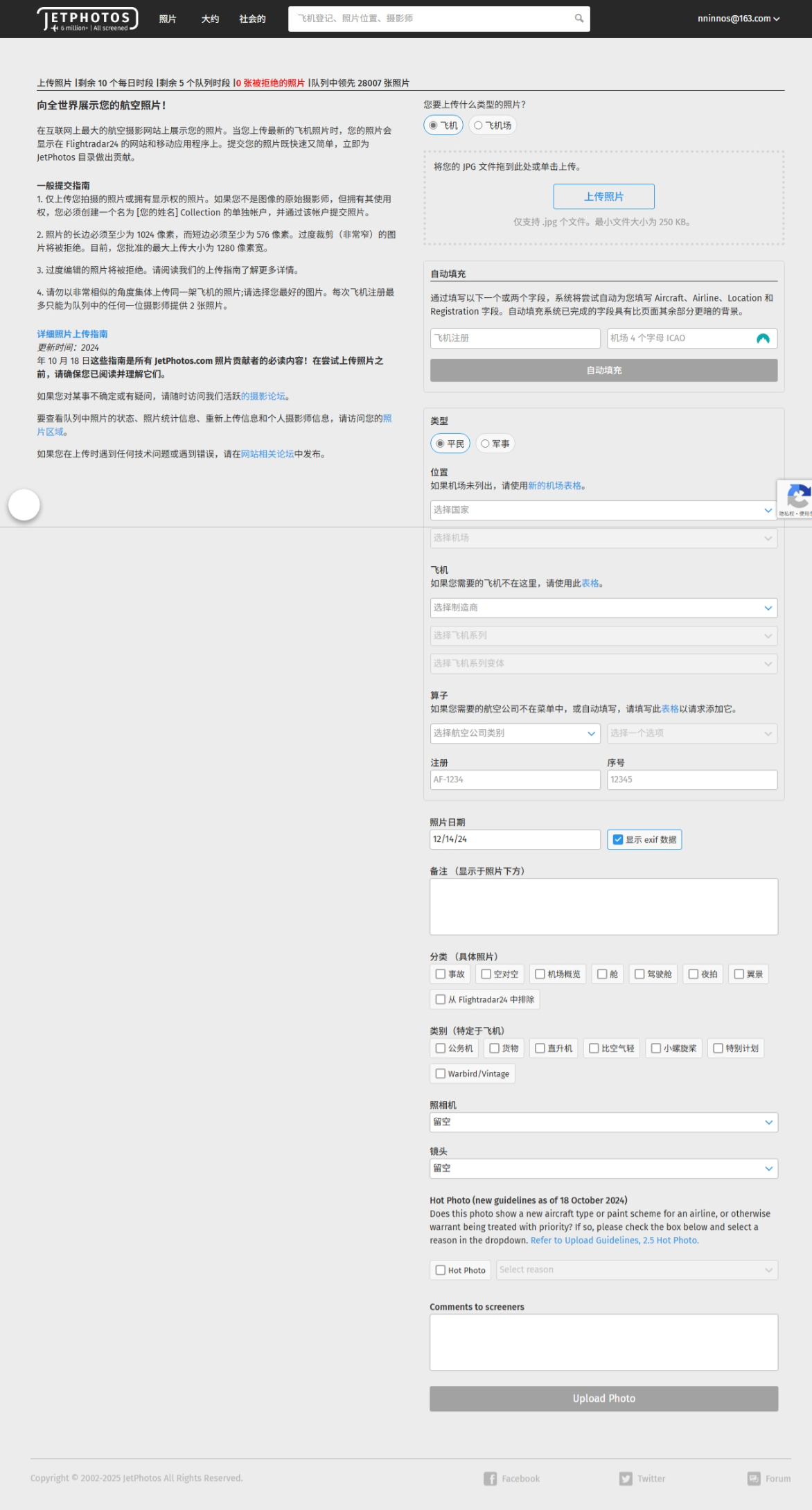
# 航空图片管理系统设计方案

## 一、项目概述

本飞机图片管理系统致力于为航空爱好者、摄影师以及相关专业人士打造一个综合性的图片管理与交流平台。系统融合前沿的前端和后端技术，运用精心设计的数据库架构，实现高效的图片管理、便捷的用户交互以及专业的审图流程，满足不同用户群体的多样化需求。

