名/邮箱、密码
  - 返回数据：访问令牌、用户基本信息
  - 数据修改：创建用户会话

- 用户退出
  - 函数名：UserLogout
  - POST
  - 功能说明：清楚用户登录状态
  - 主要输入参数：无
  - 返回数据：操作结果
  - 数据修改：清除用户会话信息

- 更新用户信息
  - 函数名：UpdateUserInfo
  - 请求方式:PUT
  - 功能说明：修改用户个人信息
  - 主要输入参数：用户名、邮箱等可修改信息
  - 返回数据：更新后的用户信息
  - 数据修改：更新 Users 表中的对应用户记录

5.2 图片上传与管理接口

- 图片上传
  - 函数名：UploadImages
  - 请求方式：POST
  - 功能说明：接收并存储用户上传的图片文件
  - 主要输入参数：图片文件（表单数据）
  - 返回数据：图片 ID、存储路径、文件信息
  - 数据修改：在 Images 表插入记录，创建文件，提取元数据

- 获取图片列表
  - 函数名：GetImagesList
  - 请求方式：GET
  - 功能说明：分页获取当前用户的图片列表
  - 主要输入参数：页码、页大小、排序方式
  - 返回数据：图片列表、分页信息

- 获取图片详情
  - 函数名：GetImageDetail
  - 请求方式：GET
  - 功能说明：获取指定图片的详细信息
  - 主要输入参数：图片 ID
  - 返回数据：图片详情、元数据、标签列表
- 删除图片
  - 函数名：DeleteImage
  - 请求方式：DELETE
  - 功能说明：永久删除指定图片及相关数据
  - 主要输入参数：图片 ID
  - 返回数据：操作结果
  - 数据修改：删除图片记录、元数据、标签关联、物理文件

5.3 标签管理接口

- 为图片添加标签
  - 函数名：AddImageTags
  - 请求方式：POST
  - 功能说明：为指定图片添加自定义标签
  - 主要输入参数：图片 ID、标签名称列表
  - 返回数据：操作结果
  - 数据修改：创建标签记录，建立图片标签关联
- 移除图片标签
  - 函数名:RemoveImageTag
  - 请求方式：DELETE
  - 功能说明：移除图片的指定标签
  - 主要输入参数：图片 ID、标签 ID
  - 返回数据：操作结果
  - 数据修改：删除图片标签关联记录
- 获取用户标签库
  - 函数名：GetUserTags
  - 请求方式：GET
  - 功能说明：获取用户使用的所有标签
  - 主要输入参数：无
  - 返回数据：标签列表

5.4 图片查询与检索接口

- 多条件搜索图片
  - 函数名：SearchImages
  - 请求方式：POST
  - 功能说明：根据复杂条件组合查询图片
  - 主要输入参数：关键词、时间范围、标签、分辨率等
  - 返回数据：匹配的图片列表、分页信息
- 高级搜索建议
  - 函数名：GetSearchSuggestions
  - 请求方式：GET
  - 功能说明：根据输入提供搜索建议
  - 主要输入参数：关键词前缀
  - 返回数据：标签建议、分类建议

5.5 图片展示与编辑接口

- 获取轮播图片集
  - 函数名：GetSlideshowImages
  - 请求方式：POST
  - 功能说明：获取指定图片集的轮播数据
  - 主要输入参数：图片 ID 列表
  - 返回数据：轮播图片信息
- 图片裁剪编辑
  - 函数名：CropImage
  - 请求方式：POST
  - 功能说明：对图片进行裁剪操作
  - 主要输入参数：图片 ID、裁剪区域参数
  - 返回数据：新图片 ID、处理结果
  - 数据修改：生成新的图片文件及记录
- 调整图片色调
  - 函数名：AdjustImageTone
  - 请求方式：POST
  - 功能说明；调整图片亮度、对比度等色调参数
  - 主要输入参数：图片 ID、色调调整参数
  - 返回数据：新图片 ID、处理结果
  - 数据修改：生成新的图片文件及记录

