文件系统开发文档

# 一、需求

## （一）身份认证

1.系统有唯一的一个管理员admin

2.管理员权限：

·进行所有文件操作：浏览、下载、上传、复制、移动、删除、压缩、新建目录

·更改管理员密码

·添加、删除用户账号

·更改用户的密码

·控制游客、用户、管理员的可见目录（添加、删除）

3.用户权限：

·浏览、下载、上传

·更改自己的密码

4.游客权限：

·浏览、下载（文件）

## （二）文件系统

1.上传时，限制大小为4GB（设置整个系统的HTTP传输最大数据量为4GB）

2.对于游客、用户、管理员，有不同的可见目录，管理员可控制可见目录（添加、删除）

3.浏览文件时：

·文件分页显示，每页30行，文件按照类型排序：目录 -- 视频 -- 音频 -- 图片 -- 文本 -- 压缩包 -- 其他

·每一行以图标 -- 文件名 -- 文件大小 -- 操作按钮的顺序排列

·支持显示的文件类型图标：目录、文本、图片、音频、视频、压缩包、其他

·文件名过长时，后面以省略号结束；操作按钮中添加“查看完整文件名”按钮

·文件大小至多保留2位小数，文本所占布局宽度要固定（4位数字.2位数字MB）

4.文件可以在线预览，与浏览器兼容性有关：文本、图片、音频、视频

5.操作按钮中要有文件下载

6.下载目录时，服务端先把目录压缩成压缩包，然后再传输，传输完毕后，服务端删除压缩包。**倘若此功能无法实现，删除用户、管理员的下载目录功能**。

7.删除文件（或目录）时，弹出确认框

8.多选操作：

·下载/复制/移动/删除：建立操作队列，优先操作文件，再操作目录

·压缩：建立压缩文件队列，先压缩文件，再压缩目录

9.搜索文件：

·可以搜索文件和目录，在当前目录中进行递归搜索，在进行搜索时有转圈，每搜索出一个匹配的结果，都实时显示在搜索窗口中

## （三）前端界面

1.系统有两套主题：浅色、深色，默认为深色

2.尽可能地显示动画、过渡等效果

3.响应式布局，移动端优先，rem适配文字等大小

4.手机端测试浏览器：华为内置浏览器、UC、火狐

5.电脑端测试浏览器：Edge、Chrome

# 二、技术栈

## （一）前端

1.使用Bootstrap5框架进行开发：

·原生HTML + JinJa2模板引擎

·CSS扩展语言SASS

·原生JavaScript

2.HTML：

·用好JinJa2的变量定义、继承、部件、宏，改善HTML重复代码

·前后端交互中，多考虑使用JinJa2的过滤器

·能不用JavaScript渲染HTML就不用，尽可能地用JinJa2渲染

3.CSS：

·尽可能地使用Bootstrap的基类、工具类来改变CSS，能不用CSS就不用CSS

4.JavaScript：

·JavaScript最好只用在少数的AJAX

## （二）后端

1.使用Flask Web框架进行开发：

·使用Flask扩展的ROM技术来管理数据库

·分成两个蓝图：auth（身份认证）、main（文件系统）

·在static下区分蓝图的静态文件

·程序的启动要在命令提示符中进行

2.代码结构：

·config.py用于保存项目基本信息，以及应用启动模式

·项目根目录下的\_\_init\_\_.py编写工厂函数启动app

·utils.py保存通用工具函数、工具类

·尽可能地使用类作为代码的基本元素

3.使用SQLite数据库

4.单元测试：

·每实现一个功能，就要编写完整的单元测试

·单元测试包括：数据库模型测试、视图函数测试、功能函数或类测试

·每编写一个前端元素，就要进行电脑端、手机端浏览器的测试

5.其他：

·requirements.txt

·Git版本控制

·sql测试文件

·应用启动的bat文件：包含开发环境和生产环境

·开发文档

## （三）开发模式

1.构思，参考开发文档的需求、技术栈部分

2.将开发的具体步骤添加到开发文档

3.先编写简单视图函数，作为前后端交互的桥梁

4.编写前端页面，注意参考开发文档的技术栈

5.由前端页面驱动：

·设计数据库

·编写功能函数或功能类

