湖 北 大 学

本 科 毕 业 论 文 （设 计）

题　　目　基于 SpringBoot 的网络文件管理系统的设计与实现

姓　　名　陈文奥

学　　号　201722111920129

专业年级　软件工程 2017 级

学　　院　计算机与信息工程学院

指导教师 /职称（可填写 1-3 人）

夏小辉　讲师

二○二一 年 5 月 9 日

目 录

基于 springboot 的网络文件管理系统的设计与实现

摘 要

早期人们要记录信息，只能依赖纸张，如果想发送信息，需要寄送邮件，信息的记录和分享效率很低。后来磁带诞生了，紧接着出现了磁盘存储介质，大大方便了数据的记录，例如我们十五年前还很常用的 U 盘，软盘。但是这些设备终究还是实体的，有设备丢失的风险，同时对于信息的分享也不太方便。互联网的出现，则将信息的分享推向了新的维度。传统的信息记录，信息分享方法已经不能很好地满足当今快节奏的社会需要。为了方便数据的存储与分享，同时保证数据的安全性，云盘这一网络产品诞生了，随着近些年云存储的发展，云盘逐渐成为主流的数据存储，数据分享的工具。几乎每一个互联网用户都使用过一到两款云存储软件。

该网络文件管理系统服务于所有对于云存储有需要的互联网用户，利用近年来越来越火的 SpringBoot 开发框架，搭配 MySql 关系型数据库，以及 HTML，BootStrap，Jquery，JavaScript 等前端技术，开发一款易用，好用的网络文件管理系统。不仅仅实现数据的上传，下载与分享这些云盘的基本功能，更要在此基础上，结合当下流行的云办公，开发一些在线文档编辑等功能，方便用户的使用。

【关键词】云存储　Idea　SpringBoot　MySql　云盘　数据存储　信息分享

The design and implementation of the network file management system based on springboot

Abstract

In the early days, people had to rely on paper to record information. If they wanted to send information, they needed to send mail. The efficiency of information recording and sharing was very low. Later, magnetic tape was born, followed by the emergence of disk storage media, which greatly facilitated the recording of data, such as the U disk and floppy disk, which were commonly used 15 years ago. But these devices are still physical after all, there is a risk of device loss, and the sharing of information is not very convenient. The emergence of the Internet has pushed information sharing to a new dimension. Traditional information recording and information sharing methods can not meet the needs of today's fast-paced society. In order to facilitate data storage and sharing, and ensure data security, cloud disk, a network product, was born. With the development of cloud storage in recent years, cloud disk has gradually become the mainstream data storage and data sharing tool. Almost every Internet user has used one or two cloud storage software.

The network file management system serves all Internet users who need cloud storage. It develops an easy-to-use and easy-to-use network file management system by using springboot development framework, MySQL relational database, HTML, bootstrap, jQuery, JavaScript and other front-end technologies. Not only realize the basic functions of data uploading, downloading and sharing, but also develop some online document editing functions based on the popular cloud office to facilitate the use of users.

【Key words】Cloud Storage Idea SpringBoot MySql Cloud Disk Data Storage Information Sharing

绪论

网络文件管理系统的研究现状

美国的芯片行业起步较早，相应的计算机行业发展也较早，得益于计算机和互联网的飞速发展，很多知名软件的雏形都在美国率先出现。网络文件管理系统的主要产品，其实就是我们熟知的网盘。网盘作为一个以分享数据，存储数据为核心功能的计算机软件，前身其实是 BT 下载。但由于 BT 下载的资源很多都没有版权，早些年的发展引起了不少用户投诉。在 2003 年，互联网用户很多用的是邮箱进行文件存储和分享，Gmail 脱颖而出作为一个以邮件为主的产品，当时却因为文件分享的功能积累了大量的用户，这说明用户们需要一个正规的资源存储和分享平台，于是云存储服务出现了。在 2007 年，谷歌推出 Gmail Drive 这一产品，网盘这才正式进入大众视野。

经过了近十年的发展，国外主流的网盘如今只剩下微软推出的 OneDrive，苹果推出的 Icloud，以及谷歌公司的 Google Drive。OneDrive 凭借 Windows 以及 word 的相容性，发展较好，与 Windows 系统的配合也极其完美，更像是内嵌在系统里的一个网络磁盘。Icloud 云盘则是苹果设备都有的附带服务。这两个网盘通过与各自的操作系统接合，体验都很不错，但它们更像是各自操作系统的附属功能。Google Drive 则凭借谷歌浏览器为用户提供了较为不错的体验。与国内环境不同，国外网盘普遍资费较高，免费空间较少，偏向于隐私性和数据安全性，主要以个人存储为主，一般也不支持在线文件编辑，压缩包解压等功能。所以，国外网盘大都是简单纯粹的云存储，主要通过用户购买存储空间来实现盈利。

国内的网盘市场发展则较晚，起步也较晚。早期 Gmail Drive 也积累了一批用户，但 2010 年 3 月，谷歌宣布退出中国市场，国内的网盘开始如雨后春笋般出现，2009 年到 2013 年期间，大量云存储系统开始瓜分国内市场，如 115 网盘，华为的 DBank，金山云盘，360 网盘，百度网盘等。各大网盘通过赠送各种免费服务抢占市场。时间到了 2016 年，由于资金以及监管压力，国内网盘迫于压力，顷刻之间大量关闭，最终只有百度云盘存活了下来。所以，百度网盘几乎可以代表当前国内云存储市场的龙头产品。不同于国外，百度网盘附加功能很多，存储空间也较大，主要依靠会员和广告实现盈利。百度网盘发展到现在，已经支持在线压缩包解压，在线文档新建，文档编辑等很多实用的功能。但由于百度网盘在国内的垄断地位，也传出过不少私自修改、删除用户文件的闲言。

研发网络文件管理系统的目的

随着互联网的普及，网盘逐渐取代了 u 盘的地位，几乎每一个人都用过网盘。利用网盘来分享数据，管理文件是极其方便，安全的。过往十年，网盘从百花齐放，变成了百度网盘的一家独大。前些年，由于各大网盘的关停，网盘的用户们用了好大劲才把自己的照片视频成功下载回本地。所以，我希望通过这次研究，自己开发一款网盘，既可以免去百度网盘的下载速度困扰，也不会担心网盘关停导致自己的数据丢失，同时还能够巩固自己积累的开发经验，检验个人的开发能力。

研发网络文件管理系统的意义

网盘市场已经发展了十几年，是一个经过了大量用户检验的市场，也为人熟知，是一个几乎每一个互联网用户都会用到的功能。由于谷歌云盘退出了中国市场，360 网盘，迅雷云盘，115 网盘，腾讯网盘也相继关停，如今的国内网盘市场已经只剩下百度网盘。虽然百度网盘已经发展的非常全面，功能丰富，但正是由于百度网盘的垄断地位，导致普通用户的使用体验越来越差，在 5g 已经快要普及的今天，文件下载速度仍然被限制的很低，此外还经常插播各种广告等。用户感觉不到幸福感。

通过开发一款新的网络文件管理系统，不仅仅能够解决上述问题，还能够提升个人的项目开发水平。同时，相比于搭建个人 NAS，自己开发的网盘可以更加灵活的添加新的功能，例如，借鉴腾讯云文档，可以加入云文档创建，编辑等功能。而随着功能的逐渐完善，用户的体验也会越来越好，逐渐成为一款小而精的网盘软件。

主要工作流程

网络文件管理系统是一个互联网软件产品，和其他软件产品类似。本文将介绍开发该系统所需要的各种技术，已经整个系统的开发流程。首先要分析需求，判断开发的可行性，然后对整个系统进行整体设计，紧接着对各个模块进行详细设计，最后进行编码和测试。

主要流程：

1.介绍开发该系统所需要的各种技术。

2.开展需求分析工作，评估该系统的开发必要性，确定该系统的主要功能，以及要用到的开发框架和开发工具等。

3.总体设计，经过充分的思考后，为该系统设计表的结构，设计功能结构。确保表结构满足系统开发需要，功能模块合理，并且具有一定的扩展性等。

4.思考每个功能模块的实现方法， 详细设计每个功能模块的实现，

5.编码实现各功能模块，组合成一个完整的系统，并测试系统的稳定性以及可用性。

6.总结与回顾。

1. 系统所用到的开发工具以及开发技术介绍

Web 软件产品是众多软件产品中的一种，也就是我们常用的互联网软件，通过浏览器访问特定网址即可使用。Web 软件的便利性，使得它非常适合用来搭建网络文件管理系统，无需安装，有网络的地方就可访问到自己的云盘文件。主流的 Web 开发分为 J2EE，Asp.net 两个阵营，而他们各自又各自分支出很多开发技术。本次开发将使用时下最为热门的 Java 语言，通过 IDEA 开发平台，基于 MVC 开发模式，通过 Spring Initial 搭建 SpringBoot 框架来进行开发，另外，该项目还将采用 Git 来进行版本控制，方便开发。相比于其他软件，网络文件管理系统所需要的前端技术并不太多，所以前端只需要 HTML，CSS，JavaScript，Jquery，BootStrap，Ajax 等技术。下面将逐一介绍该网站将会使用到的开发技术。

1.1. MVC 开发模式

该系统将会使用非常主流的 MVC 开发模式，这种开发模式能够很好地简化程序的开发，理清开发的思路。所谓 MVC，M 指的是 Model，也就是业务模型，主要负责数据的存储，发送，处理等；V 指的是 View，一般代表用户页面；C 则是控制器，控制器介于 View 和 Model 之间。三者有机统一，构成了整个应用程序。用户所能看到以及操作的主要在 View 层，通过 View 层向 Controller 层发送各种请求，例如登录请求，获取数据的请求等。作为控制层的 Controller 在接收到了请求以后，发送给对应的 Model 层，然后 View 层就可以从 Model 层获取数据，Model 也能返回一些页面给 View 层，从而展示给用户。

不难发现，MVC 模型能够通过很好的功能分化，降低程序开发的耦合度，在开发出现问题时，也能很好的定位问题，寻找问题，是一个很实用很方便的开发模型。

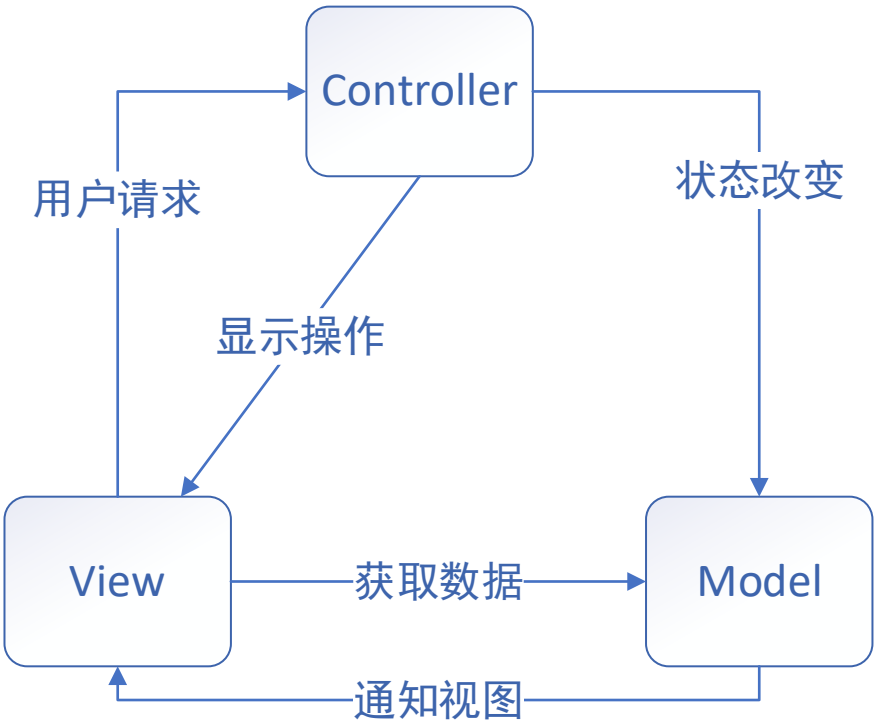

图 1-1 MVC 框架的结构

1.2. SpringBoot 开发框架

SpringBoot 是当下最为流行的开发框架，许多网站都是基于 SpringBoot 开发的。所谓 SpringBoot，Spring 指的是 Spring 框架，Boot 则指的是 Boot 引导。Spring 框架是针对大型系统提供的一个轻量级框架，凭借其 IOC 控制反转，AOP 切面编程，JDBC 的良好支持以及多种框架的兼容等，广受开发者的青睐。通过 Spring，结合 MVC 开发模式，诞生了 SpringBoot。

在传统的网站开发中，XML 文件的配置，包的添加和管理一直是一个很繁琐又枯燥的工作，而在 SpringBoot 中，只需要在 pom 文件中导入相关包的依赖，Maven 就会自动帮助开发者配置好，大大减少了开发者的工作，加快开发进度，从而实现了开箱即用的特点。通过 SpringBoot 框架，可以快速搭建 Web 程序，让精力更多地集中在功能代码上。