5.6 增强功能接口

- AI 智能标签生成
  - 函数名：GenerateAITags
  - 请求方式：POST
  - 功能说明：调用 AI 服务分析图片内容生成标签
  - 主要生成参数：图片 ID
  - 返回数据:AI 生成的标签列表
  - 数据修改：创建 AI 标签记录及关联
- MCP 图片检索
  - 函数名：MCPImageSearch
  - 请求方式：POST
  - 功能说明：通过自然语言理解检索图片
  - 主要输入参数：自然语言查询语句
  - 返回数据：匹配图片列表、检索推理说明
- 批量操作接口
  - 函数名：BatchOperation
  - 请求方式：POST
  - 功能说明：对多张图片执行批量操作
  - 主要输入参数：图片 ID 列表、操作类型、操作参数
  - 返回数据：批量操作结果
  - 数据修改：根据操作类型修改相关数据

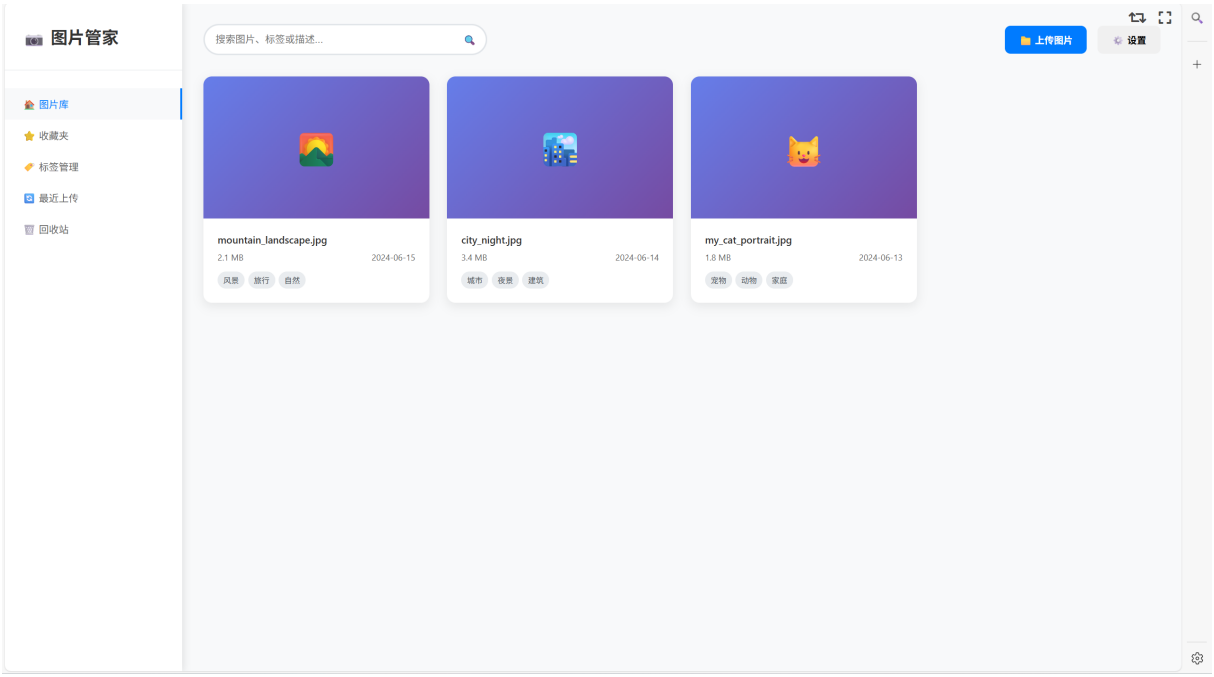
6 系统界面设计

6.1 用户登录界面

