SimpleFileServer 系统建模报告

1. 系统概述

SimpleFileServer 是一个轻量级的 Web 文件服务平台，支持用户和管理员通过浏览器访问、共享和管理本地服务器或其他用户共享的文件。系统既可以部署在内网实现 P2P 文件分享，也可以在公网访问实现跨网络文件传输。前端基于 React 实现，后端使用 Node.js 提供文件 API 接口，支持图片、视频、文本、音频、PDF、漫画等多种格式的在线预览和基本文件操作（上传、下载、删除等）。

2. 用例设计

2.1 识别和表示软件的执行者

本系统是一个轻量级的Web文件服务平台，用户间可以在内网中互传文件，也可通过部署的云端服务器互传文件；因此软件的执行者为`用户`、`管理员`、`系统`；

2.2 描述软件的用例

从执行者使用的角度来看，我们描述出以下用例：

- 用户登录系统后，会进行文件上传、预览、下载、删除等操作；但前提是必须有相应的权限，如用户在内网部署的SimpleFileServer则有管理员的权限，而用户通过互联网所访问的服务器却没有删除的权限；
- 管理员的主要职责是维护服务器的管理，允许管理员在服务器上进行批量删除、上传等操作；
- 系统的主要职责则是维护系统的运行，本系统最大的特点就是允许预览和缩略图的生成，系统必须监视文件并及时更新索引，同时为新文件生成缩略图；

软件系统的用例列表如下：

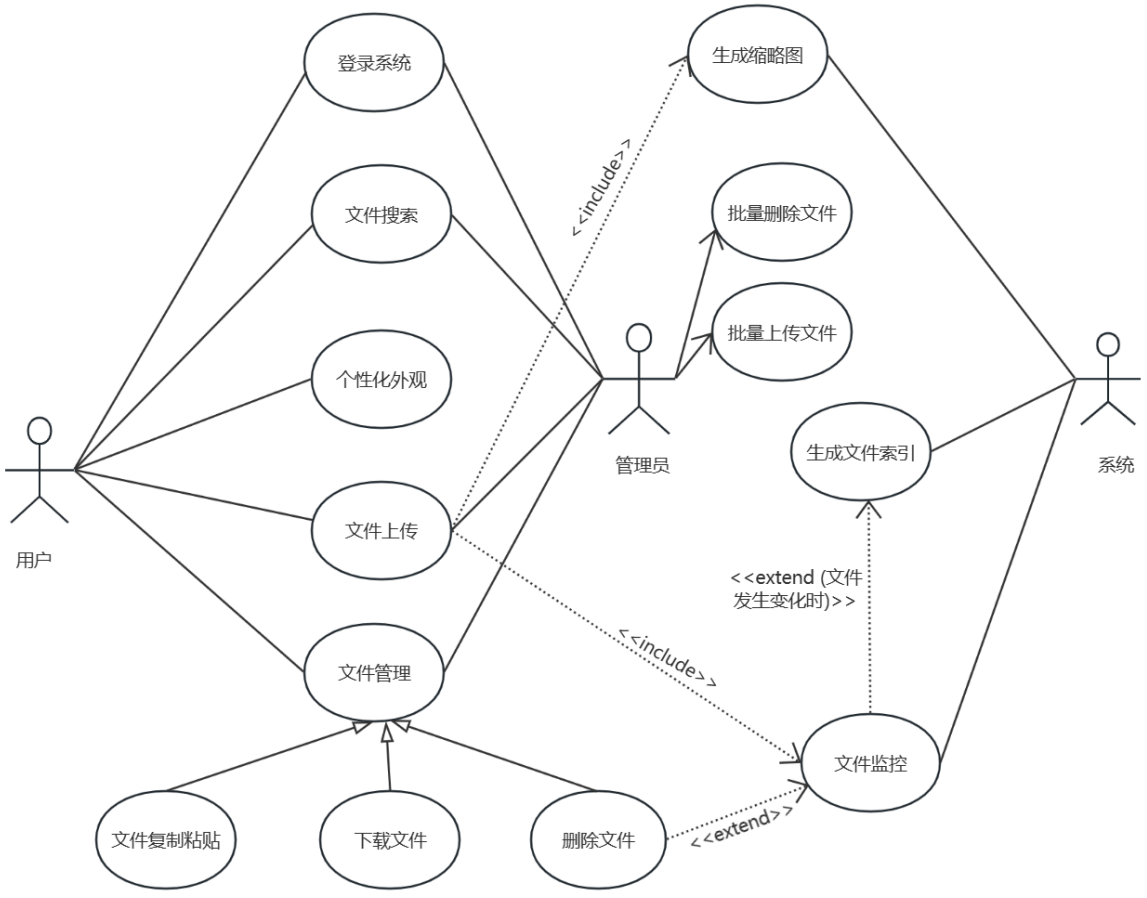
用例名称	用例标识	执行者	用例描述
登录系统	SF-Login	用户、管理员	用户通过用户名密码登录系统，获取访问权限。
文件搜索	SF-Search	用户、管理员	通过关键字在当前目录或整个索引中查找目标文件。
预览文件	SF-Preview	用户、管理员	点击 `.jpg`, `.png`, `.gif` 等图片文件，弹出图像预览器。
文件管理	SF-FileManage	用户(内网)、管理员(服务器)	对文件执行下载、删除、复制粘贴等功能；管理员可以批量删除、批量下载等；
文件上传	SF-FileUpload	用户、管理员	用户上传文件到指定目录。

用例名称	用例标识	执行者	用例描述
文件索引生成	SF-FileIndex	系统	针对所有文件生成文件索引
文件监视	SF-Supervise	系统	监视文件的变化，当删除、上传文件时为文件更新索引等
生成缩略图	SF-MiniImage	系统	为文件生成缩略图，方便辨别文件
个性化外观	SF-Personalize	用户	用户更换系统背景、调整颜色等。