1.3. 开发工具介绍

1.3.1. IDEA

IDEA 作为 IntelliJ 公司的旗舰产品，一直是大多数 Java 开发者的首选编程语言开发集成环境。

从最初和 MyEclipse 平分秋色，到近些年几乎以绝对的优势成为业内最好的 Java 开发工具，IDEA 十分优秀。IDEA 拥有着丰富的代码生成功能，代码补全，代码提示，重构等功能，同时，它的轻量性与简单完善的异常捕捉能力和错误定位能力为开发者提供了巨大的便利。另外，IDEA 对 Maven 的良好支持，也更进一步拉开了 Eclipse 与它的差距。

而在开发网络文件管理系统时，IDEA 更是得心应手，通过 IDEA 内置的 SpringBoot Initial 功能，能够迅速搭建起 SpringBoot 框架，进入开发流程。

1.3.2. Tomcat 服务器容器

当我们开发完一个网站后，往往要将它部署到服务器上，才能供各地的用户进行访问，部署网站的时候，就会用到 Tomcat 服务器容器。Tomcat 是 Apache 公司推出的一款技术先进，稳定的服务器软件，一直都是 Java 开发者最为青睐的服务器。同时，而在 SpringBoot 中，已经集成了 Tomcat 程序，我们将开发的网络文件管理系统，导出为 Jar 包或者 War 包，放进 Tomcat 中，启动服务器，即可完成部署，十分方便。

1.3.3. MySql 数据库

主流的数据库种类分为关系型数据库和非关系型数据库，MySql 是关系型数据库中的佼佼者。关系型数据库一般采用表格存储数据，十分易于查询和取数据，每个数据表都包含多个字段，每个字段都有它特殊的含义。存储一些数据结构相对规范的数据十分有优势。主流的关系型数据库有 Oracle，Sql server，MySql，其中 Sql server 主要是给 Asp.net 的网站开发使用。Oracle 功能强大，但过于复杂，考虑到网络文件管理系统的需求，IDEA 的适配性，很明显，MySql 十分适合网络文件管理系统的开发。

1.3.4. GitHub

Github 是一款开源的项目版本控制软件，能够很好的记录软件开发中的变动，版本。在程序出错时能够回溯程序进行修复，在完成一个功能时，能推送变更到服务器。简单说，GitHub 能够时刻记录项目的变更情况，将项目变为阶梯式推进的一个工程，能够为软件开发提供便利。

2. 网络文件管理系统的需求分析

2.1. 问题描述

在互联网遍布各地的今天，每个人都或多或少的有一些个人数据，早些年人们习惯于用 U 盘存储数据随身携带，但随着网络的发展，如今网盘已经成为互联网用户存储数据的主流选择。网盘存储数据不用担心数据的丢失，分享数据也更加方便快捷。因此，研发一款网络文件管理系统，来方便用户进行数据存储，分享，就是本文的核心问题。

2.2. 功能性需求

结合目前市面上已经有的百度网盘，我们知道，一个网盘系统的核心功能是文件存储和文件分享，所以，做好文件管理十分重要。一个文件管理系统，至少需要有文件夹系统，回收站功能，同时要能展示文件的一些基本信息，方便用户进行文件上传和文件分类。

因此，我们可以把这个系统的功能分为三大类：①文件的管理。②文件的操作。③登录和注册，账户文件空间管理等。

通过这三大类，我们可以很好地把网络文件管理系统拆分为多个功能模块，整个系统的结构图如下：

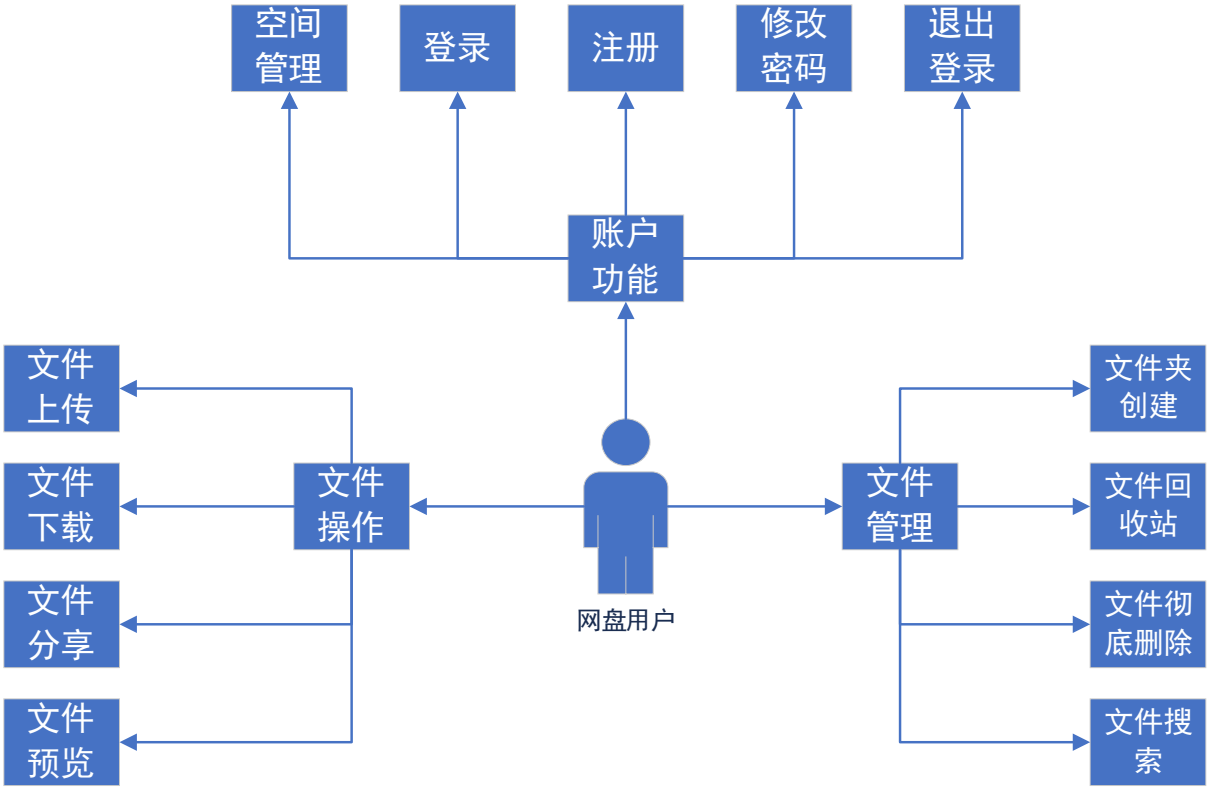

图 2-1 系统功能的总体结构

2.2.1. 文件管理功能需求

表 2-1 文件管理功能需求

功能需求	具体描述
1.文件夹创建	除了上传的文件外，系统需要允许新建文件夹，方便用户进行文件分类管理。
2.文件回收站	在用户删除文件后，不应立刻删除，而应该放入回收站，防止误删文件。
3.文件彻底删除	文件放入回收站后，用户可以访问回收站，决定是否彻底删除文件。
4.文件搜索	用户要能够在自己的文件里面搜索文件或文件夹，方便用户查询。

2.2.2. 文件操作功能需求

表 2-2 文件操作功能需求

功能需求	
1.文件上传	用户点击文件上传后，将会在当前的文件夹上传用户指定的文件。
2.文件下载	用户能够通过文件下载功能，下载指定的文件。
3.文件分享	用户要能够通过文件分享功能，生成链接分享文件。
4.文件预览	用户要能预览文件的一些基本信息，以及直接访问一些简单的文件。

2.2.3. 账户功能需求

表 2-3 账户功能需求

功能需求	
1.登录	用户通过登录功能可以进入系统
2.注册	通过注册功能新建一个账户
3.修改密码	用户能够通过手机号修改自己的密码
4.空间管理	用户有默认的最大存储空间，上传的文件不能超出账户的存储空间。
5.退出登录	用户在完成操作后，可以选择退出登录，返回登录页面

2.3. 性能需求

A．系统可扩展性 该系统要有一定的后序增加功能的能力，建表时要支持一定的后序需要，通过不断丰富功能来完善整个系统。

B．系统安全性 网络文件管理系统中，存储空间是一个很珍贵的资源，系统在给用户提供良好服

务的同时，也要保证每个用户的唯一性，防止通过创建多个账户来挤占系统的存储空间。同时，也要做到防止用户个人账户被盗，数据丢失等问题。

C．用户友好性 该网站要尽量做到界面简洁，功能可靠，增强用户的体验。

D．实时性 用户在上传文件，删除文件等操作后，要立刻能看到效果，要实时显示用户个人文件的最新情况。

3. 网络文件管理系统的总体设计

3.1. 系统总体架构

网络文件管理系统将会采用 SpringMVC 开发，主要分为控制层，显示层和逻辑层，也就是 Controller 层，View 层和 Model 层。标准的 SpringBoot 程序，后端代码将会包含四个主要类包，也就是 Bean 类，Controller 类，Service 类和 Dao 类。这几个类各司其职，由浅入深来实现用户对于个人数据的操作。

View 层主要指的是前端部分，通过编写前端页面，用户可以在页面中点击各种按钮，发送不同的请求。在项目运行的时候，Controller 层的各种函数将会接收特定的请求，通过请求来判定用户要做的操作。在得知用户要做的操作后，就可以调用 Service 层的函数来实现对应的功能，Service 层也就是服务层，主要功能是整合 Dao 层的函数，实现一个个的功能，然后为 Controller 层提供这些功能。我们知道，很多时候，现实中的一个操作并不是单纯的由一条增删改查就能解决的，Dao 层主要负责提供对数据库的增删改查功能，也就是负责操作数据库，所以我们需要用 Service 层来巧妙地整合这些功能，来将一到多个 Dao 层的增删改查组合为一个完整功能。系统架构图如下：

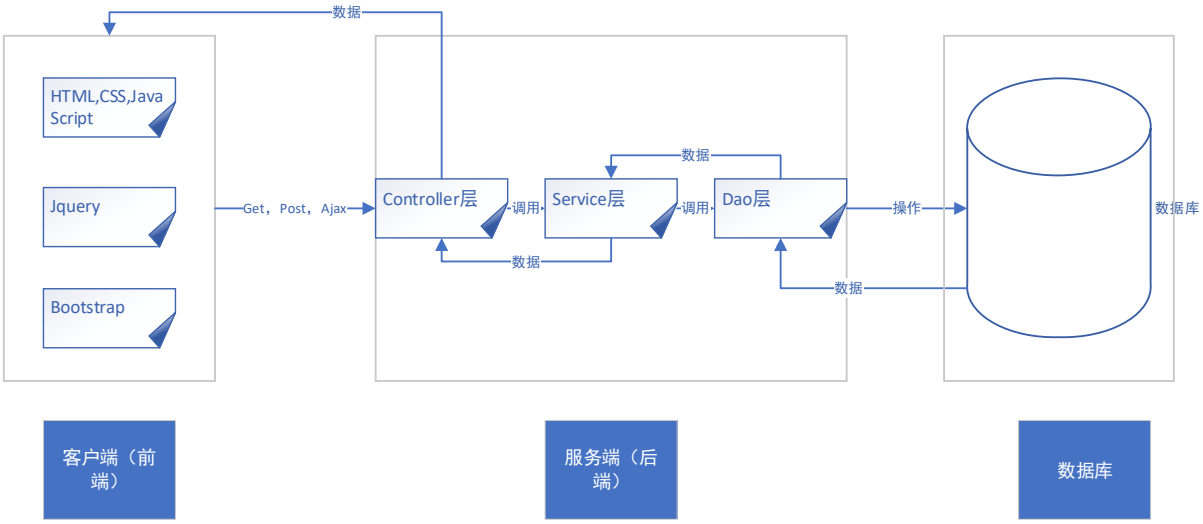

图 3-1 系统架构图

3.2. 系统功能结构的设计

需求分析阶段，我们已经确定了系统的三大核心功能模块，在这个阶段，我们要设计出系统的

总体功能结构，确定用户实用该系统的一些基本流程，来方便后序各个模块的开发和整合。以下就是该系统的功能结构图。

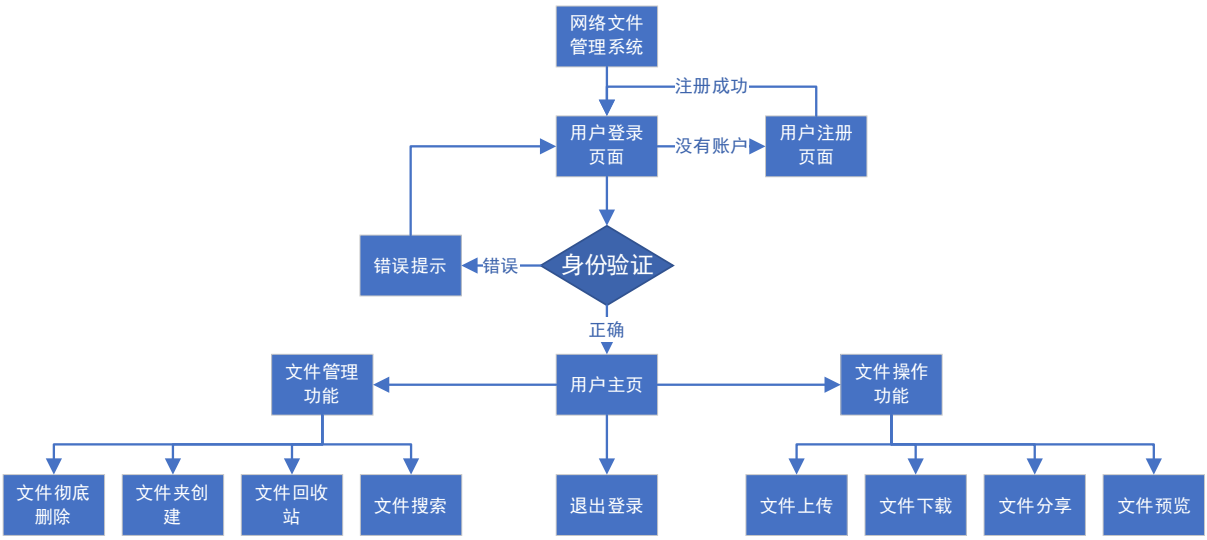

图 3-2 系统功能结构

3.3. 系统功能模块的设计

设计好网络文件管理系统的功能结构后，我们就需要详细思考这个系统的每个模块的设计。一个网络文件管理系统，是有很多模块组合而成的，每个模块负责实现特定的功能，各个模块会有不同的数据输入输出，操作结果也都各不相同，模块之间也会有或多或少的联系。我们在这一节，就需要结合业务处理逻辑和模块功能，仔细分析每个模块的设计方法，使得所有的功能模块组合在一起可以很好的配合，成为一个完整的网络文件管理系统。