## 二、系统功能模块设计

### （一）前端功能模块

1. **主页**
   * **布局**：整体遵循 Apple Design 简洁直观原则，采用玻璃拟态与圆角设计，确保在各类设备上呈现完美适配的响应式布局。导航栏位于页面顶部，以半透明玻璃质感搭配圆角边框展现，集成系统 logo、主页、上传页、文档页、账号页等核心链接，同时设有搜索框和用户登录状态显示区域。页面主体的图片展示区运用瀑布流布局，展示热门图片和最新上传图片。每个图片卡片均采用圆角设计并具备玻璃拟态效果，卡片上清晰呈现飞机型号、拍摄地点、点赞数等基本信息。当鼠标悬停在图片上时，卡片会产生轻微的放大和阴影变化效果，增强用户交互体验。页脚部分位于页面底部，包含版权信息、联系方式以及社交媒体链接，以较小字体和淡色显示，避免影响页面整体视觉效果。
   * **功能**：支持强大的图片搜索功能，用户在导航栏搜索框输入飞机型号、拍摄地点等关键词，即可快速查找相关图片；提供丰富的筛选条件，如拍摄时间、图片类型等，帮助用户精准定位感兴趣的图片；用户可对喜爱的图片进行点赞和收藏操作，点赞数实时更新，方便用户记录和分享自己的喜好。此外，增加图片推荐功能，根据用户的浏览历史、点赞和收藏行为，利用个性化推荐算法为用户推送可能感兴趣的图片。
2. **上传页**
   * **布局**：延续玻璃拟态和圆角设计风格，导航栏与主页保持一致，确保用户操作的连贯性。页面中间的上传表单区采用圆角矩形的卡片式设计，搭配玻璃拟态效果，营造出科技感和层次感。表单包含图片选择框、图片信息填写区域（涵盖飞机型号、拍摄地点、拍摄时间、图片描述等）以及提交按钮，输入框和按钮均采用圆角设计，提升用户输入体验。页脚部分包含版权信息和联系方式，保持页面整体的完整性。
   * **功能**：在用户选择图片时，系统自动验证图片格式（如 JPEG、PNG 等）和大小，确保符合系统要求；用户选择图片后，页面实时预览图片，方便用户确认上传内容；填写图片信息时，系统提供必要的提示和验证机制，确保信息完整准确；图片上传过程中，显示上传进度条，让用户清晰了解上传状态。同时，增加图片分类建议功能，根据用户输入的图片信息，利用智能算法为用户提供合适的图片分类建议，便于图片的整理和检索。
   * 
3. **文档页**
   * **布局**：同样采用玻璃拟态和圆角设计，导航栏与其他页面一致，为用户提供统一的操作入口。文档展示区采用单栏布局，通过 Markdown 解析插件将文档内容转换为可阅读的 HTML 格式进行展示。文档按照使用说明、常见问题解答、审图规则等不同类别进行分类展示，方便用户快速查找所需信息。页脚部分包含版权信息和联系方式，保持页面简洁美观。
   * **功能**：实现文档分类展示，用户可根据分类快速定位所需文档；提供搜索框，用户输入关键词即可搜索文档内容。此外，增加文档收藏功能，用户可将常用文档添加到收藏夹，方便下次快速访问；同时支持文档分享功能，用户可将有价值的文档分享到社交媒体或其他平台，促进知识交流。
4. **账号页**
   * **布局**：运用玻璃拟态和圆角设计，导航栏与其他页面保持一致。页面中间分为左右两栏，左栏展示用户个人信息（包括用户名、头像、邮箱等）以及修改信息的按钮，方便用户管理个人资料；右栏展示用户上传历史和收藏的图片列表，每个图片卡片采用圆角设计并具备玻璃拟态效果，提升视觉体验。页脚部分包含版权信息和联系方式，确保页面完整性。
   * **功能**：支持用户修改个人信息，如用户名、头像、邮箱等；提供密码修改功能，保障用户账号安全；用户可查看上传历史，包括图片基本信息和审核状态；查看收藏图片列表，点击图片可查看详情。另外，增加账号安全提醒功能，当检测到异常登录行为或账号安全风险时，及时向用户发送提醒信息，保障用户账号安全。
5. **图片详情页**
   * **布局**：采用玻璃拟态和圆角设计，导航栏与其他页面相同。页面中间分为上下两栏，上栏展示图片的大图，图片具有轻微的玻璃拟态效果，支持用户通过鼠标滚轮或按钮进行放大缩小操作；下栏展示图片的详细信息（如飞机型号、拍摄地点、拍摄时间、图片描述、点赞数、评论数等）以及评论区域，评论区域的每条评论采用圆角设计并带有轻微的玻璃拟态效果，营造出舒适的阅读和交互环境。页脚部分包含版权信息和联系方式。
   * **功能**：实现图片放大缩小功能，方便用户查看图片细节；支持图片分享到微信、微博、QQ 等社交媒体平台，扩大图片传播范围；用户可对图片发表评论，支持回复评论和删除自己的评论；对图片进行点赞操作，点赞数实时更新。在此基础上，增加图片对比功能，用户可以选择两张图片进行对比，系统自动分析并展示图片在拍摄参数、飞机型号细节等方面的差异，为用户提供更深入的图片分析体验。