根据用例列表，我们可以画出用例图如下，但注意图中有几点需要解释关系：

- **文件上传<>生成缩略图：**这是因为SimpleFileServer中，需要为每个文件生成缩略图，因此上传新文件，系统就必定会生成缩略图，因此文件上传必定会引起缩略图的生成；
- **文件上传<>文件监控：**文件监控主要是监控当前系统中文件的变化，而文件上传为系统带来新文件，那么文件监控必须要做出响应通知文件索引，因此是include的关系；
- **文件监控<>文件索引：**当有新文件上传时，文件监控会通知系统进行文件索引生成；
- **删除文件<>文件监控：**文件监控还监控权限，若当前删除该文件后，将通知系统更新文件索引；

给出用例图如下：



3. 类设计

3.1 核心类说明及介绍

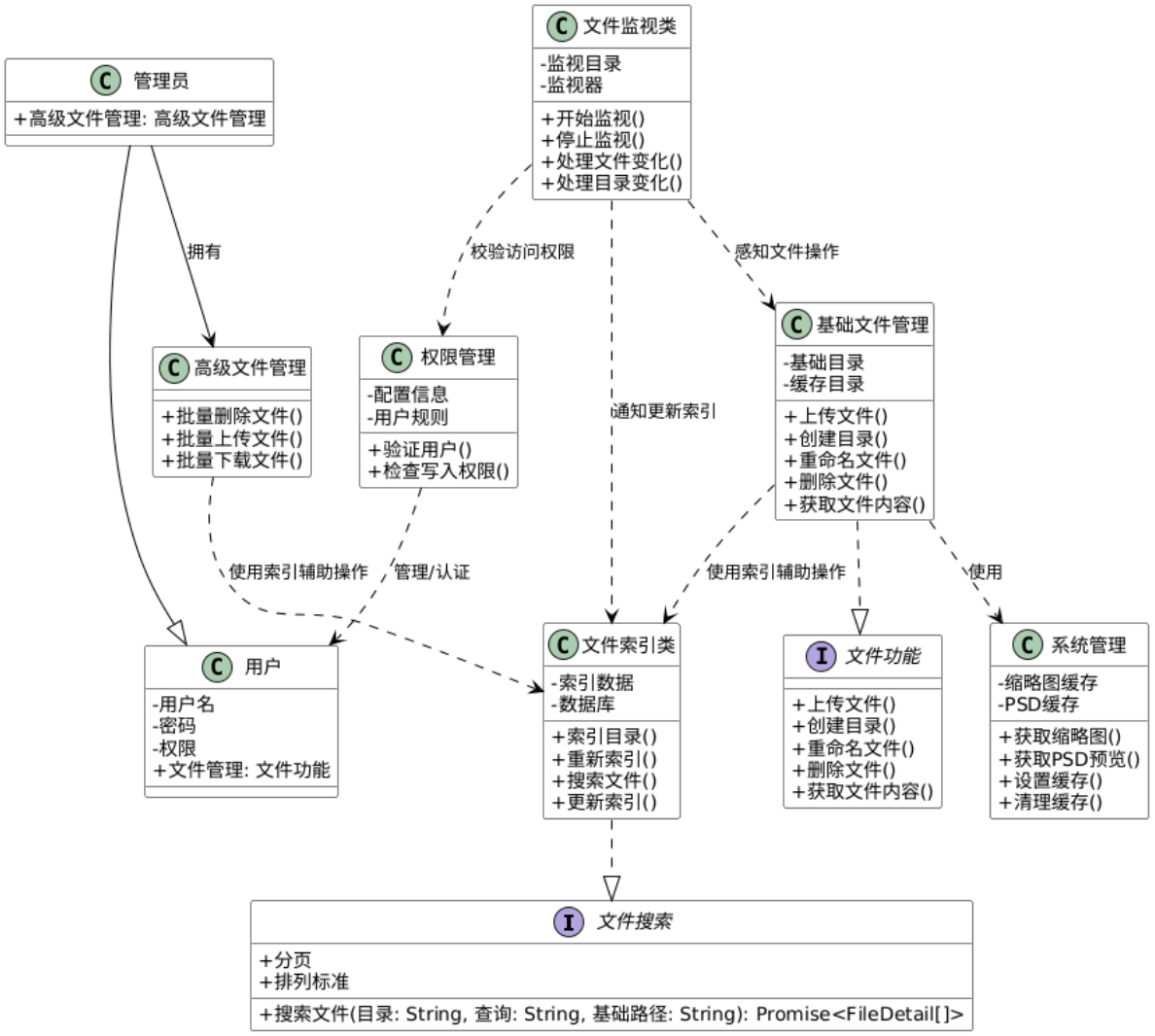
系统首先根据用例需求静态设计了以下类：

- **用户类：**即一个用户对象，负责存储用户信息以及为用户实例提供操作接口；
  - 可以看到类图，用户类中并不存在方法，这是因为功能通过引用的方式添加到了用户属性中，更好的实现了特定权限的用户有特定功能。
- **管理员类**：继承于用户类，拥有用户类功能的同时，还拥有相应的高级功能；
  - 管理员类引用了高级文件管理类，为管理员提供了特别的高级别操作；
- **基础文件管理：负责实现文件功能接口**；操作过程可能需要文件索引的辅助；
- **高级文件管理：**负责实现高级的文件功能；操作过程中可能需要文件索引的辅助；
  - 高级文件管理操作定义了管理员的额外功能，实现了相关功能和权限的独立；
- **文件监视类：**负责监视文件目录等的变化；停止文件索引更新；
  - 文件监视类有一个功能是校验访问权限，实际上是引入了权限管理的方法来实现的，因此这两个类存在依赖关系；
  - 文件监视类需要监控文件，因此需要依赖基础文件管理中的行为进行监控；
  - 文件监控还有一个重要的功能，是通知文件索引更新，因此依赖于文件索引类；
- **权限管理类**：主要负责权限信息的检查；
  - 权限管理类需要检查用户的权限，因此用户进行功能时权限管理类需要调用相关方法，因此是依赖关系；
- **系统管理类：**主要负责缩略图的生成、缓存管理等系统操作；
  - 系统管理类主要为相关操作提供了系统操作的接口，为文件的预处理等零散处理提供实现；
- **文件索引类：**主要负责生成文件索引、更新索引等功能；
  - **负责实现文件搜索接口**，为用户提供文件搜索功能；