3.3.1. 文件管理功能的模块设计

在需求分析阶段，我们已经把文件管理功能模块划分成为了几个子模块：文件夹创建，文件回收站，文件彻底删除，文件搜索。以下将对这四个模块进行设计。

表 3-1 “文件夹创建”模块

模块	文件夹创建
模块描述	在当前用户文件夹下创建一个新的文件夹
输入	新的文件夹名称
输出	在表中增添一条新的文件夹数据，并且显示到用户页面
操作的表	文件表
业务逻辑	通过 session 获取当前用户所处的文件夹，然后根据用户输入的文件夹名称，新建一个当前文件夹的子文件夹对象，插入到文件表中。

表 3-2 “文件回收站”模块

模块	文件回收站
模块描述	将用户选中的文件标记为已删除，放入回收站
输入	要删除的文件或文件夹的 id
输出	修改表数据，前端取消显示已经放入回收站的数据
操作的表	文件表
业务逻辑	通过 id 获取要放入回收站的文件或文件夹，将这一条数据标记为已删除，然后前端刷新文件列表，实现文件假删除效果。

表 3-3 “文件彻底删除”模块

模块	文件夹彻底删除
模块描述	将回收站中指定的文件彻底删除，无法恢复
输入	要彻底删除的文件或文件夹 id
输出	在表中删除指定数据，并且删除服务器中的指定文件
操作的表	文件表
业务逻辑	通过 id，删除服务器中存储的对应的文件或文件夹，然后删除表中对应的一到多条记录。

表 3-4 “文件搜索”模块

模块	文件搜索
模块描述	搜索指定的文件或文件夹
输入	要搜索的文件或文件夹的全部或部分字段
输出	在表中搜索包含输入的字段的文件或文件夹
操作的表	文件表
业务逻辑	获取到要搜索的字段，在文件表中查找包含该字段的数据，然后将这些数据显示到用户页面。

3.3.2. 文件操作功能的模块设计

文件操作功能模块分为：文件上传，文件下载，文件分享和文件预览。

表 3-5 “文件上传”模块

模块	文件上传
模块描述	在当前用户所在的文件夹上传一个新的文件
输入	用户上传的文件
输出	将文件存储到服务器，新的文件数据存储到数据库
操作的表	文件表
业务逻辑	通过 session 获取当前所在的文件夹，将传入的文件存储到服务器后，根据传入的文件信息，在数据库中新增一条文件信息。

表 3-6 “文件下载”模块

模块	文件下载
模块描述	用户通过文件下载模块，下载自己上传的数据。
输入	要彻底删除的文件或文件夹 id
输出	在表中删除指定数据，并且删除服务器中的指定文件
操作的表	文件表
业务逻辑	用户点击下载按钮后，后端获取到指令，然后从数据库中取得文件信息，找到服务器中对应的文件，发送给前端，前端通过下载器接收。

表 3-7 “文件分享”模块

模块	文件分享
模块描述	用户分享自己上传的文件
输入	要分享的文件或文件夹 id
输出	一个生成带有分享码的文件分享网址
操作的表	文件分享表
业务逻辑	通过 id 获取要分享的文件后，在数据库中找到对应的文件信息，通过加密生成一个文件分享网址，并且随机生成一个文件分享码，发送给页面。

表 3-8 “文件预览”模块

模块	文件预览
模块描述	用户预览指定文件的详细信息，对部分文件支持编辑，在线观看等
输入	用户要预览的文件或文件夹 id
输出	一个指定的文件或文件夹的预览页面
操作的表	文件表
业务逻辑	通过 id，在数据库中获取指定文件的详细信息，生成一个文件预览页面，返回给用户。

3.3.3. 账户功能的模块设计

账户的主要功能模块有：登录，注册，修改密码，空间管理，退出登录。

表 3-9 “用户登录”模块

模块	用户登录
模块描述	用户通过账号密码进行登录操作
输入	用户的账号和密码
输出	账户校验结果
操作的表	用户表
业务逻辑	通过账号查找用户表中对应的用户，比对输入的密码是否正确。

表 3-10 “用户注册”模块

模块	用户注册
模块描述	用户输入一些基本信息，注册一个新账户
输入	注册信息
输出	注册结果
操作的表	用户表
业务逻辑	通过用户输入的信息，在用户表中查找该用户是否已经注册过，如果没有注册过本系统，便可注册成功，返回登录页面。

表 3-11 “修改密码”模块

模块	修改密码
模块描述	用户通过输入账号注册信息来修改密码
输入	用户要修改的新密码和账户信息
输出	修改结果
操作的表	用户表
业务逻辑	通过输入的信息，判断用户是否为本人，若是本人，就将密码更新为新密码，否则返回修改失败提示。

表 3-12 “空间管理”模块

模块	空间管理
模块描述	每个用户的账户都应该有个指定大小的空间，上传的文件不能大于该空间
输入	无
输出	设置账户默认空间
操作的表	用户表
业务逻辑	每个用户新建账户的时候，都会有一个默认空间大小，可以通过一定渠道获取更多空间，同时，用户的总文件大小不能超过该最大空间。

表 3-13 “退出登录”模块

模块	退出登录
模块描述	用户通过指令退出当前登录的账户
输入	退出登录指令
输出	用户登录页面
操作的表	无
业务逻辑	清空用户登录的 session，根据拦截器，自动跳转回系统登录页面。

3.4. 数据库设计

3.4.1. 数据库总体结构设计

相比于其他的网络系统，网络文件管理系统的数据库结构十分清晰，根据系统的功能结构，用户需要能够登录注册，所以需要设计一个用户表，用来存储用户的账户信息，同时这个表也将是该用户的唯一账户，每个用户将会有多个文件或者文件夹，每个文件夹也可以被分享，生成一条分享数据。根据这一逻辑结构，很容易就能把数据表结构抽象出来，即：文件分享表，文件表，用户表。其中，文件分享表通过文件 id 与文件表关联，文件表则通过用户 id 与用户达成关联，用户表则为最基础的表，也是网络文件管理系统的核心表。

基于上面的分析，考虑到表的设计至少要满足需求分析的所有功能，我们可以把系统的实体抽象为：用户，文件，分享信息。这三个实体也就是对应的三张表，然后可以根据系统功能，详细设计出三个表的字段，抽象出表设计的 E-R 模型。

表 3-14 数据库的表设计

实体	属性
用户	{用户 id，用户名，密码，邮箱，手机号，创建时间，最大空间，已用空间}
文件	{文件 id，文件名，上传时间，更新时间，文件类型，文件大小，文件状态，文件存储位置，文件用户 id，文件母文件夹 id，文件路径}
分享信息	{分享文件 id，分享用户 id，文件分享时间，文件分享码，文件分享时长}

根据表结构以及系统功能的逻辑关系，很明显可以得知，用户与文件是一对多的关系，文件对于分享也是一对多的关系，如此，我们便可以得出 E-R 图：

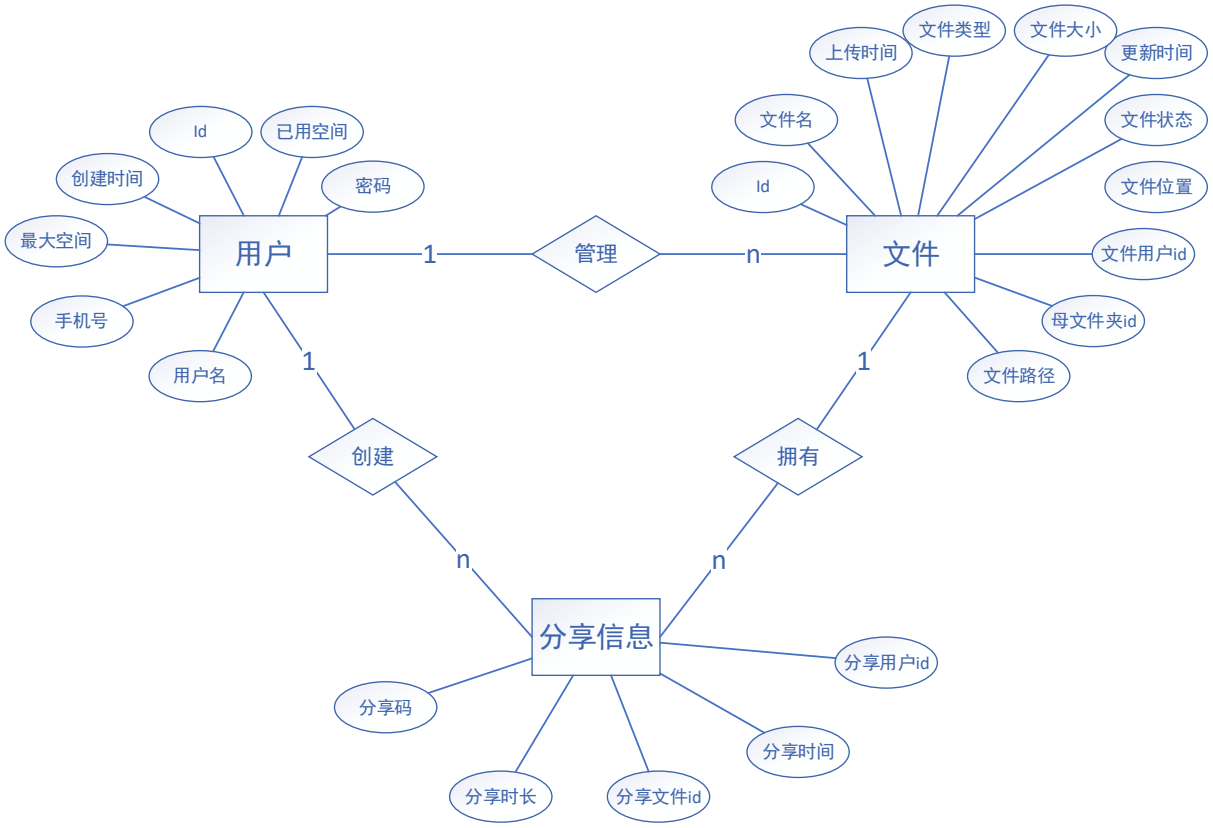

图 3-3 系统总 E-R 图

3.4.2. 数据库逻辑结构设计

现在已经设计出了数据库的总体结构，根据系统的 E-R 图，可以得知用户，文件，和分享信息之间的联系。另外，文件与文件夹实质上并没有太多本质上的区别，都可以用文件表来进行存储。用户与文件是一对多的关系，用户，文件对于分享信息也是一对多的关系，所以我们很容易可以设计出数据表的逻辑结构图：

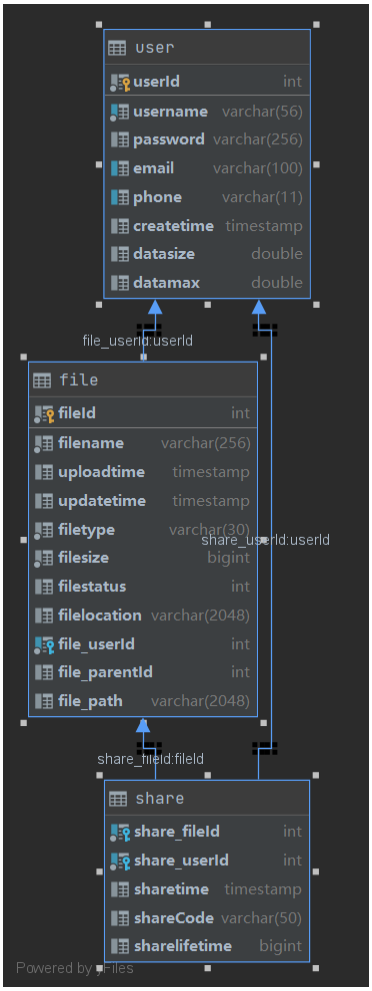

图 3-4 数据表的逻辑结构图

3.4.3. 数据库物理结构设计

这个阶段只需要将数据库的逻辑结构设计转化为表设计，要详细考虑每个表字段的属性，同时，还要考虑到表之间的联系，例如分享信息表通过文件 id 与用户 id 与文件表和用户表建立联系，所以，要详细设计每个字段的主键，外键，默认值等属性。方便后序开发工作的进展。