6.将后端功能放入视图函数，实现前后端连接

7.用电脑、手机浏览器测试前端页面功能

8.编写单元测试

9.更新requirements.txt文件

10.上述步骤没有问题后，用Git把代码推送到GitHub

# 三、项目基本架构

## （一）目录结构

- app

- auth

- \_\_init\_\_.py

- views.py

- main

- \_\_init\_\_.py

- views.py

- static

- auth

- sass

- css

- js

- main

- sass

- css

- js

- base

- sass

- css

- js

- templates

- base

- auth

- main

- database\_sqlite

- untils.py

- model.py

- \_\_init\_\_.py

- admin\_dir

- run\_product.bat

- run\_development.bat

- tests

- auth

- \_\_init\_\_.py

- main

- \_\_init\_\_.py

- model

- \_\_init\_\_.py

- untils

- \_\_init\_\_.py

- base

- \_\_init\_\_.py

- \_\_init\_\_.py

- config.py

- app.py

- .gitignore

- requirements.txt

## （二）构建项目基本结构

1.环境支持：

·Python环境

·Git环境

·PyCharm环境

·Bootstrap5环境

·创建目录结构

2.编写.gitignore

3.编写config.py

·定义项目全局变量

·定义三种应用启动模式：开发（默认）、生产、测试

4.编写项目根目录下的\_\_init\_\_.py

·根据应用启动模式初始化应用

·添加Flask扩展

·注册蓝图

5.编写app.py

·创建应用

·注册命令提示符命令flask shell

6.编写admin\_dir下的启动应用bat文件

7.编写单元测试：

·编写base测试：应用是否创建成功，是否能以三种模式启动应用

·注册命令提示符命令flask test

8.简单测试一下是否能实现视图函数的前后端连接

9.更新requirements.txt文件

10.用Git把代码推送到GitHub：

·创建GitHub仓库

·将项目代码推到创建的GitHub仓库

## （三）前端界面设计

1.参考：

·百度网盘

·GitHub

·手机端ES文件管理器

·向日葵文件传输

·windows11资源管理器

2.寻找免费的网页模板

3.logo设计

# 四、身份认证

## （一）数据库模型Users

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 约束 | 键 | 备注 |
| user\_id | int |  | 主键，自增 | 用户ID |
| user\_name | varchar(255) | 非空，唯一 |  | 用户名 |
| user\_password\_hash | varchar(128) | 非空 |  | 用户密码哈希值 |

**建表语句**

CREATE TABLE IF NOT EXISTS `users`

(

`user\_id` INTEGER PRIMARY KEY AUTOINCREMENT,

`user\_name` VARCHAR(255) NOT NULL UNIQUE,

`user\_password\_hash` VARCHAR(128) NOT NULL

);

**插入管理员记录语句**

INSERT INTO `users`(`user\_name`, `user\_password\_hash`)

VALUES('admin', '123456');