当前系统接口设计如下：

- **文件功能**：将文件功能抽象为接口，不同的用户角色可复用接口实现不同权限的功能配置；
- **文件搜索：**文件搜索作为接口，由文件索引类负责实现，抽象为接口也可为不同的类提供可扩展的搜索功能；

3.2 类图

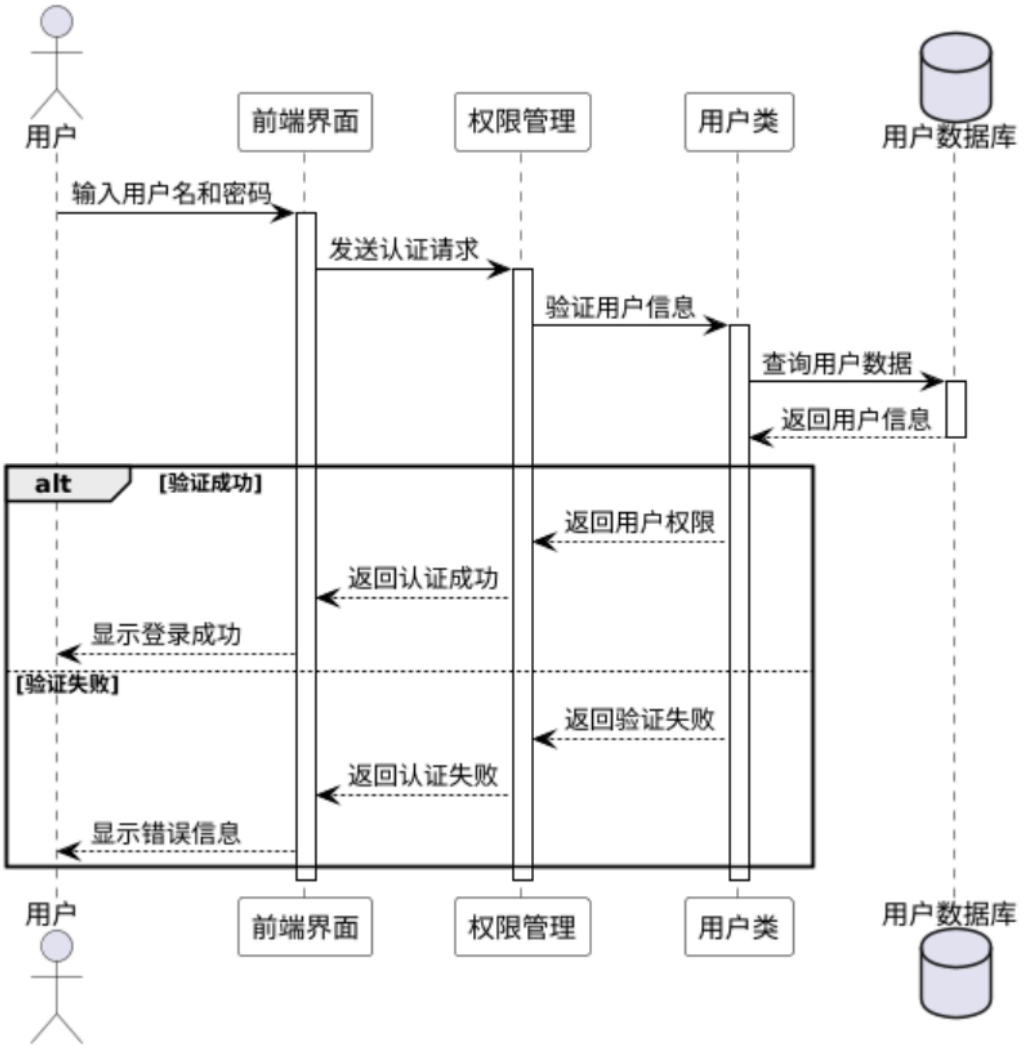


4. 序列图

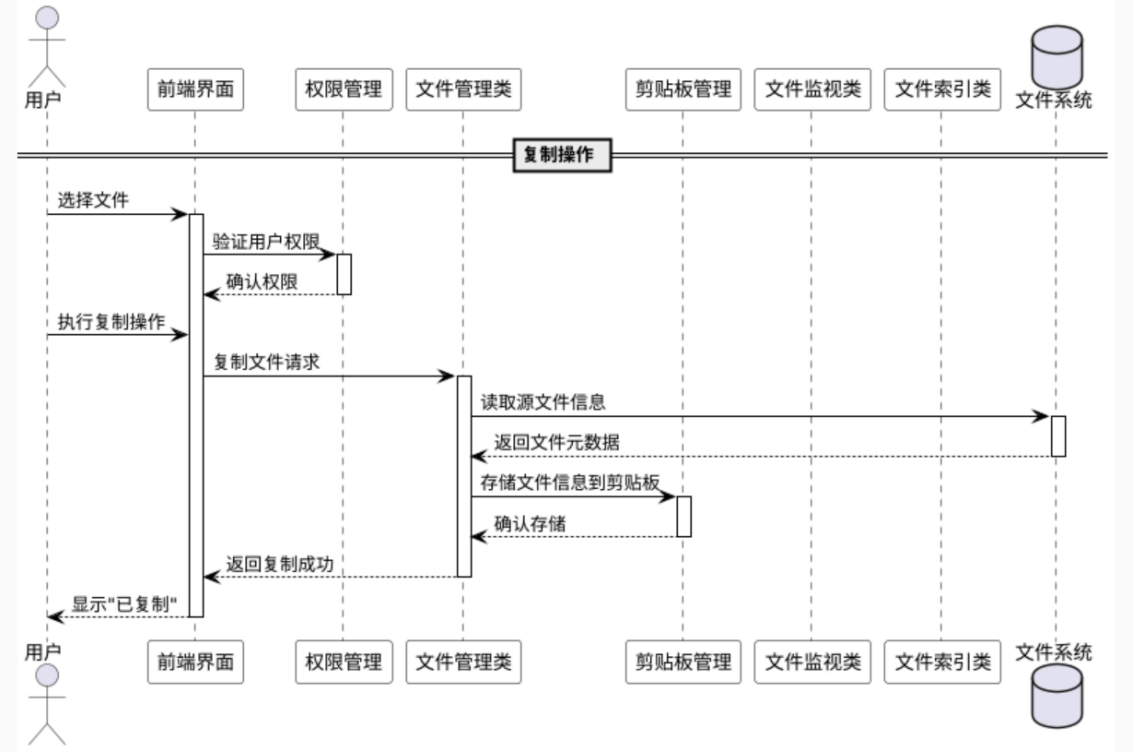
4.1 序列图及设计

根据用例设计画出系统的序列图：

1. **用户登录：**用户登录主要需要提供信息交互通知到权限管理，验证信息后返回用户实例；同时返回存在分支，即密码正确和密码错误两个分支。