表 3-15 用户表的表设计

表名	User						
字段	含义	字段类型	主键	外键	Unique	Default	Not null
userId	用户 id	int	√		√		√
username	用户名	varchar(56)			√		√
password	用户密码	varchar(256)			×	'e10adc3949ba59abbe56e057f20f883e'	

email	用户邮箱	varchar(100)			√		√
phone	手机号	varchar(11)			√		√
createTime	创建时间	timestamp				current_timestamp	
datasize	已用空间	double				0	
datamax	最大空间	double				1073741824	

表 3-16 文件表的表设计

表名	file						
字段	含义	字段类型	主键	外键	Unique	Default	Not null
fileId	文件id	int	√		√		√
filename	文件名称	varchar(256)					√
uploadtime	上传时间	timestamp				current_timestamp	
updatetime	更新时间	timestamp				current_timestamp	
filetype	文件类型	varchar(30)				‘文件夹’	√
filesize	文件大小	bigint				0	√
filestatus	文件状态	int				0	
filelocation	文件位置	varchar(2048)				‘/’	
file_userId	文件用户id	int		user.userId			√
file_parentId	母文件夹id	int				0	
file_path	文件路径	varchar(2048)				‘/’	

表 3-17 文件分享表的表设计

表名	share						
字段	含义	字段类型	主键	外键	Unique	Default	Not null
share_fileId	分享文件id	int		file.fileId			√
share_userId	分享用户id	int		user.userId			√
sharetime	分享时间	timestamp				current_timestamp	
sharelifetime	分享时长	bigint				0	

4. 网络文件管理系统的详细设计与编码实现

4.1. 项目搭建

得益于 IDEA 的强大功能，搭建一个 SpringBoot 项目十分简单。我们只需要在项目创建中，选择 Spring-Initializr，就可以很快地搭建一个空的 Spring 项目，然后我们在 pom 文件中写入以来代码，就可以引入本次项目要使用的各种框架，十分方便。具体操作如下：

1）打开 IDEA，选择 New Project，设置好组名，包名，项目名后，一个空的 Spring 项目就创建完成了，本次项目使用的 Java 版本为 11 版。

2）在项目中找到 pom.xml 文件，在这个文件中，添加配置，将这个 Spring 项目配置为 SpringBoot 项目，具体代码如下：

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.4.2</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>cwa</groupId>
<artifactId>networkdisk</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>networkdisk</name>
<description>Demo project for Spring Boot</description>
<properties>
    <java.version>11</java.version>
</properties>
<dependencies>
    <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
 </dependencies>
```

3）现在只需要在 pom 中的 dependencies 中，引入我们项目中需要的各种插件，就可以完成插件的配置。这里，以 mysql 插件举例，只需要添加如下配置，就可以引入 mysql 插件：

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
</dependency>
```

4）配置数据库，使项目能够成功访问数据库。关于项目的配置，主要在 resources 目录下的 applicatiom.properties 中进行配置，数据库也不例外，我们在这个文件中，添加如下几行代码：

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/networkdisk?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=cwa19990326
spring.jackson.time-zone=GMT+8
spring.jackson.date-format=yyyy-MM-dd HH:mm:ss
```

其中，第一条 driver 表示使用的驱动版本，第二条，也就是 url，指的是数据库的连接地址，这里连接到的是 IP 地址为 127.0.0.1 的服务器上端口为 3306 的进程，也是数据库的地址。

5）最后，我们只需要在 java 文件夹下编写后端代码，也就是 controller，bean，service，dao。前端则在 resources 下的 static 文件夹编写，就可完成整个项目的编写。其中，Bean 为数据库中的数据对应的网络文件管理系统中的类，controller 负责接收前端的请求；service 负责调用 dao 来处理各种事务，为 controller 提供服务；dao 则负责操作数据库，另外，dao 层还需要有 rowMapper 类，来实现 Bean 对数据库中字段的映射。

4.2. 用户登录注册模块的详细设计

不同于其他网站，网络云盘一般不需要额外开发首页来展示软件产品，只需要一个登录网站进入系统，用户也没有复杂的分类，但也有一些注意的要点。作为一个网盘系统，存储资源都是有限的，为了保证大多数用户的体验，我们一定要尽量做到一个用户只能拥有一个账户，这也是用户表设计所要考虑的点，所以把电话号码，邮箱设置为 unique 字段，来确保身份的唯一性。在进行代码编写之前，先作出登录注册的流程图，理清思路：

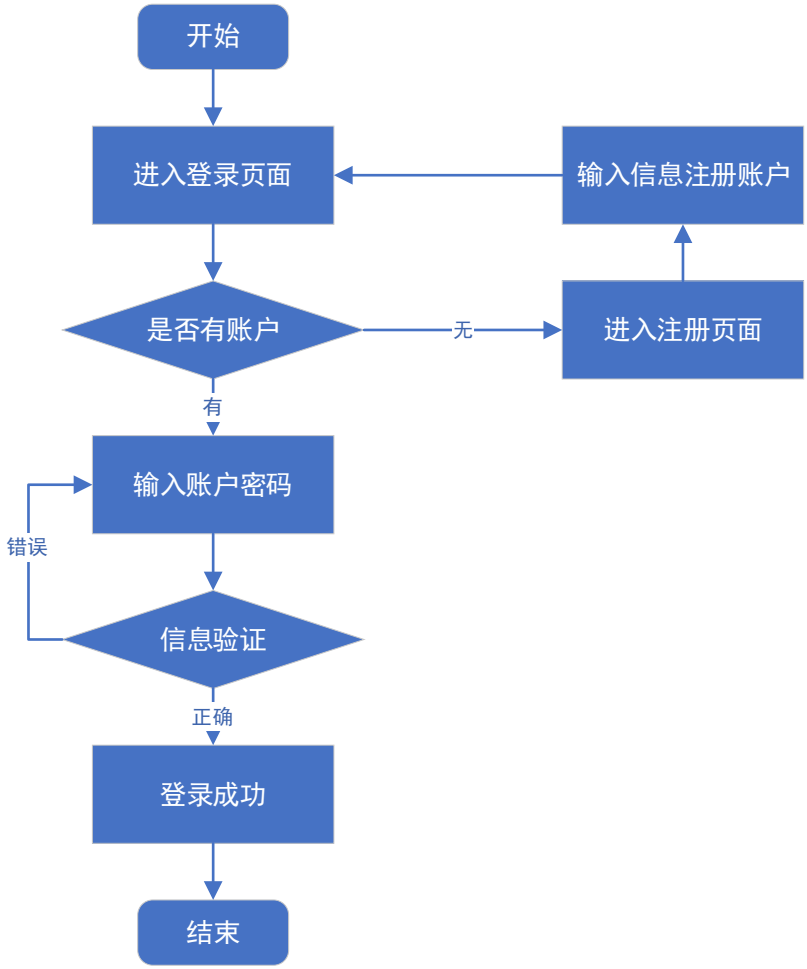

图 4-1 登录注册的业务流程图

根据业务流程图，我们需要两个页面，登录和注册，为了增强用户体验，我们可以用学习的前端知识，做一个滑块，点击响应按钮实现登录和注册的切换，部分代码如下：

```
<div class="img">
    <div class="img__text m--up">
        <h2>还未注册？</h2>
        <p>立即注册，加入我们吧！</p>
    </div>
    <div class="img__text m--in">
        <h2>已有帐号？</h2>
        <p>有帐号就登录吧，好久不见了！</p>
    </div>
    <div class="img__btn">
        <span id="registerBar" class="m--up">注 册</span>
        <span id="loginBar" class="m--in">登 录</span>
    </div>
</div>
```

对应的 CSS 代码为：

```
.content.s--signup .img__text.m--up {
    -webkit-transform: translateX(520px);
    transform: translateX(520px);
```

```
}

.img__text.m--in {
    -webkit-transform: translateX(-520px);
    transform: translateX(-520px);
}

.content.s--signup .img__text.m--in {
    -webkit-transform: translateX(0);
    transform: translateX(0);
}
```

通过以上几行代码，就可以实现登录和注册相互切换的滑块页面，点击两个 span 按钮，就可以自由切换 sign-in 和 sign-up 的两个 form 了，接下来，我们只需要编写 sign-in 和 sign-up 中的代码，实现登录注册。

4.2.1. 注册功能的详细设计

由于登录与注册大体相同，这里前端主要展示注册部分的代码：

```
<label>
    <span>账号</span>
    <input id="regUsername" name="username"/>
</label>
<label>
    <span>密码</span>
    <input id="regPassword" name="password"/>
</label>
<label>
    <span>邮箱</span>
    <input id="regEmail" name="email"/>
</label>
<label>
    <span>电话号码</span>
    <input id="regPhone" name="phone"/>
</label>
<button type="button" class="submit" onclick="register()">注 册</button>
```

点击注册按钮，触发 register 方法，register 方法代码如下：

```
//注册
function register() {
    if ($("#regUsername").val() == '') {
        toastr.warning("用户名不能为空，请重新输入！");
        $("#regUsername").focus();
        return;
```

```
    }
    if ($("#regEmail").val() == '') {
        toastr.warning("邮箱不能为空，请重新输入！");
        $("#regEmail").focus();
        return;
    }
    if ($("#regPhone").val() == '') {
        toastr.warning("电话号码不能为空，请重新输入！");
        $("#regPhone").focus();
        return;
    }
    //使用 jquery 的 serializeArray 方法，和 springBoot 的自动参数绑定
    var formData = $("#regUser").serializeArray();
    $.ajax({
        url: 'register',
        type: 'POST',
        data: formData,
        success: function (data) {
            if (data == false) {
                toastr.warning("当前用户已存在，请重新输入！");
                $("#regUsername").val("");
                $("#regPassword").val("");
                $("#regEmail").val("");
                $("#regPhone").val("");
                $("#regUsername").focus();
            } else {
                toastr.success("注册成功！");
                $("#loginBar").click();
                // setTimeout("indexLogin()", 1500);
            }
        }
    })
}
```

在点击注册按钮后，前端先会判断四个所需要的数据是否已经输入，如果没输入，会利用 toastr 提示插件进行提示，如果输入信息合法，就把 regUser 表单的数据打包，发送一个 post 请求到 register 控制器，根据控制器返回的结果，利用 toastr 弹出对应的提示。注册部分的后端代码相对简单，只需要在 UserController 中，调用 UserService 的插入 User 数据的方法，另外，UserService 还要对传入的 user 对象的密码属性进行 MD5 加密，加密存储保护用户数据安全，加密部分的代码如下：

```
//加密密码
MessageDigest md5 = MessageDigest.getInstance("MD5");
md5.update(user.getPassword().getBytes());
String password_MD5 = new BigInteger(1, md5.digest()).toString(16);
user.setPassword(password_MD5);
return userRepository.insertNewUser(user);
```

dao 层的注册方法代码如下：

```
public boolean insertNewUser(NetUser user) {
    try {
        usertemplate.update("insert into user(username,password,email,phone)
values(?,?,?,?)",
                user.getUsername(),
                user.getPassword(),
                user.getEmail(),
                user.getPhone());
        return true;
    } catch (Exception e) {
        System.out.println(e);
        return false;
    }
}
```

4.2.2. 登录功能的详细设计

登录部分的前端代码与注册大体相同，但是登录在后端需要多做一件事，就是添加当前登录的用户的 session，来记录用户登录这一操作，同时方便后序功能的实现。登录控制器的代码如下：

```
//登录功能
@PostMapping("/User/login")
public boolean login(NetUser logUser, HttpSession session) throws
NoSuchAlgorithmException {
    NetUser currentUser = userService.loginCheck(logUser);
    if (currentUser != null) {
        //设置登录用户
        session.setAttribute("currentUser", currentUser);
        //设置根目录
        NetFile currentFile = new NetFile();
        currentFile.setFile_userId(currentUser.getUserId());
        currentFile.setFileId(0);
        currentFile.setFile_Path("/");
        currentFile.setFileStatus(0);
        session.setAttribute("currentFile", currentFile);
        return true;
```

```
    } else
        return false;
}
```

我们注意到，用户登录成功之后，会添加两个 session，一个是 currentUser，另一个是 currentFile，currentUser 就是我们当前登录的账户的相关信息，currenFile 则为当前用户所在的文件夹，用户登录后，默认是在根目录，所以我们只需要设置 currentFile 的部分属性，使它代表根目录即可。

我们还需要注意，一个正规的操作系统，应该是需要有拦截器的，防止用户直接在浏览器输入 url 来访问 controller，所以，我们还需要创建一个拦截器，如果用户还没登录，这个拦截器会拦截下除了登录注册以外的其他请求，来实现系统的安全。拦截器实现如下：

```
//访问接口之前执行。
public boolean preHandle(HttpServletRequest request, HttpServletResponse
response, Object handler) throws Exception {
    HttpSession session = request.getSession();
    NetUser currentUser = (NetUser) session.getAttribute("currentUser");
    //如果 session 中没有 loginUser，表示没登陆
    if (currentUser == null) {
        //这个方法返回 false 表示忽略当前请求，如果一个用户调用了需要登陆才能使用的接口，
如果他没有登陆这里会直接忽略。
        //重定向到登录页面。
        response.sendRedirect("/User/login");
        return false;
    } else {
        return true;    //如果 session 里有 user，表示该用户已经登陆，放行。
    }
}
```

实现逻辑其实就是判断 session 中是否有 currentUser 数据，至于要拦截哪些页面，还需要我们另外建一个类来注册拦截器：

