**一、Photox 后端需求文档：**

后端核心需求：

**1. 用户模块 (Users)**

* **功能需求:**
  + 用户注册（用户名、密码、邮箱）。
  + 用户登录（用户名、密码），返回JWT Token 。
  + JWT Token 刷新。
  + 获取用户基本信息（扩展 CustomUser 模型以包含头像、简介等）。
  + 修改用户基本信息。
* **非功能需求:**
  + 密码需安全存储（例如哈希处理）。
  + 接口需进行身份认证（除注册、登录外） 。

**2. 图片模块 (Images)**

* **功能需求:**
  + 图片上传（支持JPEG, PNG格式，最大50MB）。
  + 图片存储至云存储 。
  + 图片上传后触发 AI 智能分析（对象、场景、特征识别） 。
  + 存储 AI 分析结果（标签）并与图片关联 。
  + 获取单张图片信息（URL、标题、标签、上传者、创建时间）。
  + 修改图片信息（如标题）。
  + 删除图片（需要处理云存储上的文件）。
  + 获取用户上传的图片列表（支持分页、按时间排序等）。
  + （可选）图片搜索（基于标签或其他元数据），响应时间 ≤ 2秒 。
* **非功能需求:**
  + 上传接口需进行身份认证。
  + AI 分析接口为内部调用或异步处理，需考虑其响应时间（如 < 5秒）。
  + 个人图片默认私有 。

**3. 相册模块 (Albums)**

* **功能需求:**
  + 创建相册（标题、描述、可见性） 。
  + 将图片添加到相册。
  + 从相册移除图片。
  + 修改相册信息（标题、描述、可见性）。
  + 删除相册（需确认是否删除相册内图片）。
  + 获取用户创建的相册列表。
  + 获取单个相册信息及其包含的图片列表。
* **非功能需求:**
  + 相关接口需进行身份认证。

**4. 社区模块 (Community)**

* **功能需求:**
  + 将相册分享到社区（修改相册可见性为公开） 。
  + 获取社区公开的相册列表（支持排序、分页）。

**5. 系统级需求**

* **技术栈:** Django, DRF, MySQL, JWT, 七牛云, （可选）Django Channels/WebSocket 。
* **架构:** 前后端分离 。
* **API规范:** RESTful API, JSON 数据格式 。
* **安全性:** HTTPS, CORS 配置, 防范 CSRF, XSS, SQL注入, API 限流 。
* **性能:** 支持高并发（如1000+）, 关键操作响应时间 ≤ 2秒 。
* **部署:** （可选）Docker, Gunicorn + Nginx。

**二、Photox API 接口文档:**

这里提供一个更标准的 API 文档结构示例，你可以基于这个结构，结合 photox.md 中的代码逻辑和上述需求进行填充。推荐使用 Swagger/OpenAPI 规范（drf-yasg 或 drf-spectacular 可以自动生成部分文档）。

**Photox API 文档**

**基础 URL:** http://yourdomain.com/api/v1/

**认证方式:** JWT (JSON Web Token)

* 在需要认证的接口请求头中添加 Authorization: Bearer <your\_access\_token>

**通用响应格式:**

* **成功:**  
  JSON  
  {  
   "code": 0, // 0 表示成功  
   "message": "Success",  
   "data": { ... } // 具体数据  
  }
* **失败:**  
  JSON  
  {  
   "code": 1xxxx, // 业务错误码  
   "message": "Error description",  
   "data": null  
  }  
  或 DRF 默认错误格式：  
  JSON  
  {  
   "field\_name": ["Error message."]  
  }

**1. 用户认证 (Auth)**

* **POST /auth/register/**
  + **描述:** 用户注册。
  + **请求体:** {"username": "string", "password": "string", "email": "string"}
  + **成功响应 (201):** {"code": 0, "message": "User created successfully", "data": null}
  + **失败响应 (400):** 标准错误格式。
* **POST /auth/login/**
  + **描述:** 用户登录。
  + **请求体:** {"username": "string", "password": "string"}
  + **成功响应 (200):** {"code": 0, "message": "Login successful", "data": {"access": "string", "refresh": "string"}}
  + **失败响应 (400):** {"code": 10001, "message": "Invalid credentials", "data": null}
* **POST /auth/token/refresh/**
  + **描述:** 刷新 Access Token。
  + **请求体:** {"refresh": "string"}
  + **成功响应 (200):** {"access": "string"}
  + **失败响应 (401):** Token 无效或过期。