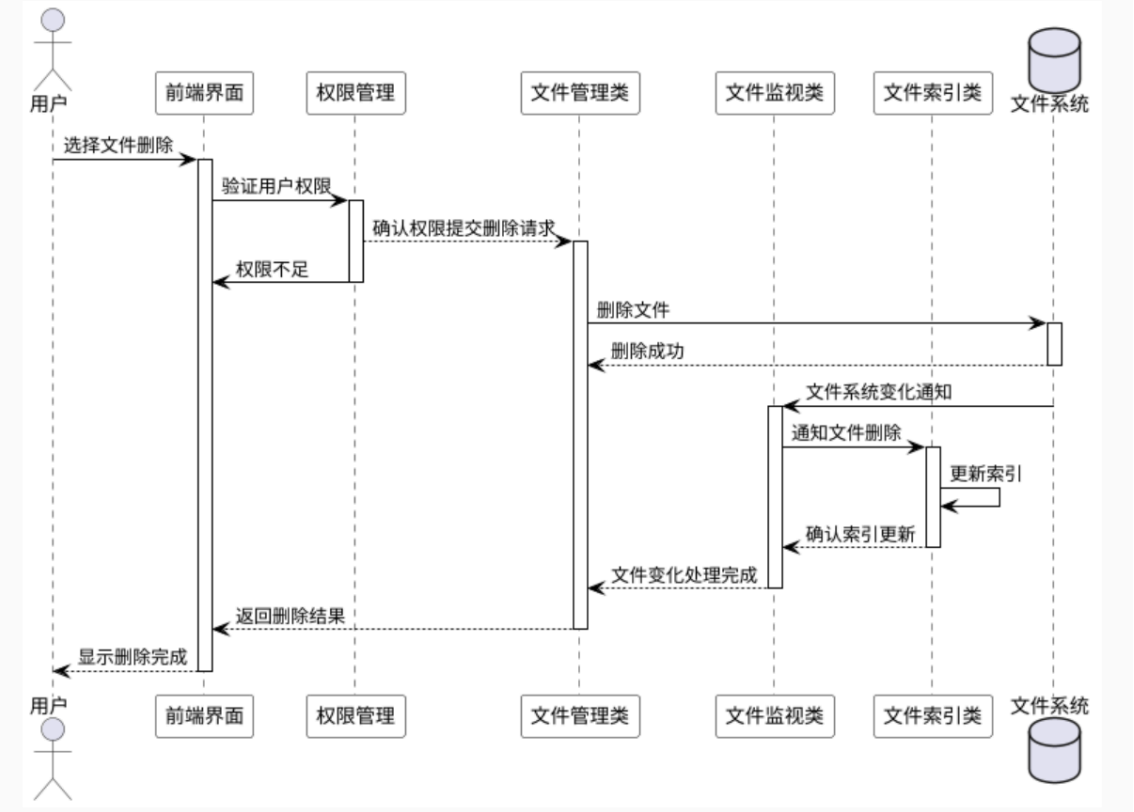


2. **文件克隆：**文件复制实际上是读操作，但由于是复制操作，比普通的读操作多了与本地剪贴板交互，剪贴板不是我们系统中的类，但是我们可以实现与本地系统的交互；

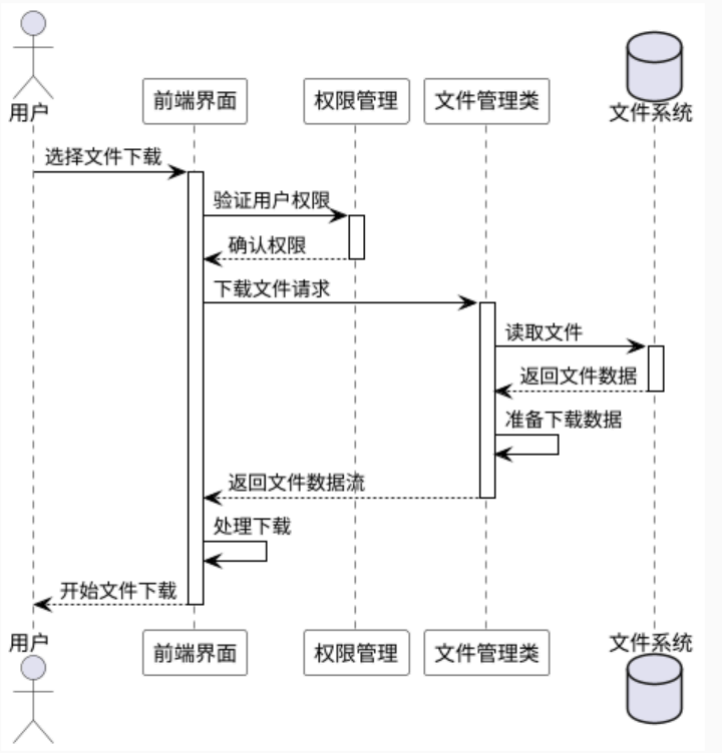


3. **文件删除：**文件删除操作必须**先验证权限**再进行，避免删除重要文件；同时删除文件后，系统会通知文件监视类，文件监视类将通知索引更新，处理结束后将返回删除流程。