```
//注册拦截器
@Override
public void addInterceptors(InterceptorRegistry registry) {
    // addPathPatterns("/**") 表示拦截所有的请求，
    InterceptorRegistration registration =
            registry.addInterceptor(loginInterceptor);
    registration.addPathPatterns("/User/**");
    registration.excludePathPatterns("/User/login");
    registration.excludePathPatterns("/User/register");
    WebMvcConfigurer.super.addInterceptors(registry);
}
```

这个拦截器首先拦截所有的请求，然后将 login 和 register 作为例外，防止拦截到注册和登录请求。如此，登录注册部分就基本完成了，实现效果如下：

图 4-2 toastr 提示框的实现效果

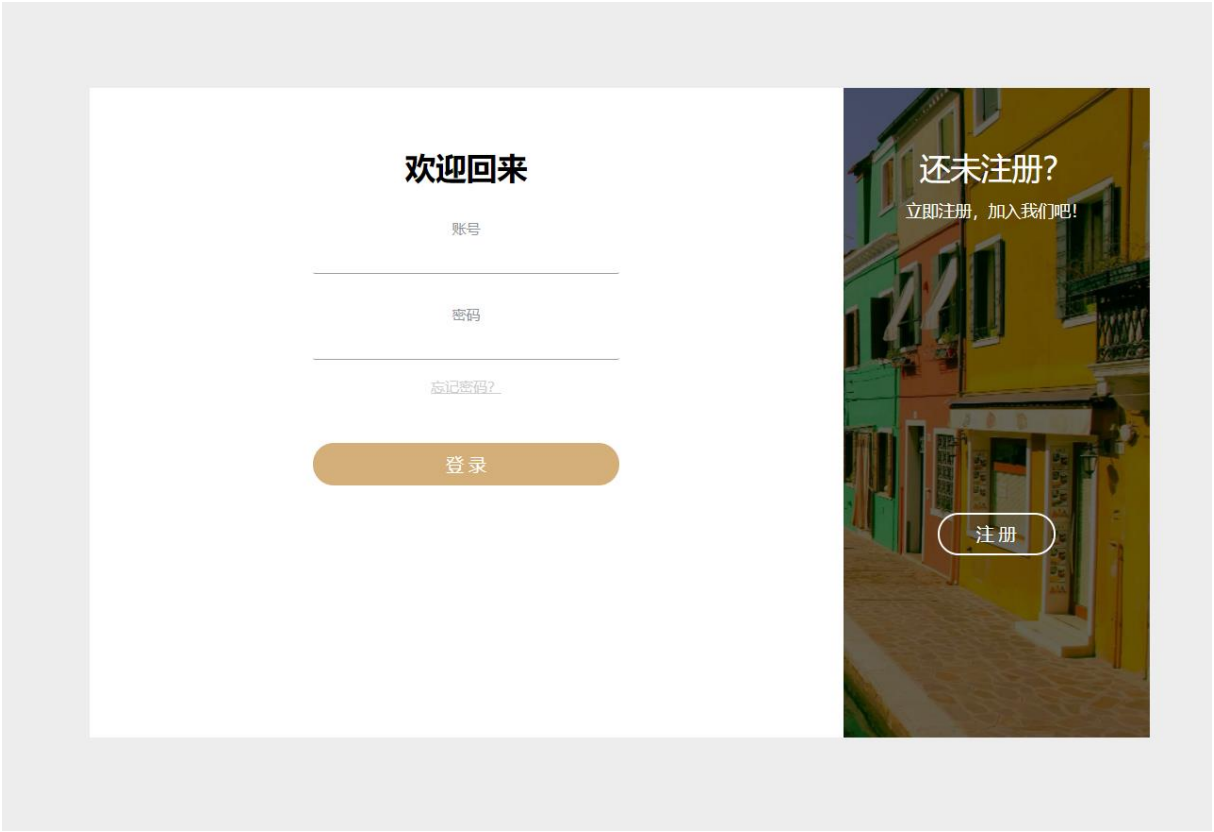

图 4-3 登录页面

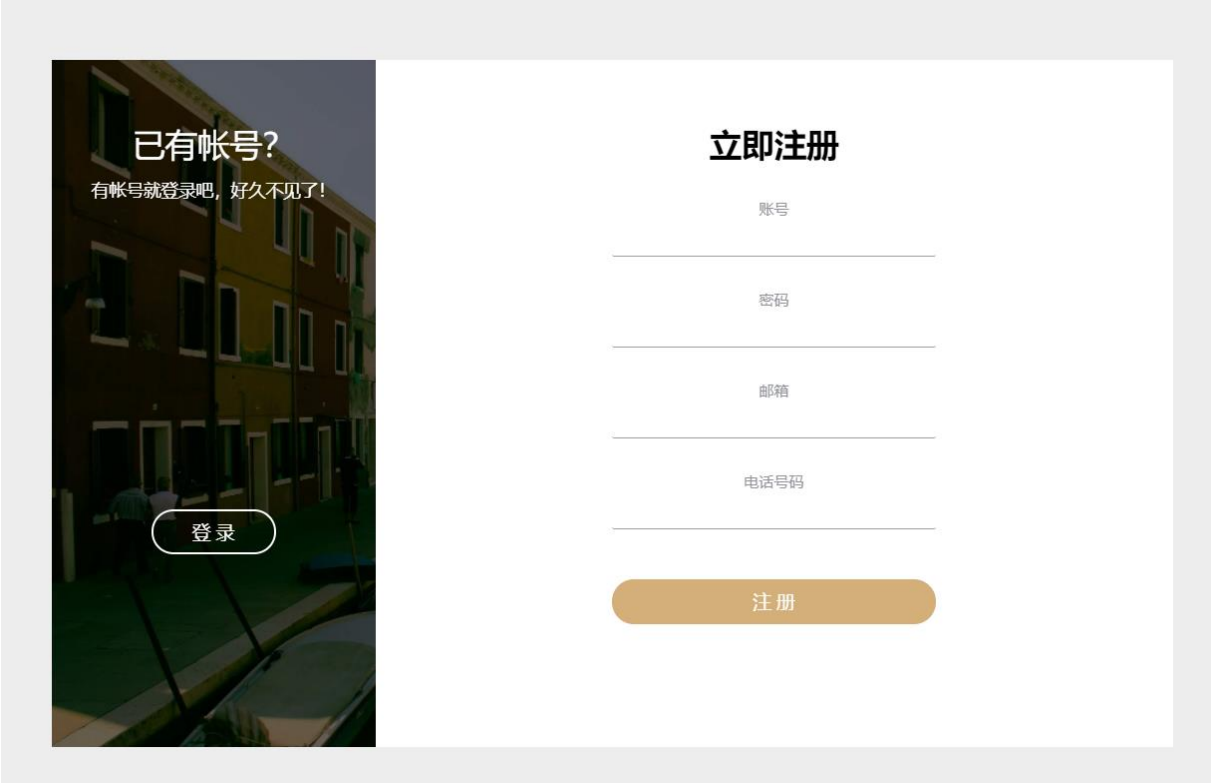

图 4-4 注册页面

4.3. 文件管理的详细设计

在编写主页之前，我要先介绍一下在登录模块部分，我们在 session 中添加 currentFile 的用处，currentFile 其实就是代表当前用户所处的文件夹，这样，前端只需要发送一个 refreshData 的请求，就可以查找 currentFile 文件夹下所包含的所有文件，实现数据刷新的效果。比如，我们在删除一个文件，或者上传一个文件之后，调用一次 refreshData，就可以实现数据更新。我们前往一个文件夹，也只需要更改 currentFile，然后调用 refreshData，就可以巧妙的实现进入子文件夹的功能。

在文件管理详细设计部分，我将主要介绍文件显示，新建文件夹，上传文件，下载文件，分享文件的详细实现。

4.3.1. 文件显示功能的详细设计

要管理文件，首先要展示文件，这里我们在前端页面添加一个表格作为文件夹列表，但是表格中没有静态数据，而是通过 ajax 动态刷新这个文件表格，来进行添加数据。HTML 代码如下：

```
<table class="table table-striped">
    <thead>
    <tr><th>名称</th>
        <th>修改日期</th>
        <th>类型</th>
        <th>大小</th>
```

```
        <th>操作</th>
    </tr>
    </thead>
    <tbody id="fileList"></tbody>
</table>
```

对应的刷新数据要用的 refreshData()方法如下：

```
//刷新页面数据
function refreshData() {
    $("#fileList").empty("");
    $.ajax({
        type: "get",
        url: "/File/refreshData",
        success: function (data) {
            $(data).each(
                function (i, oneFile) {
                    $("#fileList").append(
                        "<tr>" +
                        "<td><a onclick='clickFile(\"" + oneFile.fileId +
"\",\"" + oneFile.fileType + "\")'>" + oneFile.fileName + "</a></td>" +
                        "<td>" + oneFile.updateTime + "</td>" +
                        "<td>" + oneFile.fileType + "</td>" +
                        "<td>" + (oneFile.fileSize / 1048576).toFixed(2) + "MB"
+ "</td>" +
                        "<td><button type='button' class='btn btn-info'
onclick=\"window.open('/File/downloadFile/" + oneFile.fileId + "')\">下载
</button></td>" +
                        "<td><button type='button' class='btn btn-warning'
onclick='fileDetail(" + oneFile.fileId + ")'>预览</button></td>" +
                        "<td><button type='button' class='btn btn-warning'
onclick='shareFile(" + oneFile.fileId + ")'>分享</button></td>" +
                        "<td><button type='button' class='btn btn-danger'
onclick='deleteConfirm(" + oneFile.fileId + ")'>删除</button></td>" +
                        "</tr>"
                    );
                }
            );
        },
    })
}
```

在 refreshData()方法中，前端只需要向"/File/refreshData"端口发送一个请求，这个 controller 就会查询出 currentFile 文件夹下的所有文件，返回给前端，前端获取到数据后，将数据添加到表格里面。这样一来，我们在进入主页后，由于在登录时已经添加了 currentFile 根目录，只需要调用一次 refreshData()就可以显示出数据。最终实现的效果如下：

图 4-5 文件夹列表

4.3.2. 新建文件夹功能的详细设计

为了美化文件夹名称输入弹窗，我们在主页中安插了一个隐藏的 div 模态窗，代码如下：

```
<div class="modal fade" id="newFoldBox" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>
                <h4 class="modal-title" id="myModalLabel">新建文件夹</h4>
            </div>
            <div class="modal-body">
                <div class="input-group">
                    <span class="input-group-addon">文件名：</span>
                    <input id="newFileFoldName" type="text" class="form-control" placeholder="请输入文件夹名称">
                </div>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismiss="modal">取消</button>
                <button type="button" class="btn btn-primary" onclick="newFileFold()">新建</button>
            </div>
        </div>
    </div>
</div>
```

这个模态框主要包含一个用于获取新建的文件夹名称的 input 输入框，另外包含一个新建按钮，调用 newFileFold()方法，实现新建文件夹，常规状态下，这个模态框是隐藏的，当用户点击主页的新建文件夹按钮时，会显示该模态框，这个按钮的代码如下：

```
<button class="navbar-brand btn btn-default" data-toggle="modal" data-target="#newFoldBox">
    新建文件夹
</button>
```

新建文件夹的 ajax 请求如下：

```
function newFileFold() {
    const foldName = $('#newFileFoldName').val();
    $('#newFileFoldName').val("");
    if (foldName == "") {
        $('#newFileFoldName').attr('placeholder', "名称不能为空！");
    } else {
        $.ajax({
            url: '/File/newFileFold/' + foldName,
            type: 'GET',
            success: function (data) {
                if (data == true) {
                    $('#newFileFoldName').attr('placeholder', "请输入文件夹名称！
");
                    $('#newFoldBox').modal('hide');
                    toastr.success("新建文件夹成功！");
                    refreshData();
                } else {
                    $('#newFileFoldName').attr('placeholder', "文件夹已存在！");
                }
            }
        })
    }
}
```

新建文件夹的后端部分思路则比较简单，前端通过 ajax 的 get 请求，把要建的文件夹名称提交给后端后，后端首先拿到 session 中的当前文件夹信息，然后新建一个文件对象，这个对象的母文件夹 id 就是 currentFile 的 id，然后插入到数据库中即可。新建成功后，会返回 true，前端得到 true 后，采取相关操作并调用 refreshData()，实现数据刷新。后端部分代码如下：

```
//增
// 文件夹
public boolean insertNewFileFold(NetFile newFold) {
    try {
        filetemplate.update("insert into
networkdisk.file(file_userId,file_parentId,file_path, filename) values
(?,?,?,?)",
                newFold.getFile_userId(),
                newFold.getFile_parentId(),
                newFold.getFile_Path(),
```