6. **审图员 - 已审图片页**
   * **布局**：采用玻璃拟态和圆角设计，导航栏与其他页面一致。图片列表区以表格形式展示已审图片信息，表格包含图片缩略图、图片基本信息、审核结果、审核分数等列，每个表格单元格具有轻微的玻璃拟态效果，鼠标悬停时会有颜色变化，方便审图员查看和区分。表格下方设有分页导航栏，便于审图员查看不同页面的图片。页脚部分包含版权信息和联系方式。
   * **功能**：提供筛选条件，如审核时间、审核结果、审核分数等，帮助审图员快速找到所需图片；审图员点击图片缩略图或图片基本信息，可查看图片详情。此外，增加审核结果统计分析功能，以图表形式展示不同审核结果的图片数量、占比等信息，帮助审图员直观了解审核工作情况，发现审核趋势和问题。
7. **审图员 - 审图页面**
   * **布局**：运用玻璃拟态和圆角设计，采用左右两栏布局，导航栏与其他页面一致。左栏展示待审核图片的大图和基本信息，图片具有轻微的玻璃拟态效果；右栏是审图表单，包含打分区域、批注区域、审核结果选择框和提交按钮，表单元素采用圆角设计并具备玻璃拟态效果，营造出专业的审核环境。页脚部分包含版权信息和联系方式。
   * **功能**：使用 Canvas 或 SVG 技术实现图片标注功能，审图员可在图片上进行批注和标记；审图员根据审图方案对图片进行打分；在批注区域对图片进行详细批注和说明；选择审核结果，如通过、不通过等；完成审核后，点击提交按钮将审核结果和批注发送到后端保存。同时，增加审图模板功能，审图员可以根据不同的审核场景选择预设的审图模板，快速填写审核信息，提高审核效率。
8. **管理员 - 审图任务分配页**
   * **布局**：采用玻璃拟态和圆角设计，导航栏与其他页面一致。任务分配区位于页面中间，包含待分配任务列表、审图员列表和任务分配操作按钮，列表和按钮采用圆角设计并具备玻璃拟态效果，操作界面简洁直观。任务统计区位于页面一侧，以图表形式展示每个审图员的任务数量和完成情况，图表具有轻微的玻璃拟态效果，便于管理员实时掌握任务进度。页脚部分包含版权信息和联系方式。
   * **功能**：管理员通过拖拽或选择方式将审图任务分配给不同审图员；查看已分配任务列表，并对任务进行修改和重新分配；统计每个审图员的任务数量和完成情况，以图表形式展示。此外，增加任务优先级设置功能，管理员可以根据任务的紧急程度和重要性为任务设置优先级，确保重要任务优先处理。
9. **管理员 - 审图方案管理页**
   * **布局**：使用玻璃拟态和圆角设计，导航栏与其他页面相同。方案添加区位于页面上方，包含审图方案添加表单，表单包含方案名称、方案描述、审核规则等输入框和添加按钮，表单元素采用圆角设计并具备玻璃拟态效果。方案列表区位于页面下方，以表格形式展示审图方案列表，表格包含方案名称、方案描述、审核规则等列，表格单元格具有轻微的玻璃拟态效果，鼠标悬停时会有颜色变化。页脚部分包含版权信息和联系方式。
   * **功能**：管理员添加新的审图方案，填写方案名称、方案描述、审核规则等信息；对已有的审图方案进行修改和更新；删除不需要的审图方案。同时，增加审图方案版本管理功能，记录审图方案的修改历史，方便管理员追溯和管理不同版本的审图方案。
10. **管理员 - 系统管理页**
    * **布局**：采用玻璃拟态和圆角设计，导航栏与其他页面一致。数据库选择区位于页面中间，设有下拉菜单让管理员选择数据库类型（MySQL 或 MongoDB），下拉菜单采用圆角设计并具备玻璃拟态效果。根据选择的数据库类型，显示相应的数据库配置表单，管理员可填写数据库连接信息，如主机名、端口号、用户名、密码等，表单元素采用圆角设计并具备玻璃拟态效果。云存储配置区预留 Amazon S3、阿里云 OSS 的配置接口，管理员可填写相应配置信息，实现图片的云存储，配置表单元素同样采用圆角设计并具备玻璃拟态效果。页脚部分包含版权信息和联系方式。
    * **功能**：管理员通过下拉菜单选择数据库类型；根据选择的数据库类型，提供相应数据库配置表单，填写数据库连接信息；填写 Amazon S3、阿里云 OSS 的配置信息，实现图片的云存储。此外，增加系统日志管理功能，记录系统操作日志，包括用户登录、数据修改、任务执行等信息，便于管理员进行系统监控和故障排查。