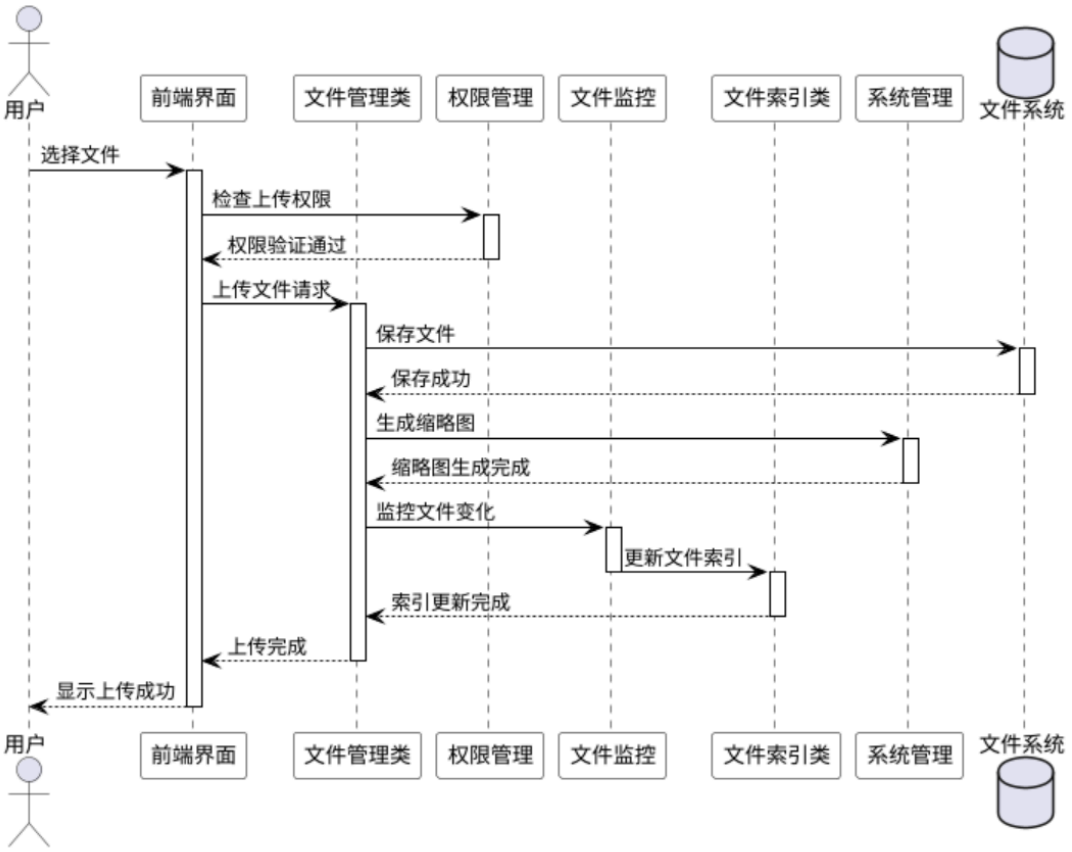
Tips:注意，删除操作的权限验证与其余操作不同，必须经过权限管理提起删除请求，而不是确认权限后由前端发起请求，这是为了安全性考虑。



