# 目录

目录

[目录 2](#_Toc156059207)

[摘要 2](#_Toc156059208)

[正文 2](#_Toc156059209)

[相关知识简介 2](#_Toc156059210)

[功能设计 3](#_Toc156059211)

[设计初衷以及思想 3](#_Toc156059212)

[OSS云存储 4](#_Toc156059213)

[功能结构 6](#_Toc156059214)

[实验结果及分析 57](#_Toc156059215)

[结束语（心得体会） 64](#_Toc156059216)

[课设总结 64](#_Toc156059217)

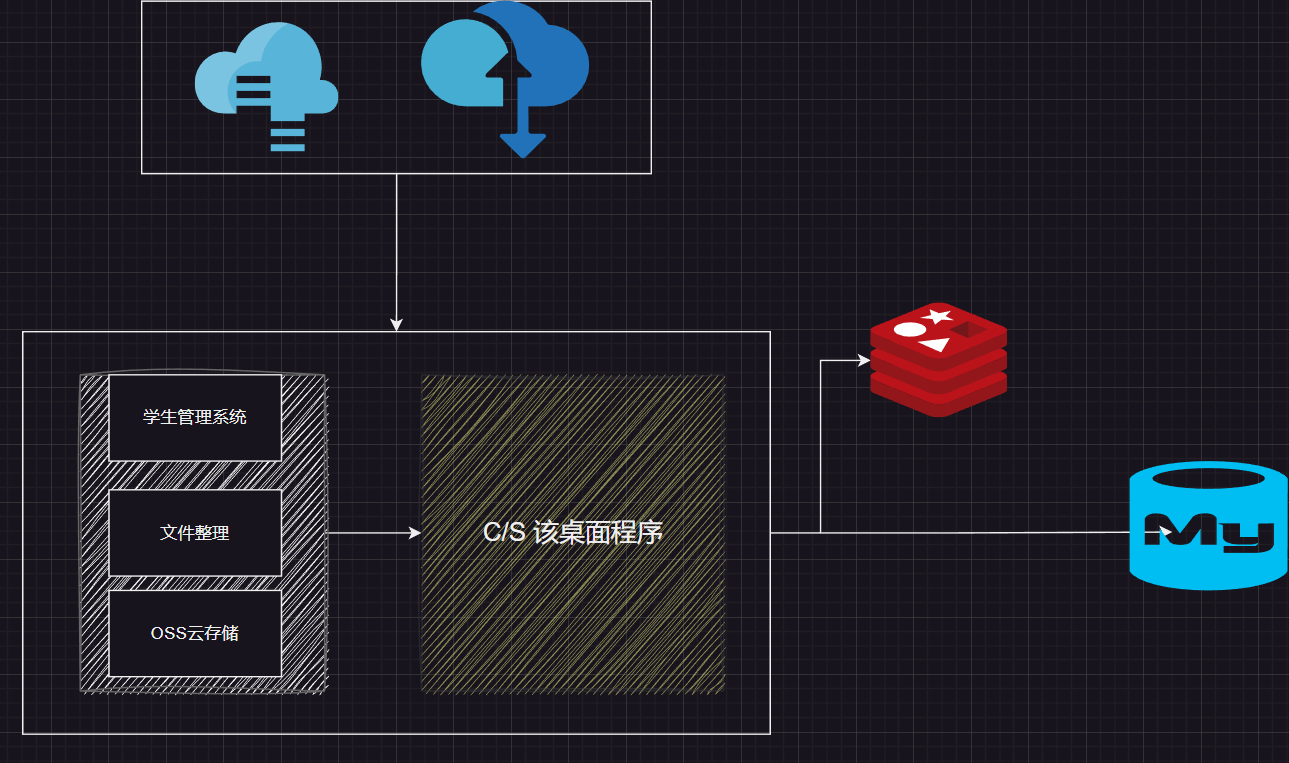
[参考文献（资料） 65](#_Toc156059218)

# 摘要

* 本系统基**于RABC权限模块实现学生管理系统**<该模块实现的功能：登入登出、权限操作、学生信息CRUD、模糊查询、分页查询、Excel导入导出>。
* 由于python IO效率并不高，使用Rust实现了一个**文件整理功能。**
* 基于阿里云**OSS实现云存储,**实现文件的上传与下载，基于Tkinter GUI实现文本编辑器实现简单文本上传于下载以及云查看。