### （二）后端功能模块

1. **用户管理**
   * **用户注册**：使用 Flask 框架搭建后端 API，接收用户注册请求，对用户输入的用户名、密码、邮箱等信息进行严格的合法性验证，确保信息符合规范后，将用户信息存储到数据库中。
   * **用户登录**：接收用户登录请求，验证用户输入的用户名和密码是否正确，若验证通过，生成并返回 JWT 令牌，用于后续的身份验证，保障用户登录状态的安全管理。
   * **用户信息更新**：接收用户信息更新请求，首先验证用户身份，确保操作的合法性，然后更新数据库中的用户信息，保证用户资料的及时准确。
2. **图片管理**
   * **图片上传**：接收用户上传的图片，运用 Pillow 库对图片进行压缩、裁剪等处理，优化图片质量和大小，然后将处理后的图片保存到本地文件系统或云存储（如 Amazon S3、阿里云 OSS），同时将图片的路径、文件名、相关描述等信息存储到数据库中，实现图片的有效管理。
   * **图片获取**：根据用户的请求，从数据库中查询图片信息，并返回图片的 URL 或图片数据，满足用户查看图片的需求。
   * **图片删除**：接收用户删除图片的请求，先验证用户身份，确认用户有权限删除该图片后，删除数据库中的图片信息，并从文件系统或云存储中删除相应的图片文件，确保图片数据的一致性和安全性。
3. **审图管理**
   * **审图任务分配**：根据审图方案和审图员的分配规则，将审图任务合理分配给不同的审图员，并将任务信息存储到数据库中，保证审图工作的有序进行。
   * **审图结果收集**：接收审图员提交的审核结果和批注，对审核结果进行合法性验证，确保审核结果的准确性和可靠性，然后将审核结果和批注存储到数据库中，为后续的图片管理和质量评估提供依据。
   * **审图方案管理**：提供 API 接口，允许管理员添加、修改、删除审图方案，并将方案信息存储到数据库中，实现审图方案的灵活管理和更新。
4. **数据库管理**
   * **数据库连接**：根据管理员的选择，动态加载不同的数据库连接和操作函数，实现与 MySQL 或 MongoDB 的稳定连接，确保系统能够高效地访问和管理数据库。
   * **数据迁移**：在数据库类型切换时，提供数据迁移工具或接口，确保数据在不同数据库之间的一致性和完整性，避免数据丢失或损坏。
5. **系统监控与日志管理**：增加系统监控功能，实时监测系统的性能指标，如服务器负载、数据库连接数、图片上传下载流量等，当指标超出正常范围时，及时向管理员发送预警信息。同时，完善系统日志管理功能，详细记录系统操作日志，包括用户行为、系统错误、任务执行情况等，便于管理员进行系统维护和故障排查。

## 三、数据库设计

### （一）MySQL 数据库

1. **用户表（users）**：存储用户 ID、用户名、密码、邮箱、头像等基本信息，为用户管理提供数据支持。
2. **图片表（images）**：记录图片 ID、用户 ID、图片路径、飞机型号、拍摄地点、拍摄时间、图片描述等信息，实现对图片的详细管理和检索。
3. **评论表（comments）**：保存评论 ID、图片 ID、用户 ID、评论内容、评论时间等信息，用于管理图片的评论数据，促进用户之间的交流和互动。
4. **审图任务表（review\_tasks）**：存储任务 ID、图片 ID、审图员 ID、审核结果、审核分数、审核时间等信息，跟踪审图任务的分配和完成情况。
5. **审图方案表（review\_schemes）**：包含方案 ID、方案名称、方案描述、审核规则等信息，管理不同的审图方案，确保审图工作的标准化和规范化。
6. **系统日志表（system\_logs）**：记录系统操作日志，包括操作时间、操作人、操作内容、操作结果等信息，便于系统监控和故障排查。

### （二）MongoDB 数据库

1. **用户集合（users）**：以 JSON 格式存储用户基本信息，支持灵活的文档结构，满足用户信息的多样化存储需求。
2. **图片集合（images）**：存储图片基本信息，以 JSON 格式存储，便于存储和管理图片的元数据，如拍摄设备信息、图片色彩模式等。
3. **评论集合（comments）**：保存图片评论信息，以 JSON 格式存储，方便扩展评论相关的附加信息，如评论点赞数、回复数量等。
4. **审图任务集合（review\_tasks）**：记录审图任务分配和完成情况，以 JSON 格式存储，适应审图任务数据的动态变化和灵活查询需求。
5. **审图方案集合（review\_schemes）**：存储不同审图方案，以 JSON 格式存储，便于管理和更新审图方案的复杂规则和描述信息。
6. **系统监控集合（system\_monitoring）**：存储系统监控数据，如服务器性能指标、系统运行状态等，以 JSON 格式存储，支持实时数据的快速存储和查询，为系统性能优化提供数据依据。