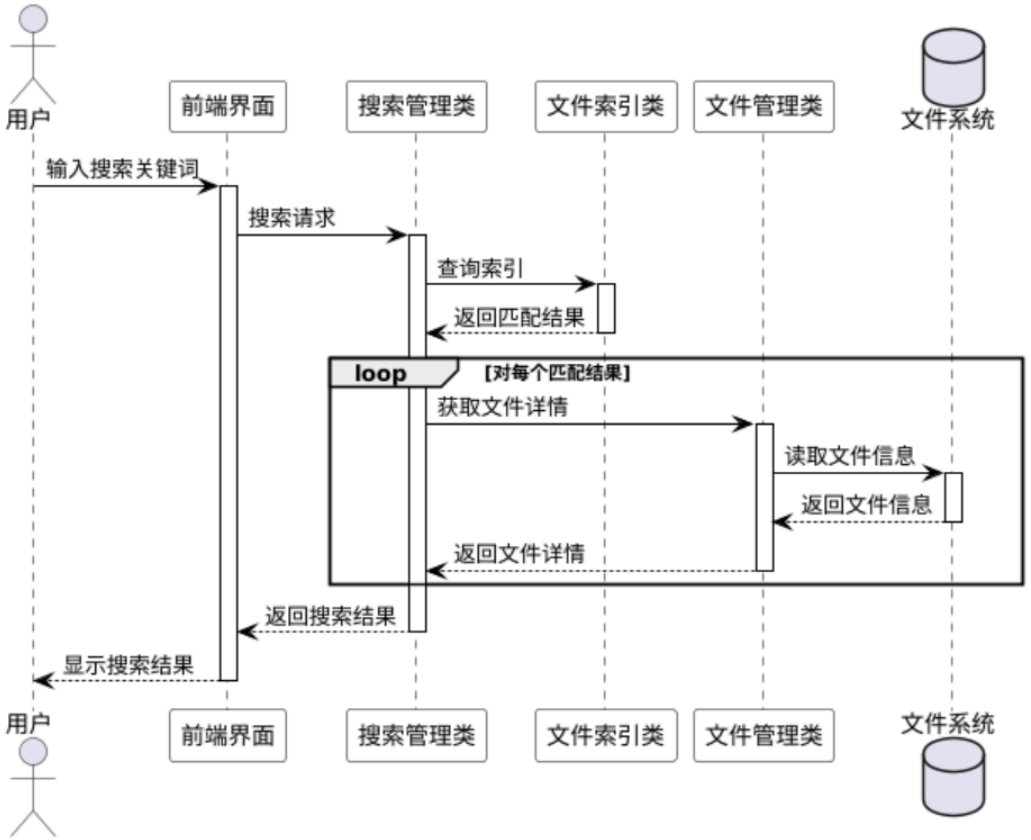
4. **文件下载：**文件下载较为简单，验证权限通过后，前端发送请求即可；



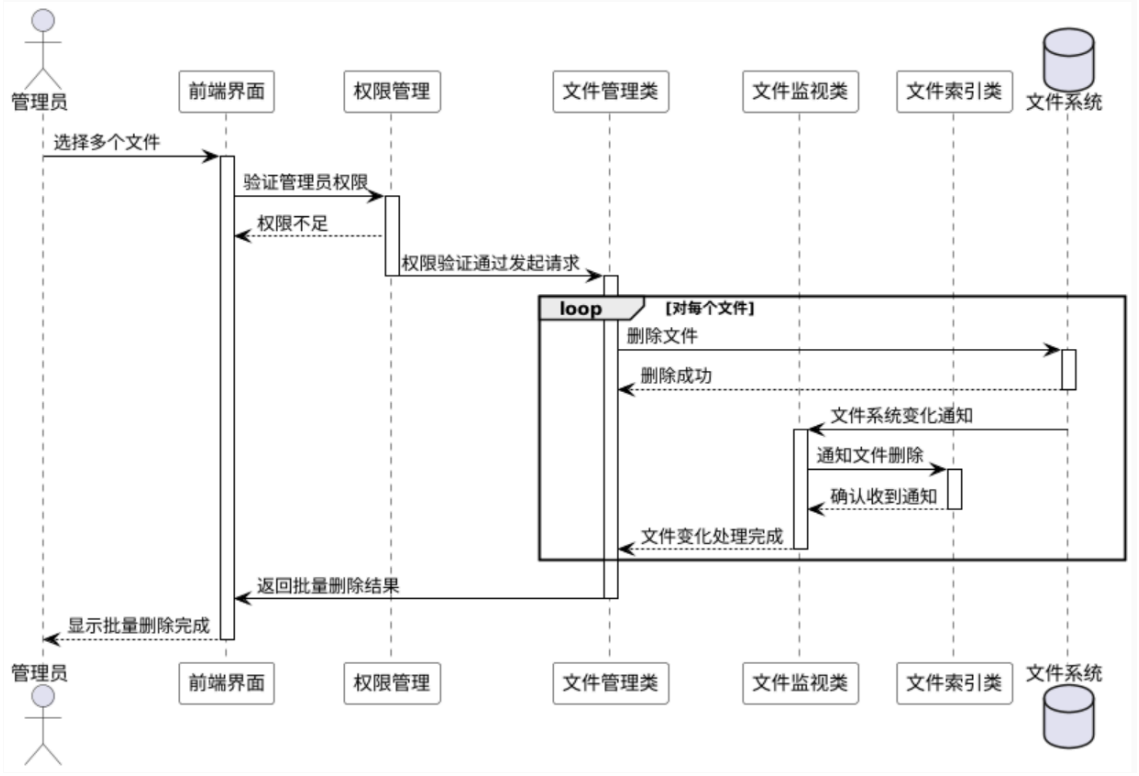
5. **文件上传**：文件上传相当于写操作，需要在文件管理类中同时通知数据库、系统管理、文件监控进行相应操作；



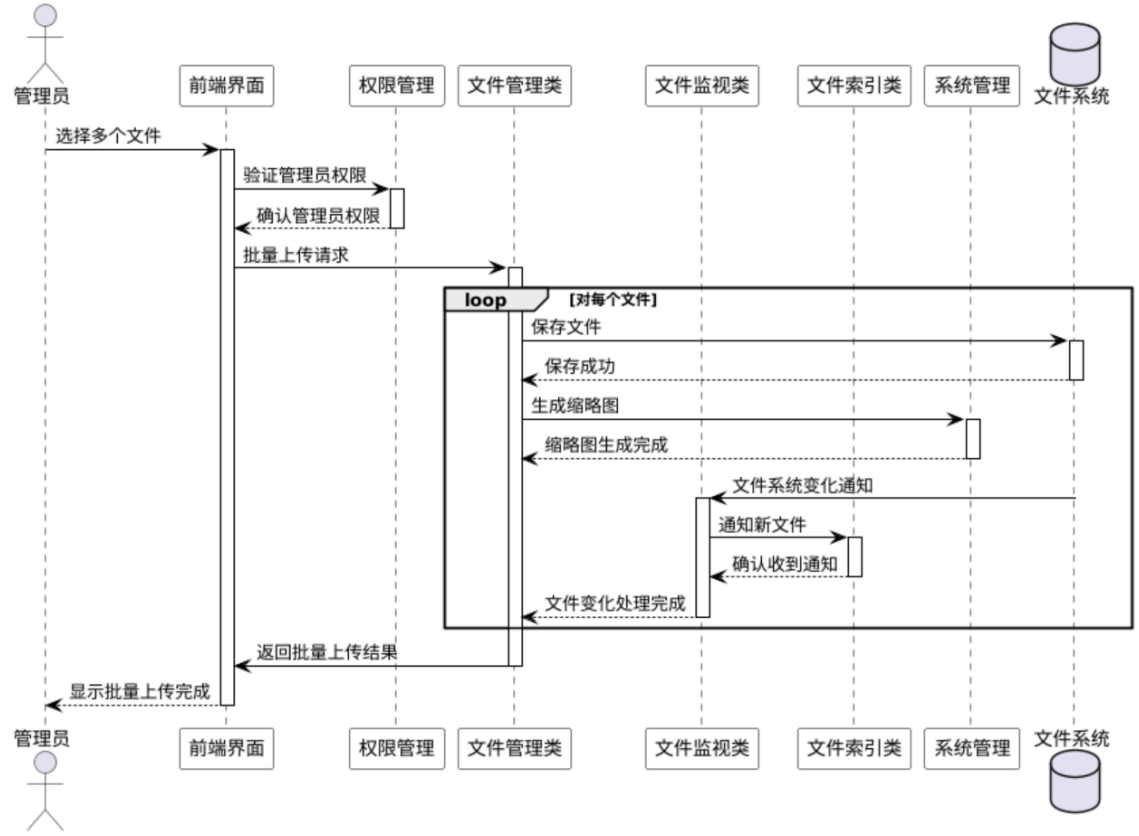
6. **文件搜索：**文件搜索可能会产生很多匹配结果，因此需要进行循环，获取每个匹配结果的信息；