```
                newFold.getFileName());
        return true;
    } catch (Exception e) {
        System.out.println(e);
        return false;
    }
}
```

图 4-6 新建文件夹页面

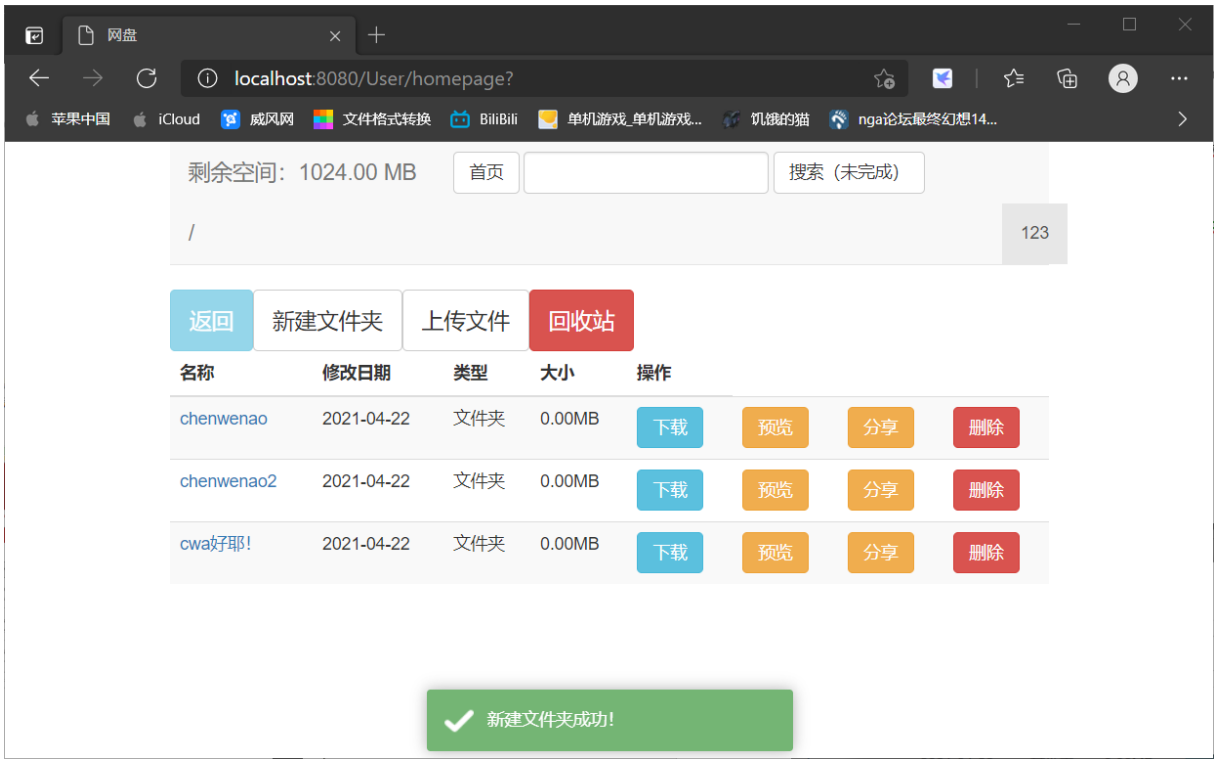

图 4-7 新建文件夹结果

4.3.3. 上传文件功能的详细设计

上传文件前端思路和新建文件夹类似，但是要更麻烦，为了用户体验，我还加入了进度条功能，这个进度条功能是利用 xhr 来建立事件，进而实现监控文件上传进度的功能，详细代码如下：

```
// 上传文件
function uploadFile() {
    const checkFile = $("#uploadFileInput").val();
    var dataRelease=0;
    if (null == checkFile || "" == checkFile) {
        toastr.warning("文件为空,请选择文件！");
        return;
    } else {
        //获取剩余空间
        $.ajax({
            type: "GET",
            url: "/User/getCurrentUser",
            success: function (data) {
                dataRelease=data.dataMax-data.dataSize;
            },
        })
        if ($("#uploadFileInput")[0].files[0].size > dataRelease) {
            toastr.warning("剩余空间不足！");
            return;
        } else {
            const formData = new
FormData(document.getElementById("uploadFileForm"));
            $.ajax({
                type: "POST",
                enctype: 'multipart/form-data',
                url: '/File/uploadFile',
                data: formData,
                cache: false,
                processData: false,
                contentType: false,
                success: function (data) {
                    if (data == "请上传一个文件！") {
                        toastr.warning(data);
                        $('#uploadFileInput').val('');
                        return;
                    } else if (data == "当前文件已存在！") {
                        toastr.warning(data);
                        $('#uploadFileInput').val('');
                        return;
```

```
                } else if (data == "上传成功！") {
                    toastr.success(data);
                    $('#uploadFileInput').val('');
                    refreshData();
                    return;
                } else {
                    $('#uploadFileInput').val('');
                    toastr.error(data);
                    return;
                }

            },
            xhr: function () {
                var xhr = $.ajaxSettings.xhr();
                if (xhr.upload) {
                    //处理进度条的事件
                    xhr.upload.addEventListener("progress", progressHandle,
false);
                    //加载完成的事件
                    xhr.addEventListener("load", completeHandle, false);
                    //加载出错的事件
                    xhr.addEventListener("error", failedHandle, false);
                    //加载取消的事件
                    xhr.addEventListener("abort", canceledHandle, false);
                    //开始显示进度条
                    showProgress();
                    return xhr;
                }
            }
        }, 'json');
    }
  }
}

//显示进度条的函数
var start = 0;
function showProgress() {
    start = new Date().getTime();
    $(".progress-body").css("display", "block");
}

//隐藏进度条的函数
function hideProgress() {
    $('.progress-body .progress-speed').html("0 M/S, 0/0M");
    $('.progress-body percentage').html("0%");
```

```
    $('.progress-body .progress-info').html("请选择文件并点击上传文件按钮");
    $(".progress-body").css("display", "none");
}

//进度条更新
function progressHandle(e) {
    $('.progress-body progress').attr({value: e.loaded, max: e.total});
    var percent = e.loaded / e.total * 100;
    var time = ((new Date().getTime() - start) / 1000).toFixed(3);
    if (time == 0) {
        time = 1;
    }
    $('.progress-body .progress-speed').html(((e.loaded / 1024) / 1024 /
time).toFixed(2) + "M/S, " + ((e.loaded / 1024) / 1024).toFixed(2) + "/" +
((e.total / 1024) / 1024).toFixed(2) + " MB. ");
    $('.progress-body percentage').html(percent.toFixed(2) + "%");
    if (percent == 100) {
        $('.progress-body .progress-info').html("上传完成,后台正在处理...");
    } else {
        $('.progress-body .progress-info').html("文件上传中...");
    }
}

//上传完成处理函数
function completeHandle(e) {
    $('.progress-body .progress-info').html("上传文件完成。");
    setTimeout(hideProgress, 2000);
}

//上传出错处理函数
function failedHandle(e) {
    $('.progress-body .progress-info').html("上传文件出错，服务不可用或文件过大。
");
}

//上传取消处理函数
function canceledHandle(e) {
    $('.progress-body .progress-info').html("上传文件取消。");
}
```

上传文件的后端部分也相对麻烦一点，核心思路是，获取到前端上传的文件后，利用这个文件以及 currentUser 和 currentFile 的相关信息，构建一个 newFile 对象，然后将这个文件存储到服务器中的项目文件夹下，再将这个文件对象插入数据库，实现文件上传，后端核心代码如下：

```
//上传文件
@PostMapping("/File/uploadFile")
```

```
public String newFile(@RequestParam("upFile") MultipartFile upFile,
HttpSession session) throws IOException {
    NetUser currentUser = (NetUser) session.getAttribute("currentUser");
    //判断是否传入图片。
    if (upFile.isEmpty()) {
        //无文件传入
        return "请上传一个文件！";
    } else {
        //有文件传入
        //新文件需要字段:
filename,filetype,filesize,filelocation,file_userId,file_parentId,file_path
        NetFile newFile = new NetFile();

        //设置 fileName。
        String originalFileName = upFile.getOriginalFilename();
        newFile.setFileName(originalFileName.substring(0,
originalFileName.lastIndexOf(".")));

newFile.setFileType(originalFileName.substring(originalFileName.lastIndexOf
(".") + 1, originalFileName.length()));
        newFile.setFileSize(upFile.getSize());
        newFile.setFile_userId(((NetUser)
session.getAttribute("currentUser")).getUserId());
        newFile.setFile_parentId(((NetFile)
session.getAttribute("currentFile")).getFileId());
        if (fileService.getUserFileByOthers(newFile) != null) {
            //有重名文件
            return "当前文件已存在！";
        } else {
            newFile.setFile_Path(((NetFile)
session.getAttribute("currentFile")).getFile_Path() + "/" +
newFile.getFileName());
            //获取课程图片存储文件夹，若不存在，就创建文件夹。
            String fileDirPath = "src/main/resources/static/data/userdata/" +
((NetUser) session.getAttribute("currentUser")).getUsername() + ((NetFile)
session.getAttribute("currentFile")).getFile_Path() + "/" +
originalFileName;
            File fileDir = new File(fileDirPath);
            if (!fileDir.exists()) {
                // 递归生成文件夹
                fileDir.mkdirs();
            }
            try {
                // 构建真实的文件路径
                File realFile = new File(fileDir.getAbsolutePath());
```