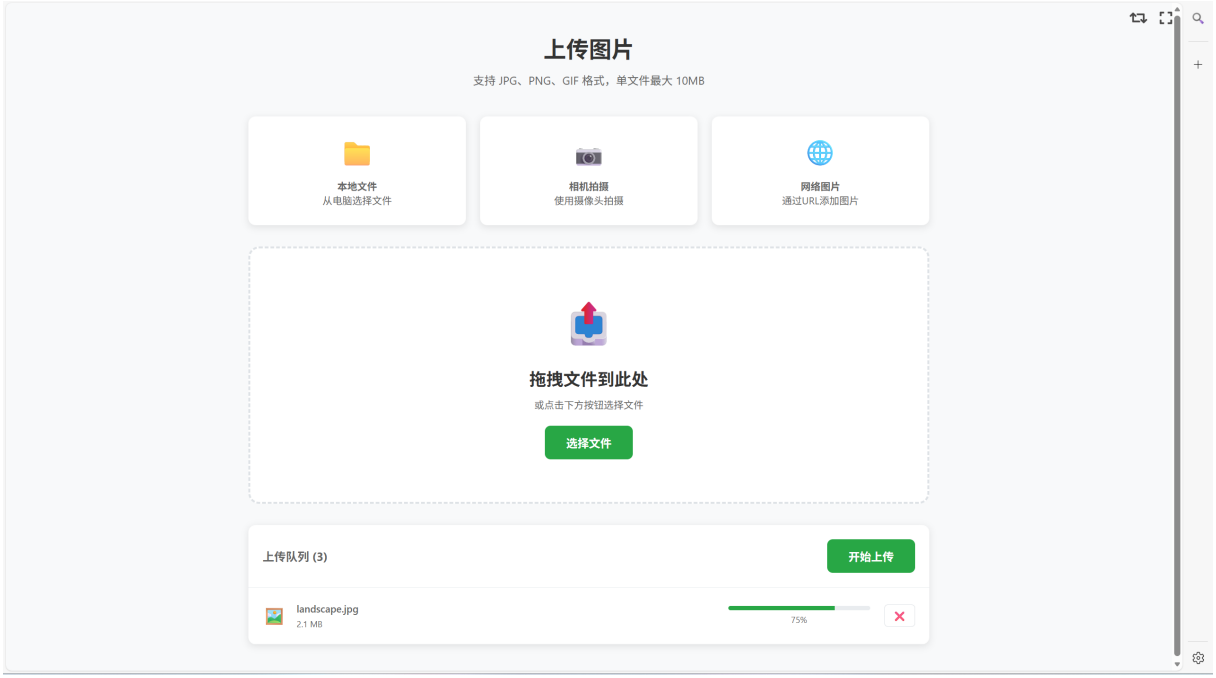
6.2 用户注册界面

、

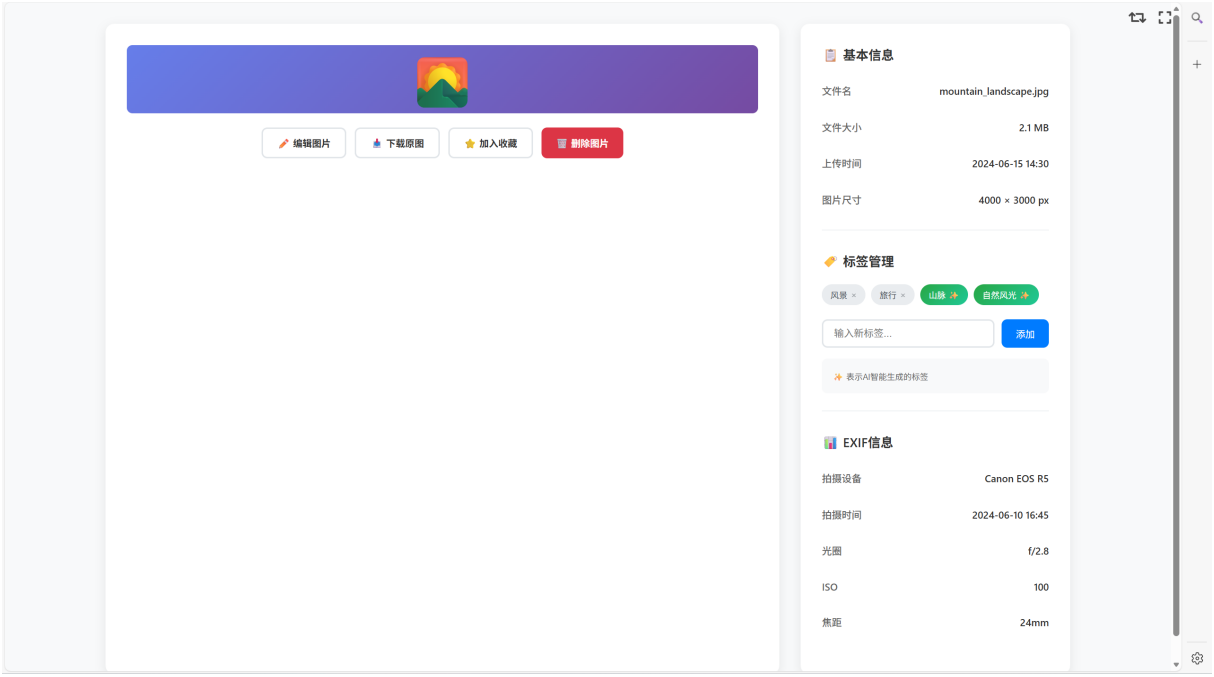
6.3 图片库总览界面



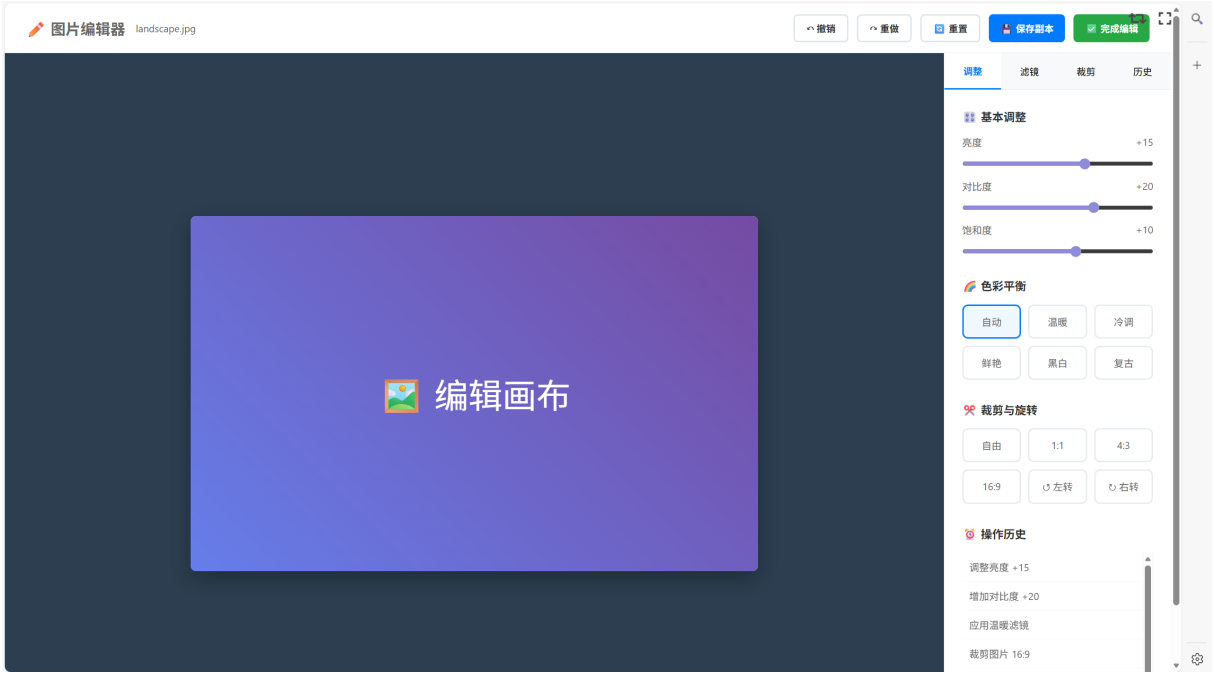
6.4 图片上传界面



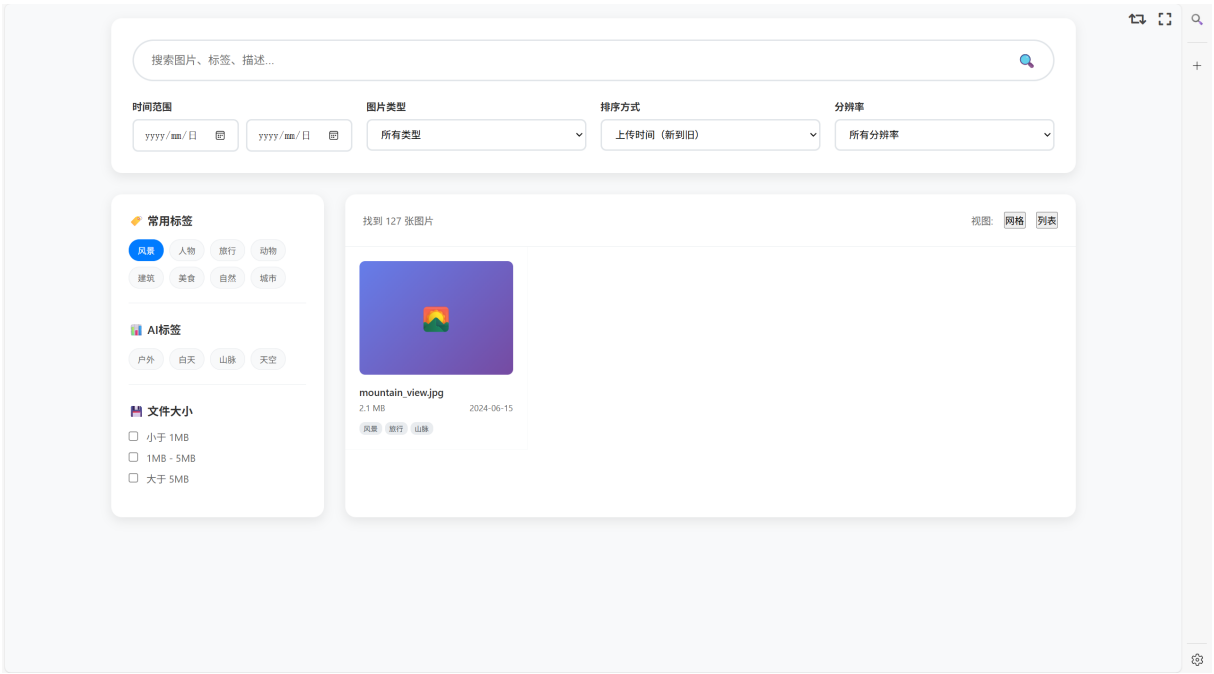
6.5 图片详情页界面



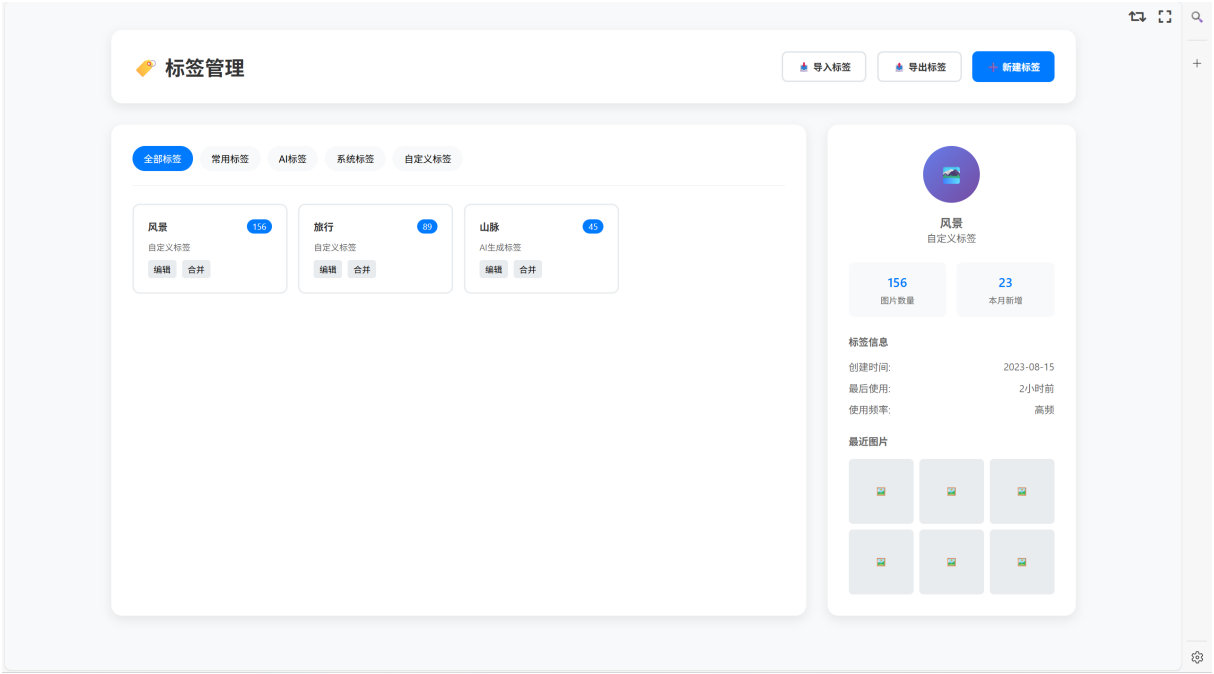
6.6 图片编辑界面



6.7 搜索与筛选界面



6.8 标签管理界面



7 系统出错处理

为确保系统的稳定性和可靠性，本部分将详细阐述系统在遇到各类错误时的处理机制和应对方案。

7.1 客户端错误（4xx）

7.1.1 400 Bad Request - 请求参数错误

- 现象：
  - 用户提交表单时界面出现红色错误提示框，显示” 请求参数格式不正确”。
  - 浏览器开发者工具网络面板显示 HTTP 400 状态码，响应体中包含具体参数错误信息。