**2. 用户 (Users)**

* **GET /users/me/**
  + **描述:** 获取当前用户信息。
  + **认证:** 需要。
  + **成功响应 (200):** {"code": 0, "message": "Success", "data": {"id": "integer", "username": "string", "email": "string", "avatar": "url", "bio": "string", ...}}
* **PUT /users/me/**
  + **描述:** 更新当前用户信息。
  + **认证:** 需要。
  + **请求体:** {"email": "string", "avatar": "url", "bio": "string", ...} (允许部分更新)
  + **成功响应 (200):** {"code": 0, "message": "Success", "data": {updated\_user\_info}}
  + **失败响应 (400):** 标准错误格式。

**3. 图片 (Images)**

* **POST /images/upload/**
  + **描述:** 上传单张图片。请求体需为 multipart/form-data。
  + **认证:** 需要。
  + **请求体:** image (文件), title (字符串, 可选)
  + **成功响应 (201):** {"code": 0, "message": "Image uploaded successfully", "data": {"id": "integer", "image\_url": "url", "title": "string", "tags": [...]}} (tags 可能需要异步获取)
  + **失败响应 (400):** 文件过大、格式错误等。
* **GET /images/{image\_id}/**
  + **描述:** 获取单张图片详情。
  + **认证:** 需要（如果图片非公开）。
  + **成功响应 (200):** {"code": 0, "message": "Success", "data": {"id": "integer", "image\_url": "url", ...}}
* **PUT /images/{image\_id}/**
  + **描述:** 修改图片信息（如标题）。
  + **认证:** 需要（仅限图片所有者）。
  + **请求体:** {"title": "string"}
  + **成功响应 (200):** {"code": 0, "message": "Success", "data": {updated\_image\_info}}
* **DELETE /images/{image\_id}/**
  + **描述:** 删除图片。
  + **认证:** 需要（仅限图片所有者）。
  + **成功响应 (204):** No Content.
* **GET /images/**
  + **描述:** 获取当前用户上传的图片列表。
  + **认证:** 需要。
  + **查询参数:** page, page\_size, ordering (e.g., -created\_at)
  + **成功响应 (200):** 分页列表格式 {"count": "integer", "next": "url/null", "previous": "url/null", "results": [{image\_info}, ...]}

**4. 相册 (Albums)**

* **POST /albums/**
  + **描述:** 创建新相册。
  + **认证:** 需要。
  + **请求体:** {"title": "string", "description": "string", "is\_public": "boolean"}
  + **成功响应 (201):** {"code": 0, "message": "Album created", "data": {album\_info}}
* **GET /albums/**
  + **描述:** 获取当前用户的相册列表。
  + **认证:** 需要。
  + **成功响应 (200):** 分页列表格式。
* **GET /albums/{album\_id}/**
  + **描述:** 获取相册详情（包含图片列表）。
  + **认证:** 需要（如果相册非公开）。
  + **成功响应 (200):** {"code": 0, "message": "Success", "data": {album\_info, "images": [{image\_info}, ...]}}
* **PUT /albums/{album\_id}/**
  + **描述:** 修改相册信息。
  + **认证:** 需要（仅限相册所有者）。
  + **请求体:** {"title": "string", "description": "string", "is\_public": "boolean"}
  + **成功响应 (200):** {"code": 0, "message": "Success", "data": {updated\_album\_info}}
* **DELETE /albums/{album\_id}/**
  + **描述:** 删除相册。
  + **认证:** 需要（仅限相册所有者）。
  + **成功响应 (204):** No Content.
* **POST /albums/{album\_id}/add\_image/**
  + **描述:** 向相册添加图片。
  + **认证:** 需要（仅限相册所有者）。
  + **请求体:** {"image\_id": "integer"}
  + **成功响应 (200):** {"code": 0, "message": "Image added to album"}
* **POST /albums/{album\_id}/remove\_image/**
  + **描述:** 从相册移除图片。
  + **认证:** 需要（仅限相册所有者）。
  + **请求体:** {"image\_id": "integer"}
  + **成功响应 (200):** {"code": 0, "message": "Image removed from album"}

**5. 社区 (Community)**

* **GET /community/albums/**
  + **描述:** 获取社区公开相册列表。
  + **查询参数:** page, page\_size, ordering (e.g., -likes, -created\_at)
  + **成功响应 (200):** 分页列表格式。

**后续步骤：**

1. **完善细节：** 填充上述 API 文档结构中的具体字段、类型、约束和更详细的错误码。
2. **使用工具：** 利用 drf-yasg 或 drf-spectacular 在 Django 项目中自动生成交互式 API 文档 (Swagger UI / Redoc)。