```
                // 上传图片到绝对路径
                upFile.transferTo(realFile);
                //System.out.println("上传成功！");
                //设置课程图片 Logo。
                newFile.setFileLocation(realFile.getAbsolutePath());
                fileService.addNewFile(newFile, ((NetUser)
session.getAttribute("currentUser")).getUserId());
                currentUser.setDataSize(currentUser.getDataSize() +
newFile.getFileSize());
                return "上传成功！";
            } catch (IOException e) {
                System.out.println(e);
            }
        }
    }
    return "未知错误！";
}
```

上传文件的功能效果如图：

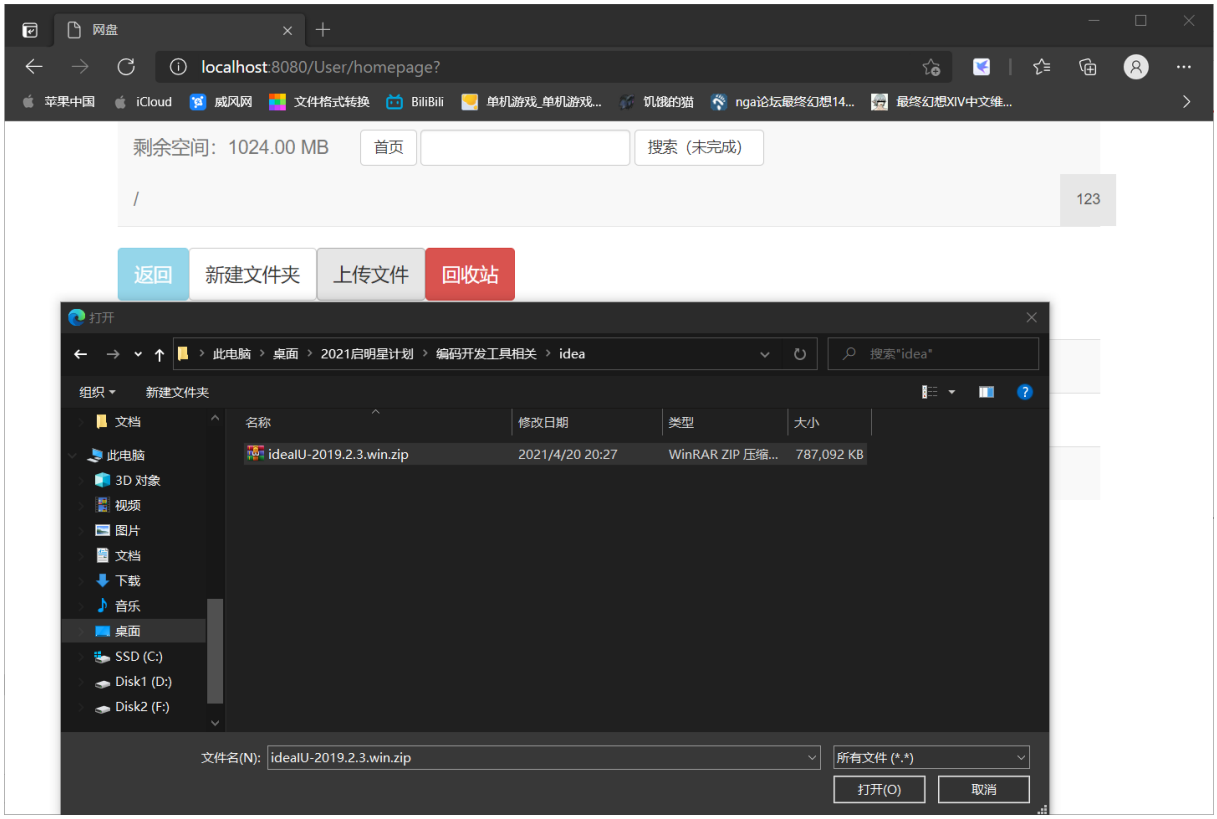

图 4-8 上传文件前

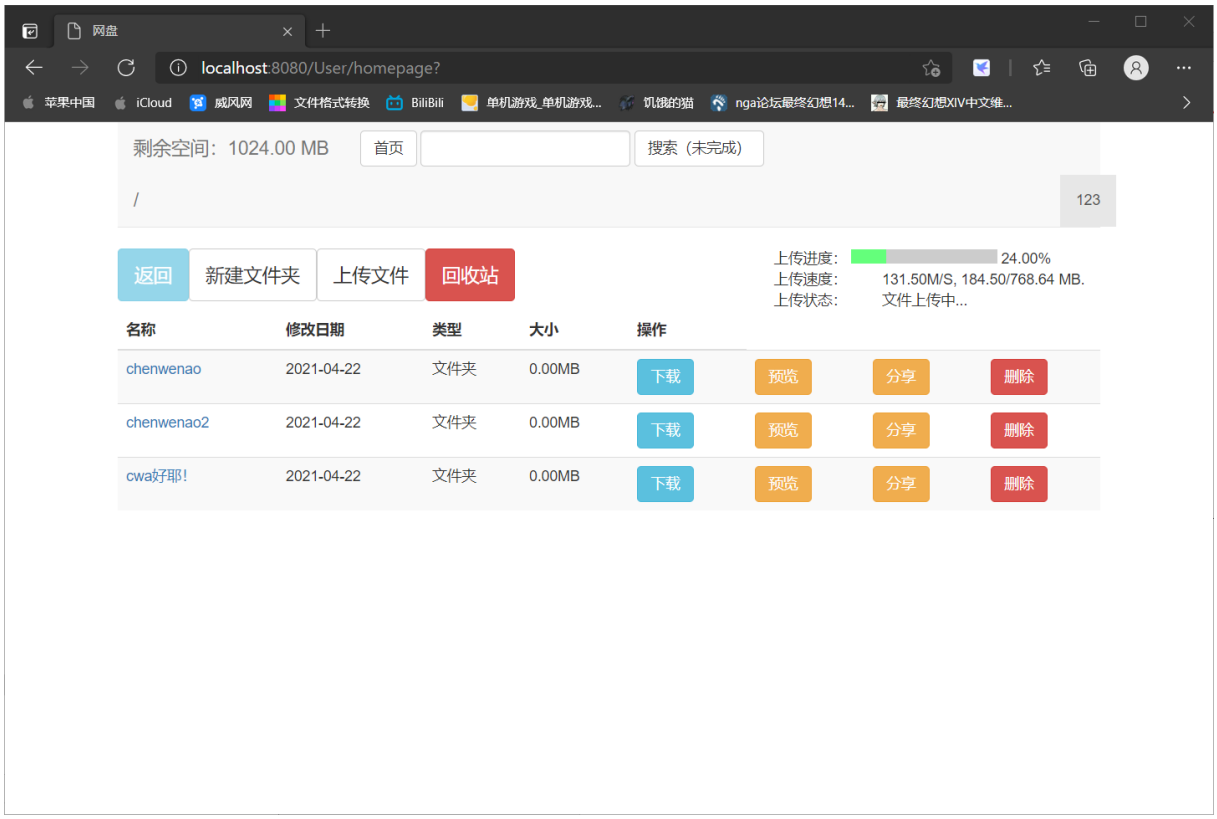

图 4-9 上传文件中

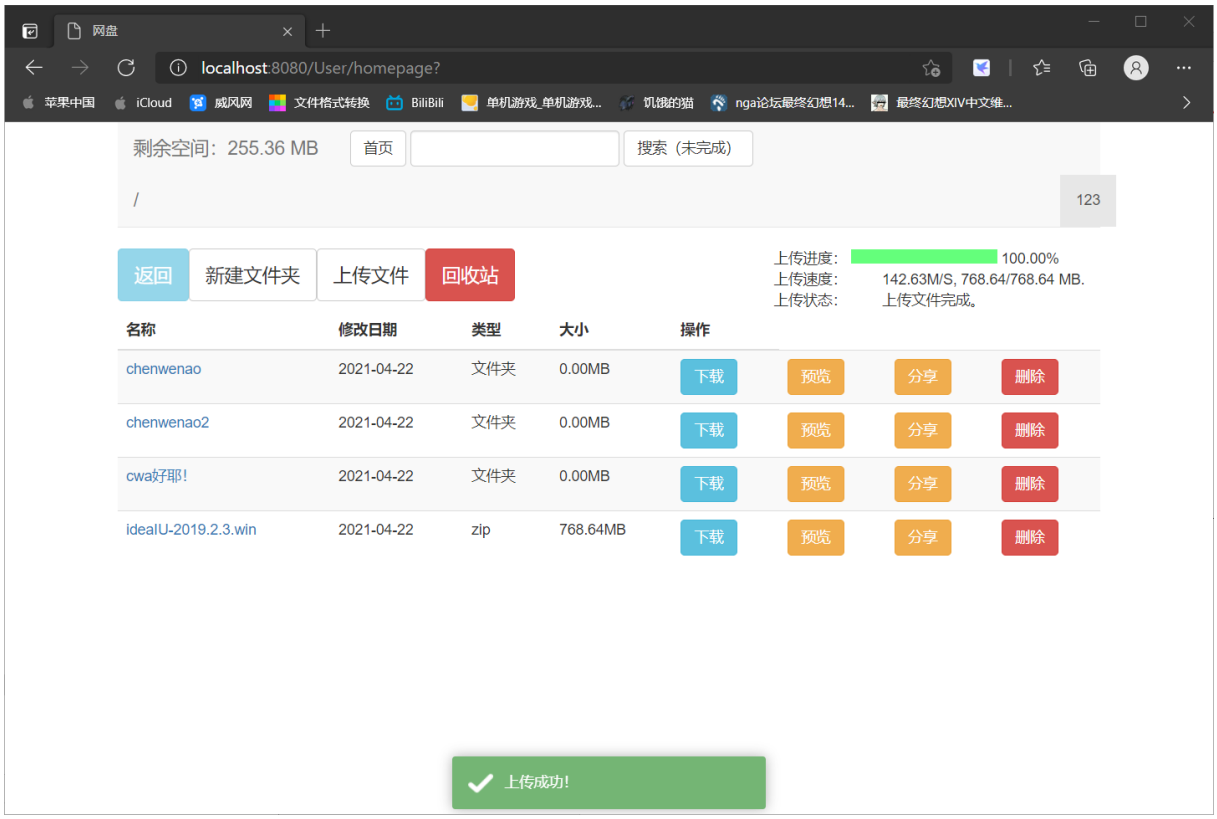

图 4-10 上传文件后

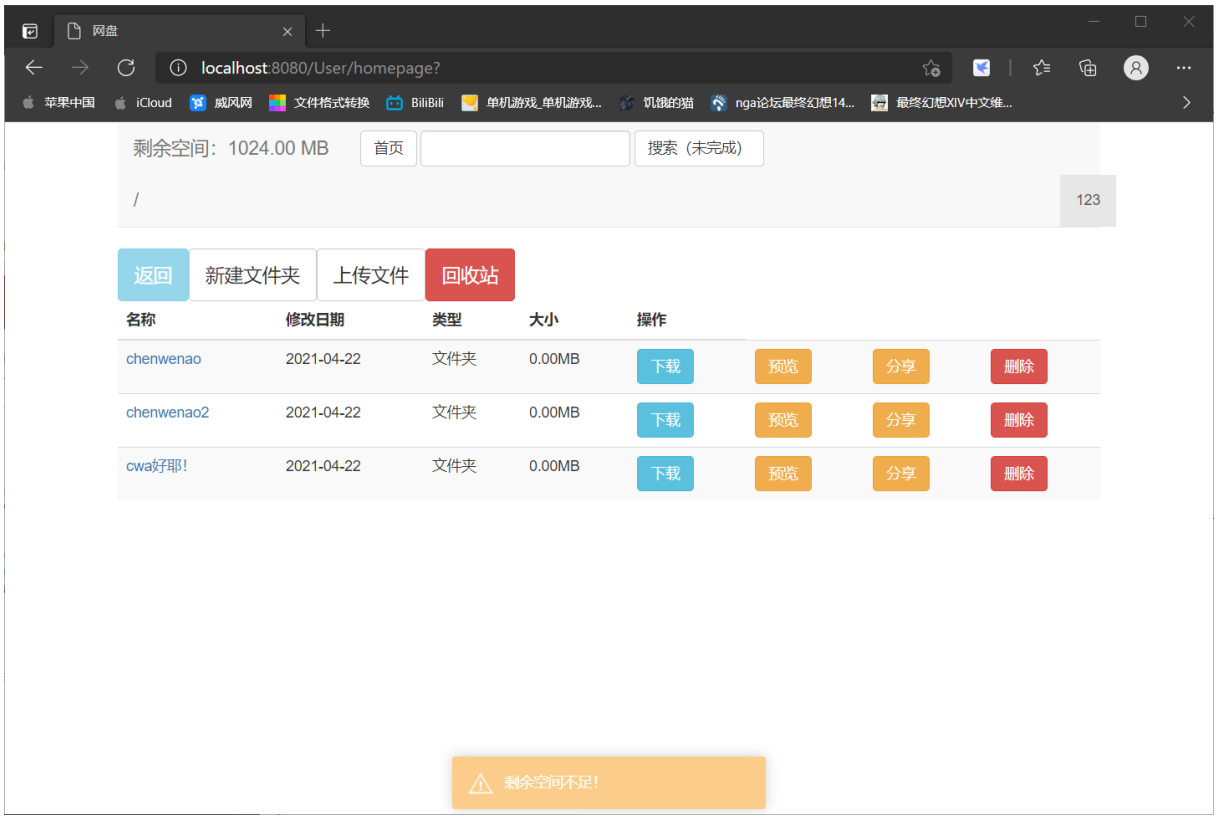

图 4-11 上传文件失败

4.3.4. 下载文件功能的详细设计

下载文件的前端很简单，其实已经包含在了 refreshData()函数里，也就是这一行代码：

```
"<td><button type='button' class='btn btn-info'
onclick=\"window.open('/File/downloadFile/" + oneFile.fileId + "')\">下载
</button></td>" +
```

这一行代码，表示的是，当我们点击表格中对应文件的下载按钮时，就会向“/File/downloadFile/{fileId}”这个 url 发送打开页面的请求，后端 controller 在接收到这个请求时，就会在数据库中，根据 fileId 找到对应文件，再根据文件的 fileLocation 属性，获取文件，然后发送给前端，前端会通过浏览器，或者调用下载器，下载自己的文件。后端代码如下：

```
//下载文件
@GetMapping("/File/downloadFile/{downloadId}")
public String downloadFile(@PathVariable("downloadId") int downId,
HttpServletResponse response, HttpSession session) {
    NetFile downFile = fileService.getUserFileById(downId);
    if (downFile.getFile_userId() != ((NetUser)
session.getAttribute("currentUser")).getUserId())
        return null;
    else {
        String fileName = downFile.getFileName() + "." +
```

```
downFile.getFileType();
        //设置文件路径
        String realPath = downFile.getFileLocation().substring(0,
downFile.getFileLocation().lastIndexOf('\\'));
        File file = new File(realPath + '/' + fileName);
        if (!file.exists()) {
            return "下载文件不存在";
        }
        response.reset();
        response.setContentType("application/octet-stream");
        response.setCharacterEncoding("utf-8");
        response.setContentLength((int) file.length());
        response.setHeader("Content-Disposition", "attachment;filename=" +
fileName);

        try (BufferedInputStream bis = new BufferedInputStream(new
FileInputStream(file));) {
            byte[] buff = new byte[1024];
            OutputStream os = response.getOutputStream();
            int i = 0;
            while ((i = bis.read(buff)) != -1) {
                os.write(buff, 0, i);
                os.flush();
            }
        } catch (IOException e) {
            return "下载失败";
        }
        return "下载成功";
    }
```

功能实现效果：

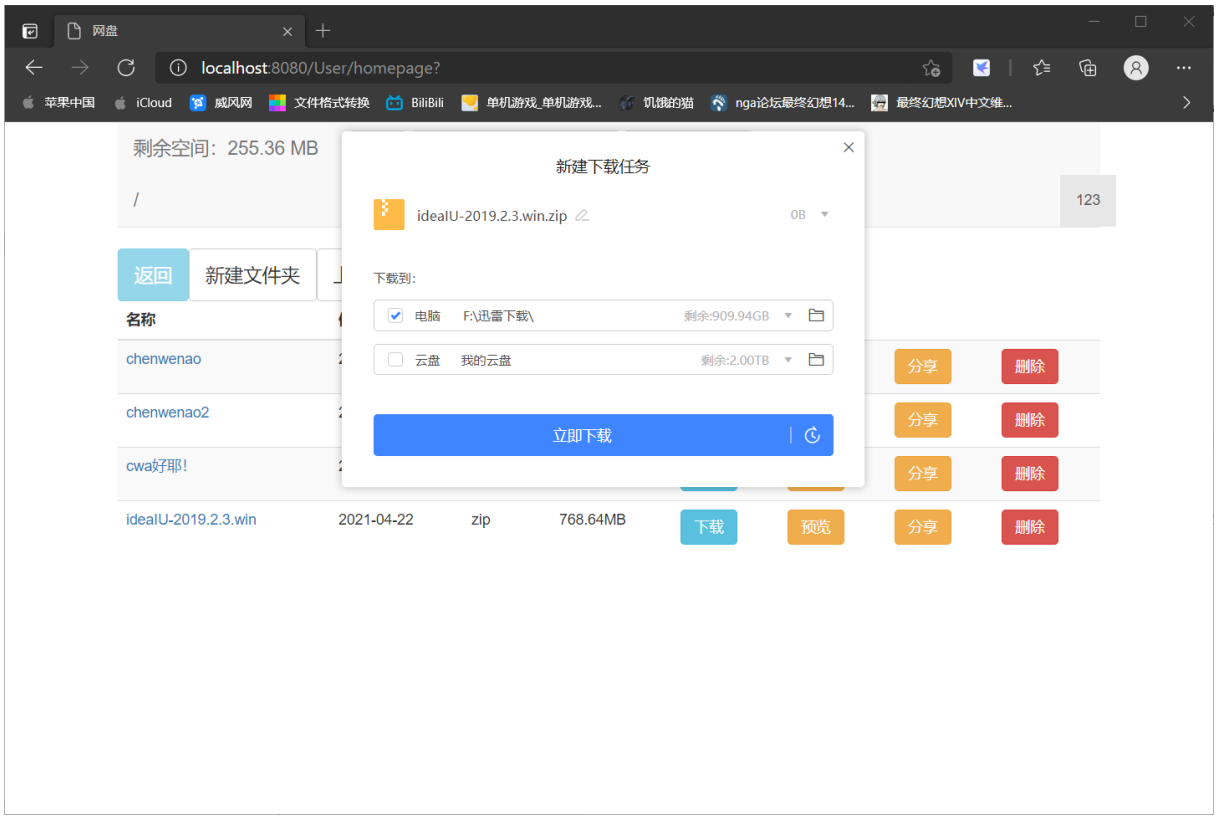

图 4-12 文件下载功能

4.3.5. 分享文件功能的详细设计

分享文件和删除确认比较相似，首先，点击文件的分享后，会用一个 hidden 属性的 input 记下分享文件的 id，然后调用`$('#shareFileBox').modal('hide');`来让 shareFileBox 这个模态框显示出来，在这个模态框中，有一个 RadioButton，选择分享时长，当用户选择完毕后，点击确认，会调用 *shareFileSubmit()* 这个 function，在这个 function 中，获取到 hidden 的 input 中的分享文件的 id 和分享时长，发送 ajax 到后端。模态框代码如下：