  - 提交按钮变为禁用状态，错误输入框显示红色边框和具体错误提示。
- 原因：
  - 用户注册时密码长度少于 6 个字符，触发后端验证失败。
  - 图片上传时文件大小超过系统限制。
  - 邮箱格式不符合规范，缺少”@” 符号或域名部分。
  - JSON 请求体格式错误，缺少必要的字段或数据类型不匹配。
- 防范措施：
  - 前端实施实时表单验证，在用户输入时即时检查格式和长度。
  - 对文件上传实施前端预检查，包括文件类型、大小和尺寸验证。
  - 使用 TypeScript、JSON Schema 确保请求数据结构正确性。
  - 在后端实施多层参数验证，包括注解验证和自定义验证器。
- 应急预案：
  - 前端自动聚焦到第一个错误字段，并显示详细修正指引。
  - 提供默认值或示例格式提示，帮助用户正确填写。
  - 对于复杂错误，提供在线客服或帮助文档链接。
  - 记录参数错误日志，用于优化验证规则和用户体验。

7.1.2 401 Unauthorized - 未授权访问

- 现象：
  - 用户操作时突然弹出” 登录已过期” 提示框。
  - 系统自动跳转到登录页面，当前操作进度丢失。
  - 页面顶部显示黄色警告条，提示” 请重新登录以继续操作”。
- 原因：
  - JWT token 过期（默认设置 24 小时有效期）。
  - 用户在其他设备修改密码，导致当前 token 失效。
  - 服务器重启或会话存储异常，导致 session 丢失。
  - 用户连续多次输入错误密码，账户被临时锁定。
- 防范措施：

  - 实现 token 自动刷新机制，在 token 过期前静默更新。
  - 前端监测用户活跃度，在会话即将过期时提示用户。
  - 使用持久化存储保存重要草稿，防止数据丢失。
  - 实施合理的密码策略和账户锁定机制。

- 应急预案：
  - 自动保存用户当前操作状态，登录后恢复现场。
  - 提供” 记住我” 功能，延长 token 有效期至 7 天。
  - 紧急情况下提供临时访问码或备用登录方式。
  - 记录安全日志，监控异常登录行为。

7.1.3 403 Forbidden - 权限不足

- 现象：
  - 用户操作时弹出红色警告对话框” 无权执行此操作”。
  - 其他用户的图片或数据在界面上显示为不可操作状态。
  - 浏览器控制台显示 403 错误和详细的权限验证失败信息。

- 原因：
  - 用户尝试删除或修改其他用户上传的图片。
  - 用户权限角色不足以执行某些管理功能。
  - URL 参数被恶意修改，尝试访问不属于自己的资源。
  - API 请求中用户 ID 与当前登录用户不匹配。

- 防范措施：
  - 实施严格的数据隔离，所有查询都基于当前用户 ID。
  - 后端对每个请求进行权限验证，不信任前端传递的权限信息。
  - 使用 RBAC（基于角色的访问控制）模型管理用户权限。
  - 对敏感操作实施双因素认证或管理员审批。

- 应急预案：
  - 立即记录安全事件，包括用户 ID、操作类型和资源信息。
  - 向系统管理员发送安全警报邮件或短信通知。
  - 对频繁触发权限错误的账户进行临时锁定和审计。
  - 提供清晰的权限说明文档和申请流程。