# 正文

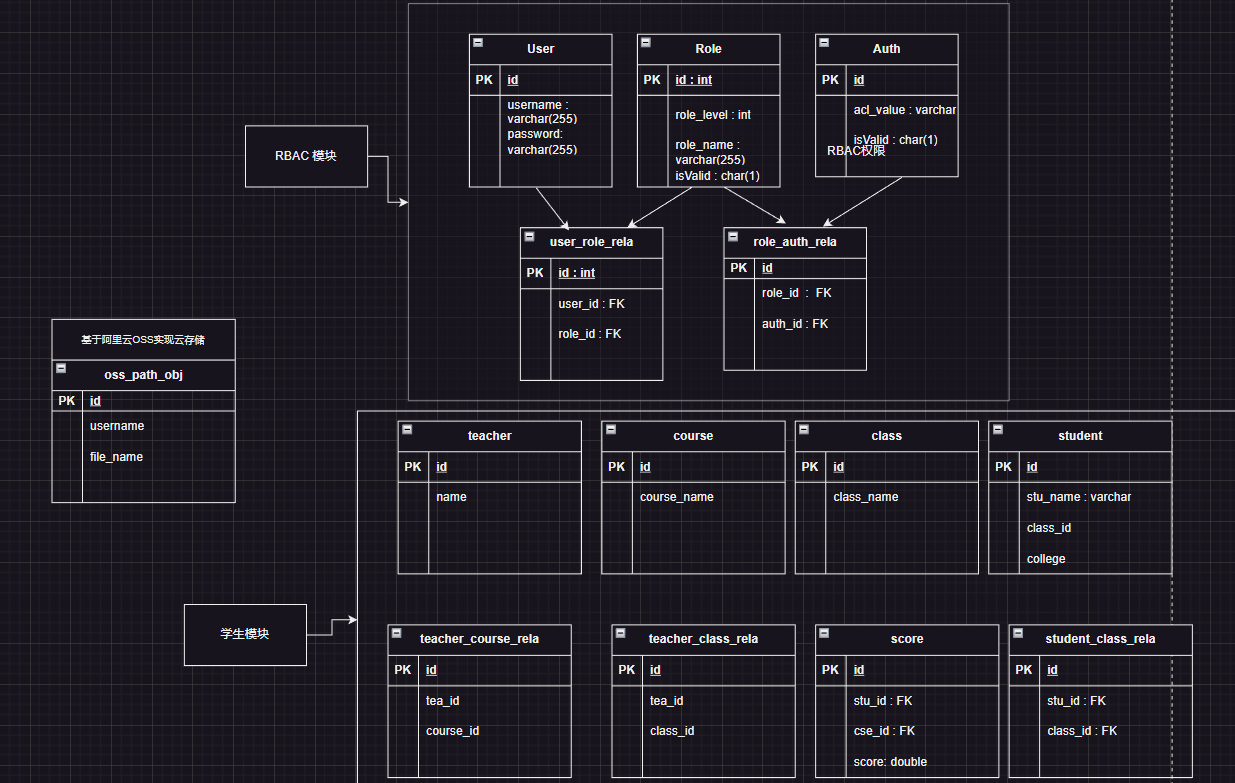
## 相关知识简介

从下往下: 数据库基于RBAC开展进行三个模块设计。白话介绍：就是不同的用户对应不同的角色进而有不同的权限，比如钉钉、腾讯会议等等。云存储直白一点就是将资源存放在互联网的cloud 资源上，随取随用。界面GUI实现，也就是桌面程序，CS架构，我们知道常见的这种CS架构，一般将主要程序放到用户本地，提高性能，必要的更新资源等配置会存放在互联网上。可以把他称为一个单点服务，只需要接入其他服务设备即可。  
 