## 四、技术选型

### （一）前端技术

1. **框架**：选择 Vue.js 构建用户界面，其组件化开发模式和响应式设计理念，能够高效地创建可复用的 UI 组件，提升开发效率和用户体验。
2. **路由管理**：运用 Vue Router 进行页面路由管理，实现页面之间的平滑切换和导航控制，确保用户操作的流畅性和连贯性。
3. **状态管理**：借助 Vuex 对用户登录状态、个人信息以及图片数据等全局状态进行统一管理，保证数据的一致性和可维护性。
4. **UI 框架**：采用 Tailwind CSS 构建美观且响应式的界面，其丰富的 CSS 类库和灵活的设计模式，能够快速搭建出符合现代审美标准的界面。同时，结合 Font Awesome 图标库，为界面增添丰富多样的图标，提升界面的可视化效果和操作便捷性。

### （二）后端技术

1. **框架**：选用 Python 的 Flask 框架搭建后端 API，其轻量级和灵活性特点，便于快速开发和部署后端服务，满足系统的功能需求。
2. **数据库交互**：使用 SQLAlchemy 与 MySQL 进行交互，通过其强大的 ORM（对象关系映射）功能，简化数据库操作，提高开发效率。利用 PyMongo 与 MongoDB 进行交互，实现对非结构化数据的高效存储和查询。
3. **身份验证**：采用 JWT（JSON Web Token）进行用户身份验证和授权，确保用户登录状态的安全管理，防止非法访问和数据泄露。
4. **异步处理**：运用 Celery 进行异步任务处理，如图片处理和审图任务分配，提高系统的响应速度和并发处理能力，避免因长时间任务导致系统卡顿。

## 五、技术难点及解决方案

### （一）图片处理和存储

1. **难点**：处理不同格式和大小的图片时，需要在保证图片质量的同时，提高存储效率，并且确保图片在不同设备上的兼容性和加载速度。
2. **解决方案**：使用 Python 的 Pillow 库对图片进行压缩、裁剪等处理，根据不同的应用场景和设备需求，调整图片的分辨率、格式和大小。将处理后的图片存储到本地文件系统或云存储（如 Amazon S3、阿里云 OSS），利用云存储的优势，提高图片的存储和访问效率，同时确保图片在各种设备上的稳定加载。

### （二）并发处理

1. **难点**：在大量用户同时上传图片、进行图片搜索或审图任务集中处理时，系统易出现响应延迟、服务器负载过高甚至崩溃的情况。不同操作对资源的竞争，如数据库连接、CPU 和内存的使用，可能导致数据不一致或处理错误。任务队列的管理也面临挑战，若任务堆积或分配不均衡，会影响系统整体性能和用户体验。
2. **解决方案**：采用异步编程结合任务队列（如 Celery）的方式处理并发请求。在用户上传图片或发起搜索时，将任务放入队列，系统可在后台异步处理，避免阻塞主线程，提高响应速度。利用缓存技术（如 Redis），将常用数据（如热门图片信息、频繁访问的文档内容）缓存起来，减少数据库的直接访问压力。同时，合理配置服务器资源，根据业务量动态调整服务器的 CPU、内存等资源分配。对于任务队列，引入智能调度算法，根据任务的优先级、服务器负载情况等因素，均衡分配任务，确保任务高效处理。

### （三）数据一致性

1. **难点**：在使用 MySQL 和 MongoDB 两种不同类型数据库的情况下，当数据在两个数据库之间同步时，可能出现数据更新不同步的问题，例如在 MySQL 中更新了图片信息，但 MongoDB 中的对应数据未及时更新。事务处理时，涉及多个数据库操作，要保证所有操作要么全部成功，要么全部失败，否则会导致数据不一致。在系统高并发读写操作下，数据的一致性维护更加困难，可能出现脏读、幻读等问题。
2. **解决方案**：使用数据同步工具（如 Canal），实时监控 MySQL 数据库的变化，并将变化的数据同步到 MongoDB 中。在业务逻辑中，对涉及多数据库操作的事务，采用分布式事务管理框架（如 Seata），确保数据操作的原子性。利用数据库的锁机制和事务隔离级别，合理设置 MySQL 和 MongoDB 的事务隔离级别，避免高并发下的数据不一致问题。定期进行数据一致性检查，编写脚本对两个数据库中的关键数据进行比对和修复，确保数据的完整性和一致性。

## 六、功能列表

### **（一）前端功能**

* + **主页**：图片搜索、筛选、点赞、收藏、推荐，热门和最新图片展示。
  + **上传页**：图片上传、格式和大小验证、预览、信息填写与验证、分类建议。
  + **文档页**：文档分类展示、搜索、收藏、分享。
  + **账号页**：个人信息修改、密码修改、上传历史和收藏图片查看、账号安全提醒。
  + **图片详情页**：图片放大缩小、分享、评论、点赞、图片对比。
  + **审图员 - 已审图片页**：已审图片筛选、详情查看、审核结果统计分析。
  + **审图员 - 审图页面**：图片标注、打分、批注、选择审核结果、提交审核、审图模板选择。
  + **管理员 - 审图任务分配页**：任务分配、查看与修改、任务数量和完成情况统计、任务优先级设置。
  + **管理员 - 审图方案管理页**：审图方案添加、修改、删除、版本管理。
  + **管理员 - 系统管理页**：数据库类型选择与配置、云存储配置、系统日志管理。