7.1.4 404 Not Found - 资源不存在

- 现象：
  - 页面显示” 您访问的图片不存在或已被删除” 提示信息。
  - 图片缩略图位置显示破损图标和错误提示文字。
  - 浏览器地址栏显示不存在的 URL 路径，页面呈现 404 错误页面。
- 原因：
  - 用户访问已被删除的图片 ID 或用户 ID。
  - URL 拼写错误或参数被意外修改。
  - 数据库记录已被物理删除，但前端缓存中仍存在引用。
  - 图片文件被手动从服务器删除，但数据库记录未同步更新。
- 防范措施：
  - 实施软删除机制，保留数据记录但标记为已删除状态。
  - 前端实施路由守卫，验证页面参数的有效性。
  - 定期清理无效的缓存数据和死链接。
  - 建立文件与数据库的同步校验机制。
- 应急预案：
  - 显示友好的 404 页面，提供搜索功能或返回首页选项。
  - 自动检查并清理本地缓存中的无效数据。
  - 记录资源访问失败日志，用于排查数据一致性问题。
  - 提供反馈渠道，让用户报告遇到的链接问题。

7.2 服务端错误（5xx）

7.2.1 500 Internal Server Error - 服务器内部错误

- 现象：
  - 用户操作后界面显示” 系统内部错误，请稍后重试”。
  - 请求长时间无响应，最终显示服务器错误页面。
  - 系统监控平台收到大量错误告警，错误率突然飙升。
- 原因：
  - 数据库连接池耗尽，无法建立新的数据库连接。
  - 第三方服务（AI 识别服务）突然不可用或响应超时。
  - 代码中存在未处理的异常或空指针引用。
  - 服务器内存溢出或磁盘空间不足。
- 防范措施：

  - 实施完整的异常捕获和全局异常处理机制。
  - 对关键服务实施熔断和降级策略。
  - 定期进行压力测试，确保系统资源的合理配置。
  - 实施代码审查和自动化测试，减少潜在 bug。

- 应急预案：
  - 自动切换到备用服务实例或降级到简化功能模式。
  - 立即触发告警通知运维团队进行紧急处理。
  - 显示系统维护页面，避免用户重复尝试失败操作。
  - 记录完整的错误堆栈和上下文信息用于问题排查。

7.2.2 503 Service Unavailable - 服务不可用

- 现象：
  - 用户访问时显示” 系统维护中，请稍后再试” 的友好提示页面。
  - 部分功能模块显示” 服务暂时不可用” 的占位界面。
  - API 响应时间明显变长，最终返回 503 状态码。

- 原因：
  - 计划内的系统维护和版本更新操作。
  - 服务器负载过高，主动拒绝新的请求以保护系统。
  - 依赖的云服务出现区域性故障。
  - 数据库进行备份或优化操作，暂时停止服务。

- 防范措施：
  - 实施负载均衡和自动扩缩容机制。
  - 建立完善的系统维护流程和变更管理。
  - 使用多区域部署，提高系统容灾能力。
  - 对关键服务实施健康检查和自动故障转移。

- 应急预案：
  - 自动将流量切换到备用数据中心或服务区域。
  - 显示系统状态页面，实时更新服务恢复进度。
  - 通过邮件、短信等多种渠道通知用户服务中断信息。
  - 准备手动应急预案，在自动化失效时人工介入处理。

7.3 业务逻辑异常

7.3.1 数据操作异常

- 现象:
  - 用户保存数据时显示” 保存失败，请检查数据有效性”。
  - 批量操作部分成功部分失败，显示详细的结果报告。
  - 系统日志中出现数据库约束违反或死锁错误信息。
- 原因:
  - 并发操作导致数据库死锁或乐观锁版本冲突。
  - 数据完整性约束违反（如外键关联数据不存在）。
  - 网络分区导致分布式事务提交失败。
  - 唯一索引冲突，尝试插入重复数据。
- 防范措施:
  - 实施合理的数据库索引设计和查询优化。
  - 使用乐观锁机制处理并发更新冲突。
  - 对关键操作实施队列化和顺序化处理。
  - 实施数据验证和业务规则检查的前置防护。
- 应急预案:
  - 自动重试机制，对可重试错误进行有限次数的重试。
  - 提供操作冲突解决界面，让用户选择处理方式。
  - 实施操作回滚和补偿事务，确保数据一致性。
  - 记录详细的操作日志，支持手动数据修复。