## 功能设计

### 设计初衷以及思想

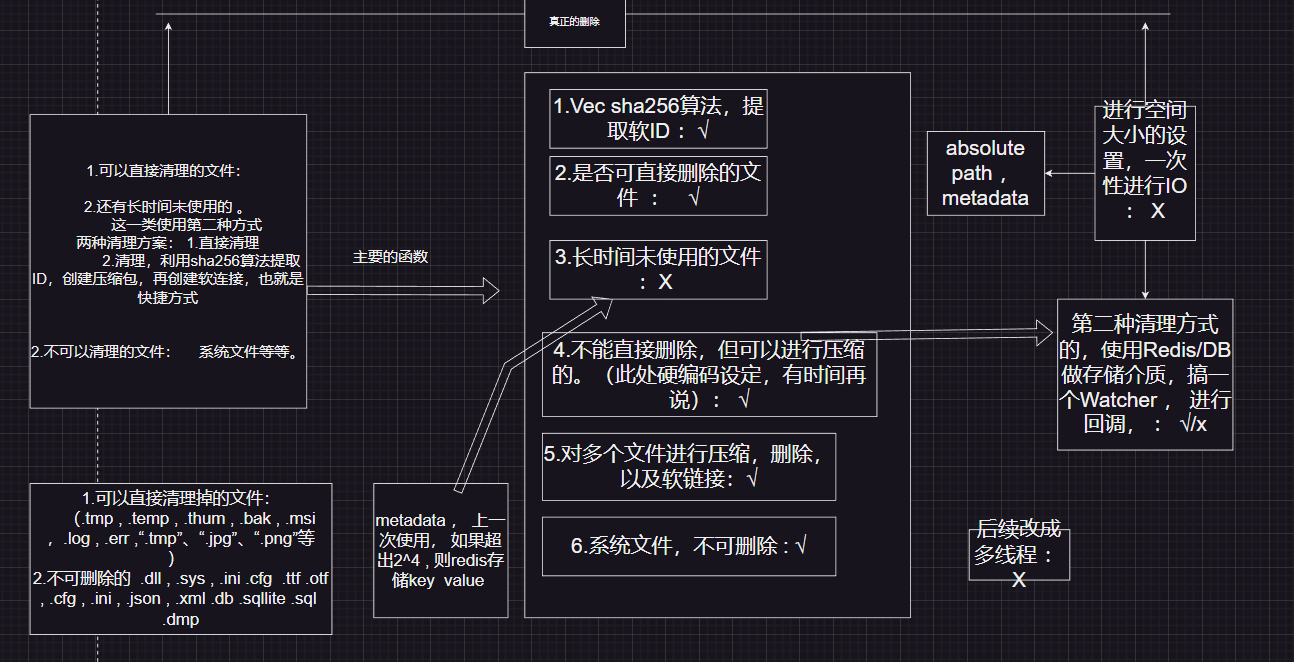
#### 数据库设计

按照三范式设计，User模块、Student模块、Oss文件模块  


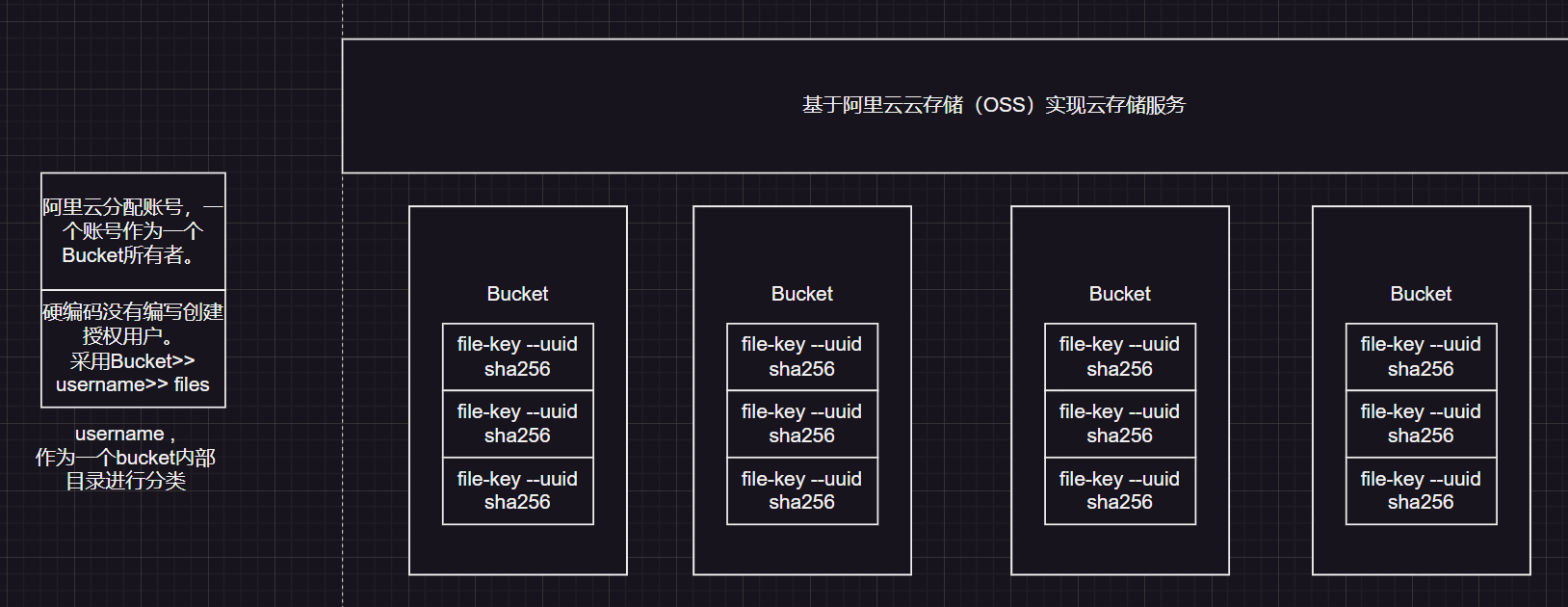
#### GUI绘画

这一块主要进行界面绘画和功能展望  


#### 文件清理

初衷： 像这种电脑管家清理，进行文件清理，遍历文件，渲染window，供用户选择。当然它内部肯定是多线程或者根据用户习惯高大上的机器学习和高效算法。   
都说python适合写小工具，打算完成一下，但由于其效率IO原因，采用Rust。python 解释性语言 IO效率较低,这一块认为建议低级程序语言来完成比较好，比如C、CPP、Rust。  


#### OSS云存储

初衷：学习过程时常需要笔记以及文件存储，对于百度阿里云盘来说，他们的云存储架构肯定要考虑实时性，并发性，备份等问题。平时会用到有道云笔记，对于这类软件我很青睐，就想自己做一个小工具玩玩，所以嵌入了这个模块  
此处使用阿里云OSS服务，对于Github 来说，也有云存储的特性，但是账号没权限，不能编码推送拉取；还有Amazon cloud storage，测试需要visa卡，没有对amazon进行测试.  
  


### 功能结构

#### 目录结构图

#### 代码结构

##### 配置文件工具类

为了移植性，打算写一个配置文件Util，一开始采用yml，对空格有效字符处理有些纰漏，最后打算用properties。

class ResourceBundle:

    properties = {}

    property\_list = []

    def \_\_init\_\_(self):

        print("can not be created to an instance")

        return

    @staticmethod

    def \_\_read\_pro(cls , file\_path:str):

        with open(file\_path) as fl:

            for line in fl:

                items = line.replace("\n", "").split("=")

                key = str(items[0]).split(".")

                # "{'key' : {"key2" :{ "key3" : }}}"

                cls.properties[key[0]] = {}

                length = len(key)

                tmp = ""

                for elem in key:

                    tmp += "{ '"

                    tmp += elem

                    tmp += "' :"

                value = str(items[1])

                tmp += "\'" + value + "\'"

                for i in range(length):

                    tmp += "}"

                cls.property\_list.append(eval(tmp))

    @classmethod

    def \_\_merge\_dict(cls):

        for prop in cls.property\_list:

            for key , value in dict(prop).items():

                if cls.properties[key]:

                    tmp = cls.properties[key]

                    for t1 , t2 in dict(value).items():

                        tmp[t1] = t2

                    cls.properties[key] = tmp

                else:

                    cls.properties[key] = dict(value)