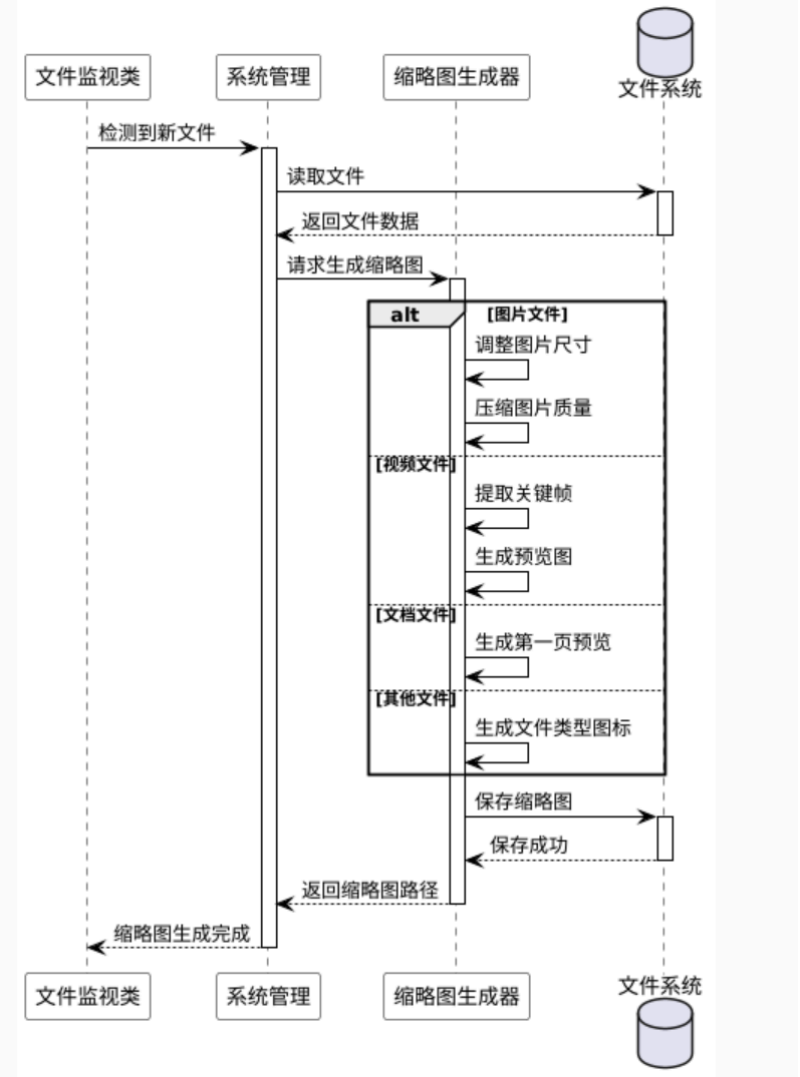
7. **批量删除文件：**批量删除文件实际上就是删除文件的循环操作，需要针对批量的文件分别进行删除；



8. **批量上传文件：**批量上传文件实际上就是上传文件的循环，针对每个文件执行上传操作；



9. **生成缩略图：**这是文件系统的重要功能之一，可以看到其起点并不是执行者，而是一个对象，这是因为它本身就是内部的一个实现，之所以拿出来是为了详细介绍其功能的实现。



5. 状态图

5.1 状态图对象

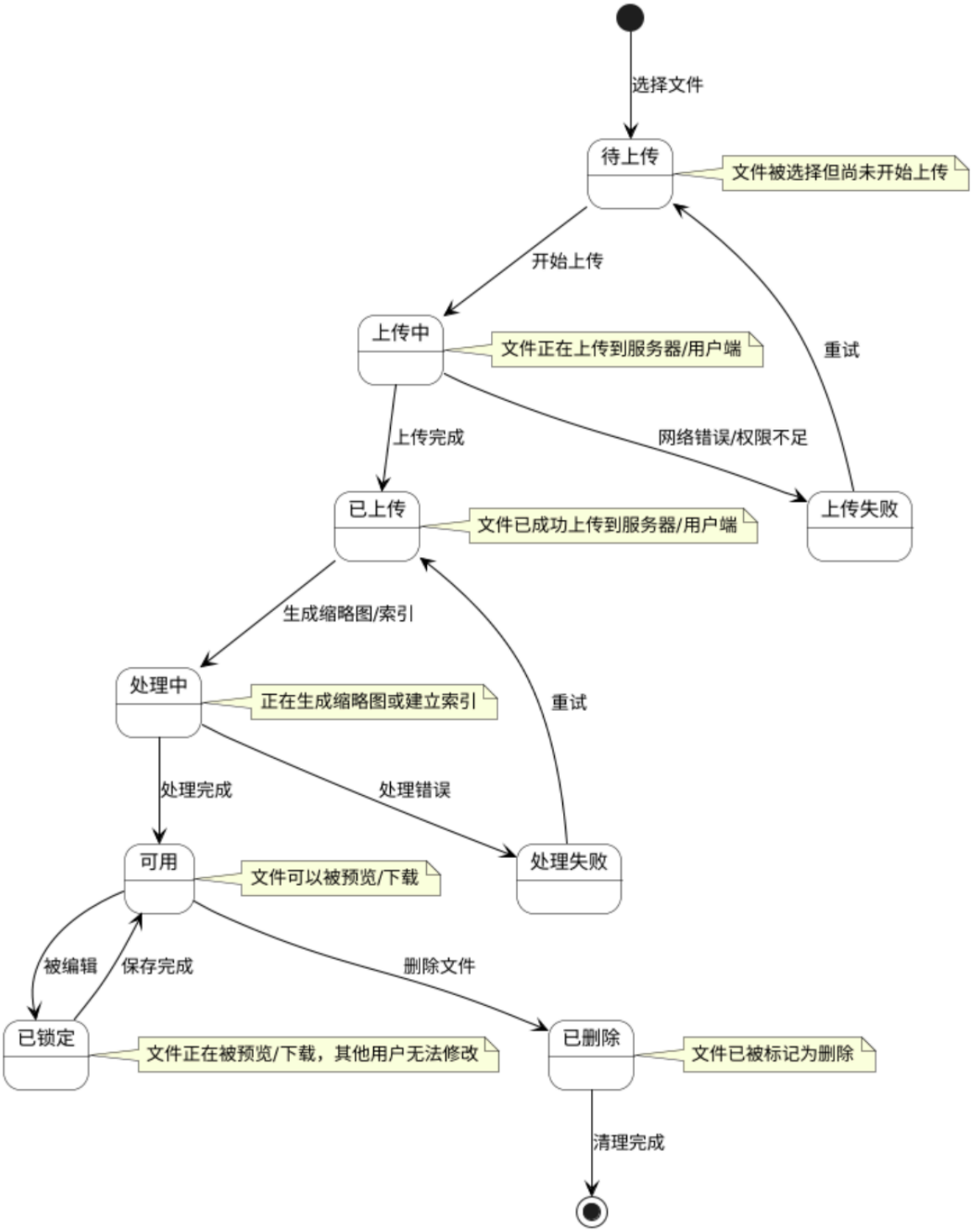
根据系统核心状态，此处给出以下两个状态图的设计：

- 文件状态图：本系统是文件系统，文件作为贯穿系统全局的对象，应该设计好其从创建到销毁的完整生命周期；
- 用户状态图：用户是本系统的主要执行者，为了理解用户的使用路径，设计状态图以更好地设计系统；