7.3.2 文件操作异常

- 现象:
  - 图片上传进度条卡住，最终显示” 文件处理失败”。
  - 图片显示为破损图标，控制台显示文件加载错误。
  - 系统监控显示磁盘使用率异常升高或 IO 性能下降。
- 原因;
  - 文件上传过程中网络连接中断。
  - 服务器磁盘空间不足，无法写入新文件。
  - 文件权限配置错误，进程无权访问存储目录。
  - 图片文件损坏或格式解析失败。
- 防范措施:

  - 实施分块上传和断点续传机制。
  - 建立磁盘空间监控和自动预警机制。
  - 定期检查文件系统权限和存储服务状态。
  - 实施文件格式预验证和病毒扫描。

- 应急预案：
  - 自动清理临时文件和过期缓存释放磁盘空间。
  - 提供手动重试机制，保留用户已上传的数据。
  - 切换到备用存储位置或降级到简化存储方案。
  - 对于关键文件，实施多副本存储和数据备份恢复。

7.4 外部服务异常

7.4.1 AI 服务异常

- 现象：
  - 图片上传后智能标签生成失败，显示”AI 服务暂不可用”。
  - AI 分析进度条长时间停滞，最终超时失败。
  - 系统监控显示 AI 服务响应时间异常或错误率升高。

- 原因：
  - AI 服务提供商 API 配额用尽或服务被限流。
  - 网络延迟导致请求超时或连接中断。
  - AI 模型升级或维护期间服务不可用。
  - 输入图片格式特殊，导致 AI 模型处理异常。

- 防范措施：
  - 实施多 AI 服务商备用方案，支持自动切换。
  - 设置合理的超时时间和重试策略。
  - 购买充足的 API 配额并实施使用量监控。
  - 对输入图片进行预处理，确保符合 AI 服务要求。

- 应急预案：
  - 自动降级到基于 EXIF 和用户行为的标签推荐。
  - 将 AI 处理任务加入异步队列，稍后重试。
  - 提供手动标签输入界面，避免流程阻塞。
  - 记录 AI 服务异常模式，优化服务商选择策略。

7.4.2 数据库异常

- 现象：
  - 系统操作响应缓慢，最终显示” 数据库连接失败”。
  - 监控系统显示数据库连接数突增或 CPU 使用率异常。
  - 部分查询返回空结果或数据不一致。
- 原因：
  - 数据库服务器硬件故障或网络分区。
  - SQL 查询未使用索引导致全表扫描和性能下降。
  - 数据库连接泄漏，耗尽连接池资源。
  - 数据库主从同步延迟或失败。
- 防范措施：
  - 实施数据库读写分离和连接池监控。
  - 建立 SQL 审核机制，优化低效查询。
  - 实施数据库监控和自动告警。
  - 定期进行数据库备份和恢复演练。
- 应急预案：
  - 自动切换到备用数据库或只读副本。
  - 实施查询超时和自动取消机制。
  - 紧急情况下启用维护模式，限制非核心功能。
  - 准备数据库回滚方案，确保数据安全。

7.5 错误处理流程标准化

7.5.1 错误信息收集

- 建立统一的错误信息收集和分类系统。
- 自动捕获用户操作上下文和环境信息。
- 实施错误信息脱敏，保护用户隐私数据。

7.5.2 错误分析改进

- 定期分析错误日志，识别系统薄弱环节。
- 建立错误根本原因分析流程。
- 将错误处理经验反哺到系统设计和开发流程。

7.5.3 用户体验优化

- 设计友好的错误提示界面，提供明确的解决指引。
- 实施错误反馈机制，收集用户遇到的问题。
- 建立用户帮助中心，提供常见问题解决方案。

8 项目开发计划

开发时间	开发内容
11 月 2 日前	系统需求分析、总体架构设计、技术选型
11 月 3 日-11 月 9 日	前后端开发知识预备，撰写中期设计报告
11 月 10 日-11 月 16 日	前端界面 UI 设计与开发
11 月 17 日-11 月 30 日	数据库及后端接口设计与开发
12 月 1 日-12 月 14 日	完成前端与后端的集成测试
12 月 15 日-12 月 21 日	完成系统的部署与测试
12 月 22 日-12 月 31 日	完成最终设计报告、系统测试文档、用户手册等文档的编写

表 6: 项目开发计划时间表