    @classmethod

    def get\_bundle(cls ,path:str="../datasource.properties"):

        cls.\_\_read\_pro(cls=cls ,file\_path=path)

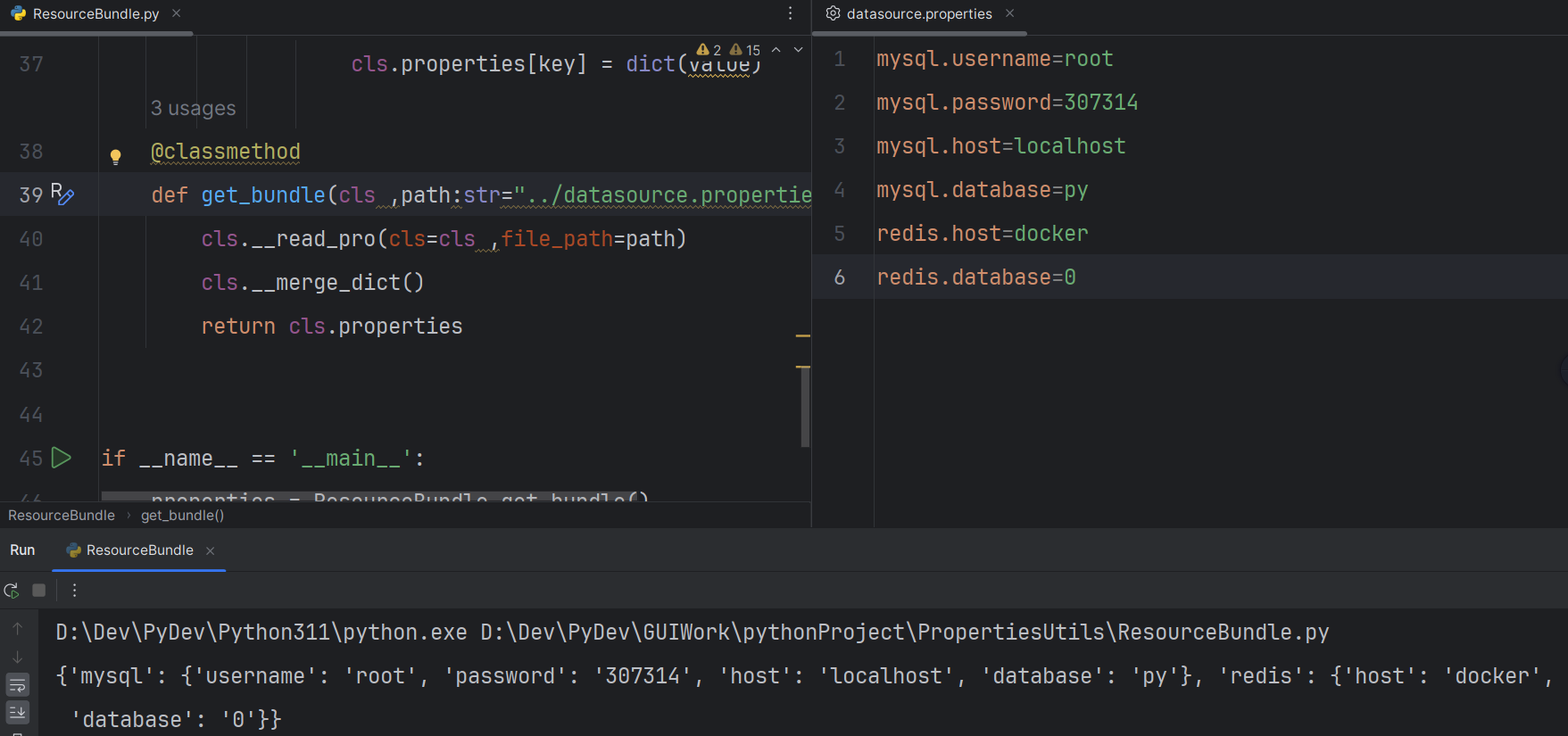
        cls.\_\_merge\_dict()

        return cls.properties

if \_\_name\_\_ == '\_\_main\_\_':

    properties = ResourceBundle.get\_bundle()

    print(properties)