其中管理员的密码会经过处理

**建立ORM映射**

·在model.py中，编写实体类Users，类变量映射数据库字段。

·设置密码的set方法，对密码进行赋值时，保存加密的密码（加密算法为pbkdf2:sha256）

·编写验证密码相等的方法

·编写检查一个用户是否是管理员的方法

**单元测试：tests/model/users**

·数据库文件会自动创建

·数据库中的users表会自动创建

·能自动插入管理员记录

·访问Users对象的密码属性会抛出异常

·设置Users对象的user\_password属性，会设置user\_password\_hash属性

·验证密码相等的方法

·是否是管理员的方法

## （二）登录登出修改密码功能

**配置Flask-Login扩展**

·为Users类添加四个Flask-Login扩展所必须的4个方法

·指定登录的视图函数

·绑定Flask-Login加载用户的函数

**编写匿名用户类AnonymousUser**

·包含方法is\_admin

**登录登出修改密码功能**

1.编写base.html页面模板，包括导航栏和页面主体

2.编写index.html基本框架和登录框组件

3.将index.html与视图函数index绑定

4.登录成功和失败的反馈：记录登录成功状态，页面随之变化；反馈登录失败信息

5.编写修改密码框

6.编写update\_password视图函数，处理请求：成功后登出；失败时提示

7.编写工具函数check\_legal用于检查用户名或用户密码是否合法

**单元测试**

·测试工具函数check\_legal

·登录失败：用户已登录

·登录失败：用户名不存在

·登录成功：重定向到首页

·登出

·修改密码失败：未登录

·修改密码失败：密码不合法

·修改密码失败：密码与原密码相同

·修改密码成功：用户登出

## （三）编写错误页面

**前端页面**

·404页面

·500页面

·405页面：响应404页面

·403页面：响应404页面

**视图函数**

·用main蓝图绑定全局路由

**单元测试**

·405响应404

·403响应404

## （四）用户管理

**用户管理页面**

·编写用户管理页面，包括：多选框，用户列表，添加用户按钮，列表中的每个用户右方有两个按钮：删除、修改密码。

·编写视图函数user\_manage

·所有的用户管理操作都需要管理员权限，编写装饰器admin\_required

·编写工具函数check\_and\_update\_password用于修改密码

**单元测试**

·管理员身份才能访问用户管理页面，非管理员访问响应403（实际响应404）

·测试工具函数check\_and\_update\_password

·添加用户失败：需要管理员权限

·添加用户失败：用户名不合法（只能包含字母数字特殊符号，长度小于16）

·添加用户失败：相同用户名的用户已存在

·添加用户失败：密码不合法（只能包含字母数字特殊符号，长度小于16）

·添加用户成功：检查要添加的用户是否存在

·删除用户失败：需要管理员权限

·删除用户失败：用户不存在（只要有1个不存在则都不删除）

·删除用户成功：检查要删除的用户还存不在

·修改密码失败：需要管理员权限

·修改密码失败：用户不存在

·修改密码失败：密码不合法

·修改密码失败：与原密码相同

·修改密码成功：检查新密码

# 五、可见目录管理

## （一）面向对象的路径操作

1.编写is\_legal\_path函数，检查一个路径是否合法：是否有'/'或'\\'；绝对路径是否存在

2.编写类Path，包含\_\_init\_\_和\_\_repr\_\_，传入一个path可进行实例化，实例化前会用is\_legal\_path检查传入的path是否合法

3.编写类MountPath继承Path，表示磁盘根

4.编写类DirPath继承Path，表示目录

5.编写类FilePath继承Path，表示文件

6.DirPath和FilePath对象有基础信息：dirname/filename、type、size、father\_path、mount\_path等

7.DirPath和FilePath对象还包含father和mount属性，分别是DirPath对象和MountPath对象，表示上一级目录、所在磁盘。这两个属性是两个对象，只有在第一次访问时才会被实例化（**防止链式生成**）

8.DirPath和MountPath包含属性children，是一个元素是DirPath对象或FilePath对象的列表，表示一个目录的所有子目录、子文件。这个属性初始为None，只有在第一次访问时里面的元素才会被实例化（**防止链式生成**）

## （二）可见目录模型VisibleDir

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 约束 | 键 | 备注 |
| dir\_id | int |  | 主键，自增 | 目录ID |
| dir\_path | text | 非空，唯一 |  | 目录路径 |
| dir\_permission | int | 非空 |  | 访问目录所需权限 |

对于dir\_permission，规定：

·dir\_permission=1，游客、普通用户、管理员可访问

·dir\_permission=2，普通用户、管理员可访问

·dir\_permission=4，管理员可访问

**建表语句**

CREATE TABLE IF NOT EXISTS `visible\_dir`

(

`dir\_id` INTEGER PRIMARY KEY AUTOINCREMENT,

`dir\_path` TEXT NOT NULL UNIQUE,

`dir\_permission` INT NOT NULL

);

**建立ORM映射**

·编写类VisibleDir，包含属性dir\_id、dir\_path、dir\_permission

·每一个VisibleDir对象会包含一个DirPath对象，用于更方便地进行路径相关的操作

·在config.py中编写类Permission，用于保存游客、普通用户、管理员的权限值，并添加用@property修饰的方法permission到Users、AnonymousUser中，让所有的用户对象都具有属性permission