### **（二）后端功能**

* + **用户管理**：用户注册、登录、信息更新。
  + **图片管理**：图片上传、获取、删除。
  + **审图管理**：审图任务分配、结果收集、方案管理。
  + **数据库管理**：数据库连接、数据迁移。
  + **系统监控与日志管理**：系统性能监控、预警、日志记录与管理。

## 七、分步实现步骤

### （一）前端功能分步实现步骤

1. **主页**
   * **布局搭建**：运用 Vue 组件和 Tailwind CSS，构建符合设计的导航栏、图片展示区和页脚，设置响应式布局，实现玻璃拟态与圆角效果。
   * **搜索功能实现**：在导航栏搜索框绑定输入事件，获取用户输入关键词，调用后端 API 进行搜索，将结果展示在图片展示区。
   * **筛选功能实现**：创建筛选条件选择组件，绑定选择事件，将筛选条件传递给后端 API，获取筛选后的图片数据并展示。
   * **点赞和收藏功能实现**：为图片卡片添加点赞和收藏按钮，绑定点击事件，向后端发送请求，更新点赞数并在前端实时显示，将收藏信息存储在用户本地或后端。
   * **图片推荐功能实现**：在用户操作时收集浏览历史、点赞和收藏数据，定期向后端发送数据，后端利用算法生成推荐图片列表，前端接收并展示。
2. **上传页**
   * **布局设计**：使用 Vue 组件和 Tailwind CSS，设计上传表单区和页脚，保持整体设计风格一致。
   * **图片格式和大小验证**：在文件选择框绑定 change 事件，使用 JavaScript 验证图片格式和大小，不符合要求则提示用户。
   * **图片预览功能实现**：选择图片后，利用 FileReader 对象读取图片数据，在页面生成预览图。
   * **信息填写验证功能实现**：为输入框绑定 blur 或 submit 事件，验证输入内容，为空或格式错误时提示用户。
   * **图片分类建议功能实现**：在用户输入信息时，将信息发送到后端，后端算法分析后返回分类建议，前端展示。
3. **文档页**
   * **布局构建**：运用 Vue 组件和 Tailwind CSS，构建文档展示区和页脚，确保布局简洁美观。
   * **文档分类展示功能实现**：从后端获取文档分类数据，使用 Vue 的 v - for 指令遍历渲染文档列表。
   * **搜索功能实现**：在搜索框绑定输入事件，调用后端搜索 API，展示搜索结果。
   * **文档收藏和分享功能实现**：添加收藏和分享按钮，收藏时向后端发送请求保存收藏信息，分享时调用社交媒体分享接口。
4. **账号页**
   * **布局设计**：使用 Vue 组件和 Tailwind CSS，设计左右两栏布局展示个人信息和图片列表。
   * **个人信息修改功能实现**：点击修改按钮，切换到编辑模式，保存时向后端发送更新请求。
   * **密码修改功能实现**：创建密码修改表单，绑定提交事件，验证原密码，向后端发送修改请求。
   * **上传历史和收藏图片查看功能实现**：向后端请求数据，使用 v - for 指令遍历渲染图片列表。
   * **账号安全提醒功能实现**：前端轮询或通过 Websocket 接收后端提醒信息，弹窗提示用户。
5. **图片详情页**
   * **布局搭建**：运用 Vue 组件和 Tailwind CSS，构建上下两栏布局展示图片和详细信息。
   * **图片放大缩小功能实现**：为图片绑定鼠标滚轮和点击事件，通过 CSS 或 JavaScript 控制图片缩放。
   * **图片分享功能实现**：添加分享按钮，调用社交媒体分享接口。
   * **评论功能实现**：创建评论输入框和提交按钮，绑定事件，向后端发送评论请求，获取评论列表并展示。
   * **图片对比功能实现**：添加对比按钮，选择两张图片后发送到后端，后端分析返回差异数据，前端展示。
6. **审图员 - 已审图片页**
   * **布局设计**：使用 Vue 组件和 Tailwind CSS，构建表格展示已审图片信息和分页导航栏。
   * **筛选功能实现**：创建筛选条件选择组件，绑定事件，向后端发送筛选请求，展示筛选结果。
   * **图片详情查看功能实现**：点击图片相关信息，向后端请求详情数据，弹窗展示。
   * **审核结果统计分析功能实现**：向后端请求统计数据，使用图表库（如 Echarts）渲染图表。
7. **审图员 - 审图页面**
   * **布局搭建**：运用 Vue 组件和 Tailwind CSS，构建左右两栏布局展示图片和审图表单。
   * **图片标注功能实现**：使用 Canvas 或 SVG 技术，在图片上绘制标注，将标注数据保存到后端。
   * **打分、批注和选择审核结果功能实现**：为表单元素绑定事件，收集数据，点击提交按钮发送到后端。
   * **审图模板功能实现**：从后端获取审图模板列表，在模板选择框绑定事件，填充表单。
8. **管理员 - 审图任务分配页**
   * **布局设计**：使用 Vue 组件和 Tailwind CSS，构建任务分配区和任务统计区。
   * **任务分配功能实现**：通过拖拽或选择操作，绑定事件，向后端发送任务分配请求。
   * **查看和修改任务功能实现**：从后端获取任务列表，展示在页面，点击修改按钮进入编辑模式，提交修改请求。
   * **任务统计功能实现**：向后端请求统计数据，使用图表库渲染图表。
   * **任务优先级设置功能实现**：添加优先级设置组件，绑定事件，向后端发送设置请求。
9. **管理员 - 审图方案管理页**
   * **布局构建**：运用 Vue 组件和 Tailwind CSS，构建方案添加区和方案列表区。
   * **审图方案添加功能实现**：在添加表单绑定提交事件，收集数据，发送到后端。
   * **审图方案修改和删除功能实现**：从后端获取方案列表，展示在页面，点击修改或删除按钮，发送相应请求。
   * **审图方案版本管理功能实现**：从后端获取版本历史数据，展示在页面，方便管理员追溯。
10. **管理员 - 系统管理页**
    * **布局设计**：使用 Vue 组件和 Tailwind CSS，构建数据库选择区和云存储配置区。
    * **数据库类型选择和配置功能实现**：在下拉菜单绑定选择事件，显示相应配置表单，提交配置信息到后端。
    * **云存储配置功能实现**：在配置表单绑定提交事件，收集信息，发送到后端。
    * **系统日志管理功能实现**：从后端获取日志数据，展示在页面，提供搜索和筛选功能。