##### OSS云存储

# -\*- coding: utf-8 -\*-

import os

import uuid

import oss2

from oss2.credentials import EnvironmentVariableCredentialsProvider

from sqlUtils.SqlSession import SqlSession as sql\_session

class OSS:

    cwd = os.getcwd().replace("\\", "/")

    auth = oss2.ProviderAuth(EnvironmentVariableCredentialsProvider())

    endpoint = 'https://oss-cn-hangzhou.aliyuncs.com'

    username = "lartimes"  # bucket\_name

    list\_tags = ["file\_oss\_list", "files\_oss\_info"]

    def \_\_init\_\_(self):

        print("OSS class can not be new to an instance")

        return

    @classmethod

    def does\_bucket\_exist(cls, bucket: oss2.Bucket):

        try:

            bucket.get\_bucket\_info()

        except oss2.exceptions.NoSuchBucket:

            return False

        except:

            raise

        return True

    @classmethod

    def put\_file\_oss(cls, path: str, user: str = "lartimes"):

        bucket = oss2.Bucket(OSS.auth, OSS.endpoint, OSS.username)

        if not OSS.does\_bucket\_exist(bucket):  # 不存在

            bucketConfig = oss2.models.BucketCreateConfig(oss2.BUCKET\_STORAGE\_CLASS\_STANDARD,

  oss2.BUCKET\_DATA\_REDUNDANCY\_TYPE\_ZRS)

            bucket.create\_bucket(oss2.BUCKET\_ACL\_PRIVATE, bucketConfig)

        # 生成 UUID / sha256 ， put file ， filePath ，

        list\_path = path.split(".")

        file\_name = path.split("/")[-1]

        obj\_name = user + "/journal/" + uuid.uuid4().hex + "." + list\_path[1]

        bucket.put\_object\_from\_file(obj\_name, path)

        sql\_session.insert\_file\_obj(user=user, file\_name=file\_name, obj\_name=obj\_name)

        os.remove(path)

    @classmethod

    def pull\_file\_to\_local(cls, file\_name: str, username: str = "lartimes", is\_download: bool = False, ):

        bucket = oss2.Bucket(OSS.auth, OSS.endpoint, OSS.username)

        # file\_name TODO 根据file——name 获取OSS filePath （obj\_name）

        local\_path = OSS.cwd + "/" + uuid.uuid4().hex + "." + file\_name.split(".")[1]

        bucket.get\_object\_to\_file(file\_name, local\_path)

        content = ""

        try:

            with open(local\_path, "r", encoding="utf8") as fl:

                result = fl.readlines()

                for line in result:

                    content += line

        except:

            print("不持支该格式直接读入")

        print(content)

        if not is\_download:            os.remove(local\_path)

        return (content, local\_path)

    @staticmethod

    def get\_files():

        files = sql\_session.get\_oss\_files()

        oss\_file = {}

        length = len(files)

        print(length)

        for index in range(length):

            oss\_file[str(files[index][0])] = str(files[index][1])

        return oss\_file

print(OSS.cwd)  # D:\Dev\PyDev\GUIWork\pythonProject\oss  # OSS.put\_file\_oss("D:/Dev/PyDev/GUIWork/pythonProject/oss/qweqweqqw.txt", "lartimes")  # OSS.pull\_file\_to\_local("lartimes","qweqweqqw.txt")  # render\_files()

##### NotePad 简单文本编辑器

实现简单文本编辑器，保存即上传

import os

import time

from tkinter import \*

from tkinter import filedialog

from tkinter import messagebox

from  oss.oss import  OSS as aliyun\_oss

from  userInfo.UserOssFiles import  render\_files

class NotePad:

    @staticmethod

    def nodefined():

        pass

    def openfile(self):

        filename = filedialog.askopenfilename()

        f = open(filename, 'r')

        f2 = f.read()

        f.close()

        self.text.insert(INSERT, f2)

    def savefileas(self):

        filename = filedialog.asksaveasfilename(filetypes=[("TXT", ".txt")]) + ".txt"

        print(filename)

        content = self.text.get('0.0', 'end')

        print(content)

        with open(filename, 'w' ,encoding="utf8") as f:

            f.write(content)

            f.flush()