```
<div class="modal fade" id="shareFileBox" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <label>请选择分享时长：</label>

            <div>
                <label class="radio-inline">
                    <input type="radio" name="shareFileDay" id="shareDay3" value="option1" checked> 3 天
                </label>
                <label class="radio-inline">
                    <input type="radio" name="shareFileDay" id="shareDay7"
```

```
value="option2"> 7 天
                </label>
                <label class="radio-inline">
                    <input type="radio" name="shareFileDay" id="shareDay30"
value="option1" checked> 30 天
                </label>

                <div class="modal-footer">
                    <input id="shareId" type="hidden"></input>
                    <button type="button" class="btn btn-default" data-
dismiss="modal">取消</button>
                    <a onclick="shareFileSubmit()" class="btn btn-success"
data-dismiss="modal">确定</a>
                </div>
            </div>
        </div>
    </div>
</div>
```

分享信息提交方法为：

```
function shareFileSubmit() {
    var fileId = $('#shareId').val();
    var shareDay = $("input[name='shareFileDay']:checked").val();
    $('#shareFileBox').modal('hide');
    $.ajax({
        url: '/File/shareFile/' + fileId + '/' + shareDay,
        type: 'GET',
        success: function (data) {
            alert(data);
            $('#shareId').val("");
        }
    })
}
```

后端根据输入的数据，新建一个 NetShare 对象，插入到数据库中，并且根据时间戳和 MD5 加密，自动生成分享密码。功能实现如图：

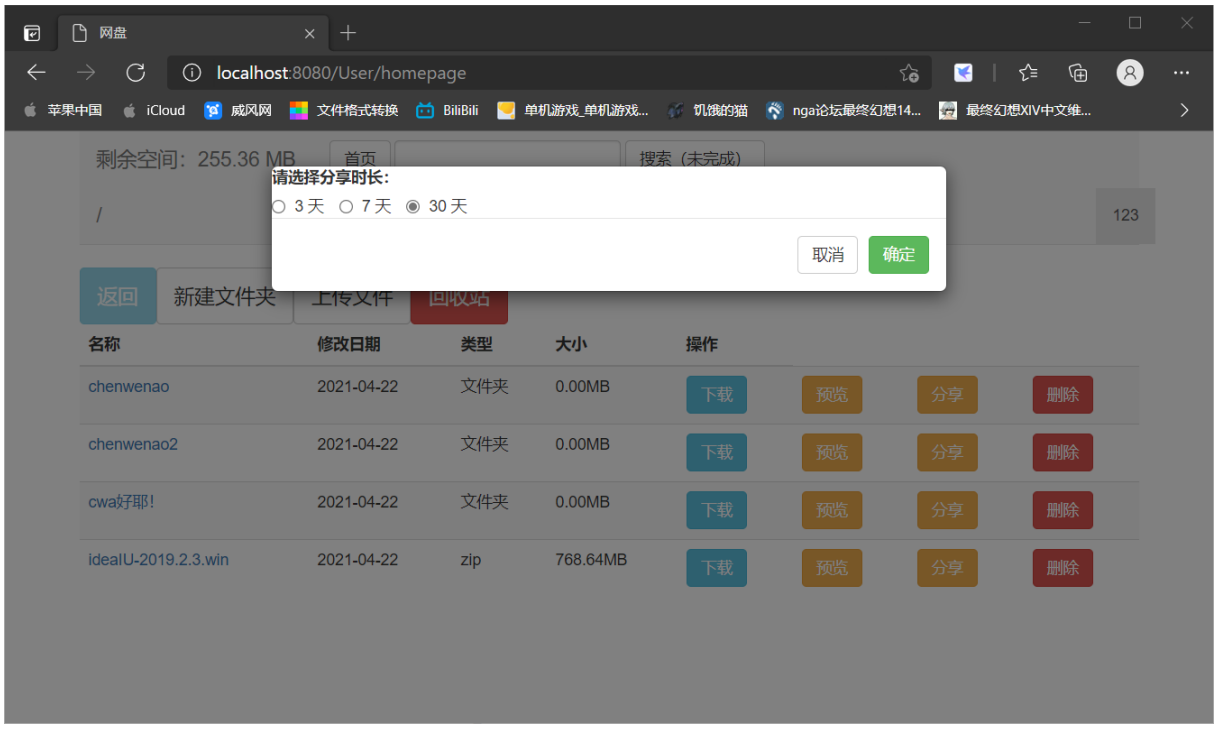

图 4-13 分享时长选择

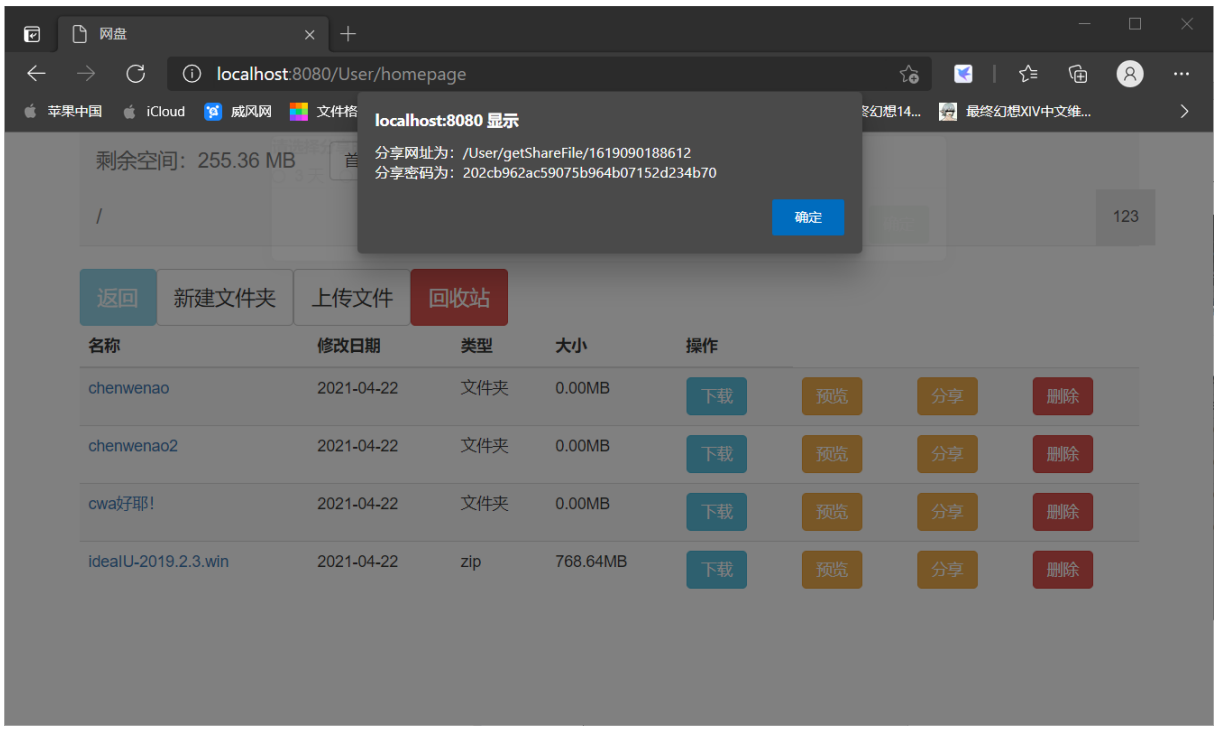

图 4-14 分享完成

5. 总结与回顾

经过多次与百度网盘对比，学习，虽然与百度网盘这一成熟的网盘产品相比，本系统还有较大差距，但是已经实现了网盘的主要功能。系统采用的是当下最流行的 SpringBoot 的框架和 MVC 开发模式，整体开发比较顺利，同时，运用了 BootStrap 前端插件，这一插件是将前端页面划分为 12 列，得益于这一网格系统，本系统已经具备一定的屏幕适应能力，可以根据屏幕大小自动切换组件间距等。该网络文件管理系统已经具备一定的实用价值。

但是，本次开发也遇到过不少难题。首先，由于个人的前端能力所限，页面设计不够精美，为了弥补，我使用了 toastr 插件，BootStrap 的样式，但是最终结构还是差强人意，本次开发暴露出我对于前端框架的掌握还不够，需要继续深化这一方面的学习，磨炼自己的技能。另一方面，系统的小功能还不多，目前只支持 word 的预览操作，还不支持视频在线观看以及压缩包在线解压等功能，这些功能虽然不是一款网盘产品必备的功能，却是极大提升用户体验的突破口，随着自身能力的提升，这些功能也会尽力实现。

参考文献

[1] 沈泽刚，秦玉平.Java 语言程序设计[M].第二版.北京:清华大学出版社,2013.
[2] 王珊，萨师煊.数据库系统概论[M].第五版.北京:高等教育出版社,2014.
[3] 张继平，龚靖.云存储解析[M].第六版.北京:任命邮电出版社,2013.
[4] 耿祥义，张跃平.Java2 实用教程[M].第五版.北京:清华大学出版社，2017.
[5] 周可，王桦，李春花. 云存储技术及其应用[J]. 中兴通讯技术，2010.
[6] 张峰. 应用 SpringBoot 改变 web 应用开发模式[J]. 科技创新与应用，2017.
[7] 谢金星. 基于云存储的网盘系统设计与实现[D]. 2016.
[8] 雷凯. 基于 Web 的网络视频播放器的设计与实现[D]. 2013.
[9] 张奎,李哲,苑庆涛. 基于网盘的项目文档在线管理系统[J].西安邮电学院学报，2012.
[10] 何海东，张文秋. 基于 Web 的网络硬盘的设计与实现[J]. 四川理工学院学报(自然科学版)，2010(02):175-177.
[11] Hanna P . JSP 技术大全[M]. 机械工业出版社，2002.
[12] 李林朋. 浅析网盘系统结构中云存储技术[J]. 信息通信，2013.
[13] Ponomarenko G , Klyucharev P . JavaScript Programs Obfuscation Detection Method that Uses Artificial Neural Network with Attention Mechanism[C]// Secure Information Technologies 2019 (BIT 2019). 2020.
[14] Hong-Jun Y , Jin-Ying W . The Design and Implementation of Blog System Based on ASP.Net and AJAX[J]. Computer Knowledge and Technology, 2011.

致 谢

转眼间，大学四年已经到了尾声，回顾这四年的时光，自己收获颇丰，但望向以后的人生，仍然充满迷茫。从大一的 C 语言，到大四的小型网站，小型软件的开发，我一步一步走过来，遇到了很多可爱的同学和老师，也遇到过各种各样的难题。

我很庆幸自己在大三能够遇到一个学习伙伴，是他带领我进入网站开发的领域，教会了我网站开发的基础，也正是在大三，我才真正确定自己要进入网站开发这一方向。同时，我也遇到了众多学识渊博的大学教师，他们思维灵活，技术深厚，不仅仅教会了我很多计算机的基础知识，也带领我开发了各式各样的小软件。本次毕业设计，正是我利用自己大学期间学习的知识，来为我的大学的时光写上的一个完美句号。

从我十岁开始接触第一台计算机，到喜欢上计算机里各式各样的软件，再到如今自己已经能够开发一些简单的软件，我终于能够迈向自己喜欢的行业，在自己喜欢的领域，遇到更多喜欢的人，

一起干更多喜欢的事。这一路走来，免不了众多同学老师的帮助，在此祝所有软件工程的同学们，前途无量，未来可期；也祝湖北大学的导师，教授们，桃李遍天下，万事皆胜意！