·在VisibleDir类中编写方法can，用于检查一个用户是否有权限访问一个可见目录（用异或运算）

·在VisibleDir类的\_\_init\_\_方法中，检查dir\_permission的枚举值是否合理，若不合理，抛出异常

**单元测试**

·应用启动后是否会自动创建visible\_dir表

·创建VisibleDir对象时，可见目录权限值不合理的情况下不会产生数据库条目

·检查游客、普通用户、管理员是否能访问3种不同权限的可见目录

## （三）可见目录页面

**页面元素**

1.可见目录列表（管理员、普通用户、游客各一个列表）

2.管理员可见的只有管理员可见，普通用户可见的是指管理员、普通用户可见，游客可见的是指管理员、普通用户、游客都可见

3.全选、全不选、反选按钮

4.添加、删除可见目录按钮

**添加可见目录**

·目前只支持输入绝对路径的方式添加可见目录

·下拉框选择谁可见

**单元测试**

·可见目录页面只有管理员才能访问

·添加可见目录失败：无权限

·添加可见目录失败：路径不合法

·添加可见目录失败：权限值不合法

·添加可见目录失败：已存在

·添加可见目录成功：检查数据库

·删除可见目录失败：无权限

·删除可见目录失败：路径不存在于数据库中

·删除可见目录失败：检查数据库

# 六、文件系统

## （一）浏览文件

**前端**

·每一个文件行包括：勾选框、文件名、操作下拉菜单

·根据当前用户的权限来判断什么能显示，什么不能显示

·访问上一级按钮

·不同类型文件有不同的图标，图标颜色也不同，文件类型和图标CSS类的映射表用JinJa2定义在html中，文件类型和图标颜色的映射表放在SASS变量中

·在文件列表上方显示面包屑导航

·在导航栏增加输入框，输入绝对路径跳转到指定目录

**后端**

·访问主页

·访问可见目录根路径

·访问可见目录下的目录

·访问可见目录下的文件

·编写match\_visible\_dir函数，用于将一个目录的绝对路径匹配数据库中的可见目录路径

·访问文件系统时，携带查询参数path，表示要访问的绝对路径

·文件列表中的文件按照类型进行排序：dir > package > video > image > audio > text > unknown

**单元测试**

·不合法的path参数

·没有匹配path的可见目录

·有匹配path的可见目录，但无权限

·匹配成功

## （二）单文件操作

**前端**

·最每一行文件的最右端添加操作下拉菜单，菜单显示的按钮根据权限不同而不同：

- 下载文件、查看完整文件名（所有类型用户）

- 重命名、复制、移动、删除文件（管理员）

·添加复制当前目录的绝对路径的按钮

·上传文件按钮，按下后弹出模态框，可以点击上传文件

·上传文件使用ajax与后端进行通信

·新建目录按钮，按下后弹出模态框，确定新建目录的名称

**后端**

·编写下载、删除、重命名、移动、复制文件的视图函数

·编写上传文件的视图函数，以json形式响应

·编写新建目录的视图函数

**单元测试**

·访问下载链接：权限不够响应404

·访问下载链接成功

·删除文件：权限不够响应404

·删除文件成功：检查文件是否还存在

·重命名文件：权限不够、新旧名称相同、名称包含不合法字符、同名文件已存在，均响应404

·重命名文件成功：检查文件是否存在

·移动文件：权限不够

·移动文件成功

·复制文件：权限不够

·复制文件成功

·上传文件：权限不够（**TODO**）

·上传文件：目录不合法（**TODO**）

·上传文件：未选择文件（**TODO**）

·上传文件：文件名不合法（**TODO**）

·上传文件：已存在同名文件（**TODO**）

·上传文件：文件过大（**TODO**）

·上传文件成功（**TODO**）

·新建目录：无权限

·新建目录：名称不合法

·新建目录：同名目录已存在

·新建目录成功

## （三）单目录操作

1.删除、重命名、移动、复制目录

对于删除、重命名、移动、复制目录，只需对单文件操作的视图函数稍作修改即可

单元测试：

·删除目录失败：尝试删除可见目录

·删除目录成功

·重命名目录失败：尝试重命名可见目录