### （二）后端功能分步实现步骤

1. **用户管理**
   * **用户注册**：使用 Flask 创建注册 API 接口，接收前端请求数据，验证用户名、密码、邮箱等信息格式，通过 SQLAlchemy 或 PyMongo 将数据存储到数据库。
   * **用户登录**：创建登录 API 接口，接收请求数据，验证用户名和密码，验证通过后使用 JWT 生成令牌，返回给前端。
   * **用户信息更新**：创建更新 API 接口，验证用户身份，接收更新数据，更新数据库中用户信息。
2. **图片管理**
   * **图片上传**：创建上传 API 接口，接收前端上传的图片数据，使用 Pillow 库处理图片，将图片保存到本地或云存储，将图片信息存储到数据库。
   * **图片获取**：创建获取 API 接口，接收前端请求参数，从数据库查询图片信息，返回图片 URL 或数据。
   * **图片删除**：创建删除 API 接口，验证用户身份，从数据库删除图片信息，从存储系统删除图片文件。
3. **审图管理**
   * **审图任务分配**：创建任务分配 API 接口，根据审图方案和分配规则，将任务分配给审图员，存储任务信息到数据库。
   * **审图结果收集**：创建结果收集 API 接口，接收审图员提交的审核结果和批注，验证数据合法性，存储到数据库。
   * **审图方案管理**：创建方案管理 API 接口，实现添加、修改、删除审图方案功能，存储方案信息到数据库。
4. **数据库管理**
   * **数据库连接**：根据管理员选择的数据库类型，使用 SQLAlchemy 或 PyMongo 动态加载连接和操作函数，建立数据库连接。
   * **数据迁移**：开发数据迁移工具或接口，在数据库类型切换时，将数据从原数据库迁移到新数据库，确保数据一致性和完整性。
5. **系统监控与日志管理**
   * **系统监控**：使用相关工具（如 Flask - MonitoringDashboard）或自定义代码，实时监测系统性能指标，指标异常时通过邮件或短信发送预警信息。
   * **日志管理**：在系统关键操作处记录日志，将日志信息存储到数据库，提供查询和分析功能。