5.2 状态图

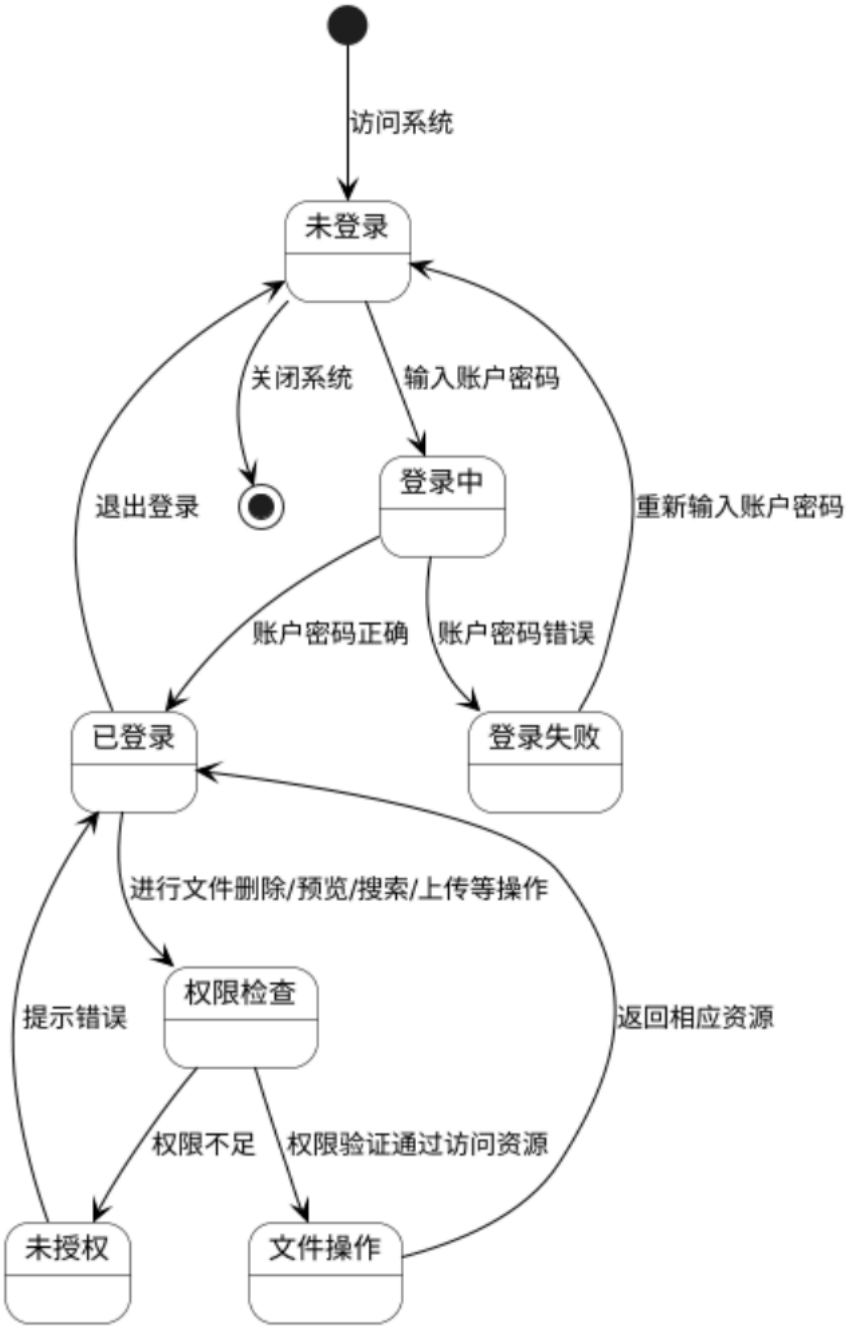
文件状态图如下：

- 可以看到，对象的每个状态都是一个文件所可能经历的；
- 需要注意的是，文件的生命周期的结束是其被删除的时候，而非用户系统关闭的时候；
- 文件的所有状态转换都是基于用户的操作，如点击按钮等；
- 文件的异常状态处理：如上传失败、权限不足等，都可返回原点重试；



用户状态图如下：

- 用户对象的生命周期从其访问系统开始到关闭系统结束；
- 用户的状态基本都与权限相关，用户进行相关操作的时候，需要进行相关检查；
- 用户出现异常情况，如账户密码错误等，都可以重新尝试；



6. 系统建模结论

从前面所给出的静态模型图和动态模型图可以看出，本系统是基于轻量级来设计的文件管理系统，整体结构清晰，模块职责明确，支持良好的扩展性与可维护性。以下对系统建模的整体特征和后续改进方向做出总结：

6.1 系统建模优点

- **职责分离明确：**通过类图与接口设计，系统将各个功能模块进行了良好的职责划分，符合单一职责原则；
- **执行者行为建模清晰**：通过用例图和序列图完整描述了用户、管理员、系统三类执行者的交互逻辑与触发机制；
- **状态驱动建模准确**：文件和用户的状态图全面展现了对象生命周期，有助于后续流程控制与异常处理设计；

6.2 系统建模过程中存在的不足

- 文件索引、文件监控等类功能过于单一，尚未完善；
- 缺少并发控制的处理，在多用户高并发的情况下未进行建模考虑；
- 执行者过于单调，当前系统仅基于公共网盘考虑，因此所有用户权限基本一致，是否可以对不同用户赋予不同权限？

6.3 展望与优化建议

- 后期系统首先需要根据本报告实现基本建模及功能；
- 在实现本报告功能基础上，根据6.2所指出的不足逐步完善；