·重命名目录成功

·移动目录失败：尝试移动可见目录

·移动目录成功

·复制目录成功

2.下载整个目录（方案一）

**前端页面**

1.点击目录的“下载”按钮

2.弹出模态框，提示正在加载目录结构，同时AJAX请求目录结构，若请求失败，在模态框中提示失败，若成功，在模态框中构建目录结构

3.构建目录结构时，使用ul和li来构建层级，并为每一个层级的目录添加可折叠/展开的点击事件

4.为每一个文件（非目录）添加图标、大小

5.模态框底部有3个按钮：全部折叠、全部展开、开始下载。只有当目录构建完毕后，才能点击“开始下载”按钮。

6.点击“开始下载”按钮后，AJAX逐个下载目录结构中的文件

7.对于下载成功的文件，在文件最后打钩，对于下载失败的文件，在文件的最后打叉

8.下载完一个文件后，js把文件归档到zip中

9.下载所有文件后，js导出zip

10.额外添加一个功能：查看目录大小

·点击“查看目录大小”按钮，弹出模态框，提示等待，同时AJAX请求目录大小

·若请求成功，则显示目录大小，否则提示失败

**后端页面**

1.负责下载的视图函数中，若检查到需要下载的文件类型是目录，则响应目录结构（使用队列进行遍历，将遍历结果返回，注意将文件按照类型进行排序）

2.编写查看目录大小的视图函数，只有普通用户或管理员才有权限访问

·检查路径是否合法、是否位于可见目录下

·检查路径是否是目录路径，是否是根路径

·验证通过，遍历目录下所有文件，获取大小并计算总和，然后响应

**单元测试**

·下载目录：返回目录结构（检查是否是json形式）

·查看目录大小：无权限

·查看目录大小：不是目录

·查看目录大小：不在可见目录下

·查看目录大小：是根路径

·查看目录大小：成功

3.下载整个目录（方案二）

点击“下载”按钮后，前端提示文件正在压缩，后端开始压缩文件，压缩完毕后，使用**Flask流内容技术**，分块响应压缩文件，需要手动设置请求头以指定文件名、文件类型，文件流传输完毕后，马上在生成器中删除生成的压缩包。

注意：在生成压缩包时，有可能存在同名压缩包冲突，可以在压缩包文件后面加入时间戳来解决

**单元测试**

下载目录：尝试下载根目录

下载目录：成功

## （四）多选操作

**前端**

·添加多选按钮，点击后处于多选状态，再次点击退出多选状态

·在多选状态下，显示全选、全不选、反选按钮

·在多选状态下，显示多选操作下拉菜单

·添加删除、移动、复制选项，点击后弹出模态框，显示即将进行操作的文件列表

·添加查看大小选项，点击后弹出模态框，会显示所有已选择的文件或目录的大小，按照大小的降序排序，并显示总计大小

·添加压缩选项，点击后弹出模态框，可选择压缩算法，填入压缩文件名称，提交后，用转圈动画表示正在压缩，压缩完毕后刷新页面

·添加下载选项，点击后，将弹出模态框，列出即将下载的文件，并提示会进行压缩后再下载，点击确定后，显示正在压缩的加载动画，压缩完毕后隐藏动画

**后端**

·对于删除、移动、复制操作，只需把原视图函数的操作改成循环操作即可

·将原查看目录大小的视图函数，改成可以计算多个文件大小，更改响应的json结构

·对于压缩操作，写一个视图函数，接收多个path参数，检查参数合法性，然后再添加文件到压缩包中

·对于下载操作，检查参数合法性，然后压缩、下载。注意压缩包同名问题，使用时间戳解决，此外还需要注意下载完毕后自动删除压缩包

**单元测试**

·删除多个文件：只要有一个不合法，所有的文件都不会被删除

·删除多个文件成功

·移动多个文件：只要有一个不合法，所有的文件都不会被移动

·移动多个文件成功

·复制多个文件：只要有一个不合法，所有的文件都不会被移动

·复制多个文件成功

·查看多个文件大小：未选择任何路径

·查看多个文件大小：路径不存在

·查看多个文件大小：路径是根路径

·查看多个文件大小：只要有一个文件路径不合法，就都不进行操作

·查看多个文件大小：成功

·压缩文件：未输入文件名

·压缩文件：只要有一个路径不合法（不能有根目录），就不进行操作

·压缩文件：文件名不合法

·压缩文件：同名文件已存在

·压缩文件：压缩算法类型不合法

·压缩文件：成功

待完成：多选：分页；搜索文件；查看压缩包；