## 八、构建

### （一）前端主页实现步骤

1. **搭建项目基础结构**：创建 Vue 项目，配置好基本的项目环境，安装所需依赖，如 Vue、Vue Router、Vuex、Tailwind CSS 和 Font Awesome 等。
2. **构建主页布局**：
   * 利用 Vue 组件创建导航栏组件。在导航栏中，按照设计添加系统 logo、主页、上传页、文档页、账号页等核心链接，设置搜索框和用户登录状态显示区域。使用 Tailwind CSS 样式，为导航栏添加半透明玻璃质感和圆角边框效果。
   * 同样使用 Vue 组件创建图片展示区组件。采用瀑布流布局展示热门图片和最新上传图片。每个图片卡片运用 Tailwind CSS 设计成圆角并具备玻璃拟态效果，在卡片上清晰呈现飞机型号、拍摄地点、点赞数等基本信息。
   * 创建页脚组件，在页脚中添加版权信息、联系方式以及社交媒体链接，设置较小字体和淡色显示。
3. **实现搜索功能**：
   * 在导航栏搜索框绑定input事件，获取用户输入的关键词。
   * 通过axios（或其他 HTTP 请求库）调用后端 API，将关键词作为参数发送请求。
   * 在前端接收后端返回的搜索结果数据，使用v - for指令遍历渲染图片列表，展示在图片展示区。
4. **实现筛选功能**：
   * 创建筛选条件选择组件，如拍摄时间、图片类型等筛选选项。使用 Tailwind CSS 设计组件样式，使其美观且易用。
   * 为筛选组件的选择元素绑定change事件，当用户选择筛选条件时，收集筛选条件数据。
   * 将筛选条件数据与搜索关键词（若有）一起通过axios发送给后端 API。
   * 接收后端返回的筛选后的图片数据，更新图片展示区的图片列表。
5. **实现点赞和收藏功能**：
   * 为图片卡片添加点赞按钮和收藏按钮，使用 Font Awesome 图标库为按钮添加合适的图标。
   * 给点赞按钮绑定click事件，当点击点赞按钮时，通过axios向后端发送点赞请求，携带图片 ID 等必要信息。
   * 后端接收到点赞请求后，更新数据库中该图片的点赞数，并返回新的点赞数。
   * 前端接收到新的点赞数后，实时更新图片卡片上的点赞数显示。
   * 收藏功能类似，点击收藏按钮时，向后端发送收藏请求，后端存储收藏信息（可存储在用户收藏表关联图片 ID 和用户 ID），前端可在本地存储收藏标识（如在 Vuex 中管理收藏状态），用于在用户操作时快速判断图片是否已收藏，并更新页面显示。
6. **实现图片推荐功能**：
   * 在用户进行浏览、点赞、收藏等操作时，利用 Vuex 收集用户行为数据，如浏览的图片 ID、点赞和收藏的图片 ID 及操作时间等。
   * 定期（如每隔一段时间或在用户特定操作后）通过axios将用户行为数据发送给后端。
   * 后端接收到数据后，利用个性化推荐算法（如协同过滤算法等）分析数据，生成推荐图片列表。
   * 前端接收后端返回的推荐图片列表数据，在图片展示区展示推荐图片。

## 九、UI/UX设计要求

### 1. 视觉风格

- 玻璃拟态设计 ：使用半透明背景(background: rgba(255, 255, 255, 0.25))和模糊效果(backdrop-filter: blur(12px))

- 圆角设计 ：统一使用大圆角(border-radius: 24px)

- 渐变背景 ：线性渐变背景(background: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%))

### 2. 色彩系统

- 主色调 ：蓝紫色系(rgba(38, 45, 145, 0.281))

- 文字颜色 ：深灰色(#2c3e50)和浅灰色(#7f8c8d)

- 强调色 ：蓝色(#007aff)用于激活状态

- 错误提示 ：红色(#ff4d4f)用于通知条

### 3. 排版规范

- 标题 ：居中，字体重量500(font-weight: 500)，1.5rem下边距

- 输入框 ：

- 统一内边距(padding: 0.75rem)

- 圆角12px(border-radius: 12px)

- 半透明白色背景(background: rgba(255, 255, 255, 0.3))

- 按钮 ：

- 圆角12px

- 无边框(border: none)

- 悬停效果：轻微上移(transform: translateY(-1px))和阴影

### 4. 交互设计

- 表单切换 ：无边框标签，激活时下划线动画

- 验证码按钮 ：禁用状态有透明度变化(opacity: 0.7)

- 通知系统 ：顶部滑入的圆角通知条，3秒自动消失

- 输入反馈 ：聚焦时有蓝色边框和阴影效果

### 5. 响应式设计

- 桌面端 ：更大的内边距(padding: 3rem)和阴影(box-shadow)

- 平板 ：缩小内边距(padding: 1.5rem)，调整圆角(20px)

- 手机端 ：进一步缩小内边距(padding: 1.25rem)，调整字体大小

### 6. 动效规范

- 通用过渡 ：所有交互元素都有0.3秒过渡效果(transition: all 0.3s ease)

- 标签切换 ：下划线有缩放动画(animation: underline 0.3s ease-out)

- 通知条 ：0.5秒滑入动画(transition: top 0.5s ease)

### 7. 一致性原则

- 登录和注册页面保持完全相同的视觉风格和交互模式

- 所有输入框和按钮的样式完全一致

- 相同的响应式断点和处理方式