**一、性能方面的问题与策略**

1. **问题：文件上传/下载过程中服务器I/O瓶颈严重，影响其他用户请求响应**
   * + 使用分片上传 + 并发限制机制，前端限制同时上传分片数（如3~5），后端使用异步I/O处理；
     + 引入限速模块或流量调度中间件，防止单用户上传/下载占满带宽；
     + 热门资源采用CDN缓存加速下载，减少主服务器负载。
2. **问题：高并发用户同时操作导致接口响应变慢，甚至出现宕机**
   * + 增加服务端线程池或异步队列机制（如使用Spring Boot异步控制器）；
     + 对数据库层进行读写分离，热门查询使用Redis等缓存系统；
     + 文件操作类接口加上限频措施，避免恶意操作如短时间反复上传。
3. **问题：大文件上传失败率高，影响用户体验**
   * + 采用断点续传机制（记录上传进度和文件MD5）；
     + 文件上传后端改为流式写入磁盘而非内存缓存，避免OOM；

**二、可用性方面的问题与策略**

1. **问题：界面在不同浏览器上表现不一致或不兼容**
   * + 使用HTML5标准语法开发前端，避免依赖不兼容的CSS特性；
     + 定期做主流浏览器的兼容性测试（Chrome、Edge、Firefox、移动端）；
     + 对IE或低版本浏览器使用提示“请升级浏览器”。
2. **问题：关键操作缺乏反馈，用户不确定是否成功**
   * + 所有操作如上传、删除、分享、保存等提供明确提示（成功/失败/加载中）；
     + 文件上传界面显示实时进度条和状态信息；
     + 失败操作应有重试机制并记录日志。
3. **问题：用户误操作（如误删文件）无法挽回**
   * + 所有删除操作先进入回收站，设定如“10天后自动清除”；
     + 删除或覆盖操作前弹出二次确认弹窗；
     + 可支持历史版本恢复机制。
4. **问题：移动端使用不友好，操作空间小、按钮难点**
   * + 精简移动端功能，如默认只展示预览和下载；
     + 操作按钮使用大尺寸可点击区域，适配手指操作。
5. **问题：服务崩溃或网络断开时操作数据丢失**
   * + 前端使用本地缓存机制记录未完成上传任务，支持恢复；
     + 后端使用异常捕获和事务机制保障操作原子性；
     + 系统部署高可用结构，如主从热备、定期数据备份、异地灾备。