    # TODO filename ---> 云存储， 渲染

        print(filename)

        aliyun\_oss.put\_file\_oss(filename)

        render\_files()

    def quit(self):

        self.root.destroy()

    # 复制功能函数

    def copy(self):

        global content

        content = self.text.get(SEL\_FIRST, SEL\_LAST)

        return content

    def cut(self):

        global content

        content = self.text.get(SEL\_FIRST, SEL\_LAST)

        self.text.delete(SEL\_FIRST, SEL\_LAST)

        return content

    # 粘贴功能函数

    def paste(self):

        self.text.insert(INSERT, content)

    @staticmethod

    def about():

        messagebox.showinfo("关于", "开发者：Lartimes")

    def popup(self, event):

        self.popupmenu.post(event.x\_root, event.y\_root)

    def \_\_init\_\_(self):

        # 顶级菜单窗口

        self.root = Tk()

        self.topmenu = Menu(self.root)

        self.root.title("Text Editor")

        # 文字编辑区text

        self.text = Text(self.root, width=90, height=40, selectforeground="black", undo=True, font=50)

        self.text.pack()

        self.menu\_render()

    def menu\_render(self):

        # 创建文件下拉菜单，添加到顶层菜单窗口

        filemenu = Menu(self.topmenu, tearoff=False)

        # 添加下拉内容：

        filemenu.add("command", label="打开", command=self.openfile)

        filemenu.add\_command(label="保存/上传", command=self.savefileas)

        filemenu.add\_command(label="另存为", command=self.savefileas)

        filemenu.add\_separator()

        filemenu.add\_separator()

        filemenu.add\_command(label="退出", command=quit)

        self.topmenu.add\_cascade(label="文件", menu=filemenu)

        editmenu = Menu(self.topmenu, tearoff=False)

        editmenu.add\_separator()

        editmenu.add\_command(label="撤销", command=self.callback)

        editmenu.add("command", label="剪切", command=self.cut)

        editmenu.add\_command(label="复制", command=self.copy)

        editmenu.add\_command(label="粘贴", command=self.paste)

        editmenu.add\_separator()

        editmenu.add\_command(label="查找", command=NotePad.nodefined)

        editmenu.add\_command(label="替换", command=NotePad.nodefined)

        editmenu.add\_command(label="转到", command=NotePad.nodefined)

        editmenu.add\_separator()

        self.topmenu.add\_cascade(label="编辑", menu=editmenu)

        formatmenu = Menu(self.topmenu, tearoff=False)

        self.topmenu.add\_cascade(label="格式", menu=formatmenu)

        viewmenu = Menu(self.topmenu, tearoff=False)

        viewmenu.add\_command(label="查看状态栏", command=self.callback)

        self.topmenu.add\_cascade(label="查看", menu=viewmenu)

        helpmenu = Menu(self.topmenu, tearoff=False)

        helpmenu.add\_command(label="查看帮助", command=self.callback)

        helpmenu.add\_separator()

        helpmenu.add\_command(label="关于笔记本", command=NotePad.about)

        self.topmenu.add\_cascade(label="帮助", menu=helpmenu)

        self.popupmenu = Menu(self.root, tearoff=False)

        self.popupmenu.add\_command(label="保存", command=self.savefileas)

        self.popupmenu.add\_command(label="另存为", command=self.savefileas)

        self.popupmenu.add\_separator()

        self.popupmenu.add\_command(label="撤回", command=self.callback)

        self.popupmenu.add\_separator()

        self.popupmenu.add("command", label="剪切", command=self.cut)

        self.popupmenu.add\_command(label="复制", command=self.copy)

        self.popupmenu.add\_command(label="粘贴", command=self.paste)

        self.popupmenu.add("command", label="删除", command=self.textdelete)

    # 删除函数

    def textdelete(self, text: Text):

        text.delete(SEL\_FIRST, SEL\_LAST)

    def callback(self):

        self.text.edit\_undo()

def start\_notepad():

    notepad = NotePad()

    notepad.text.bind("<Button-3>", notepad.popup)

    notepad.root.config(menu=notepad.topmenu)

    mainloop()

if \_\_name\_\_ == '\_\_main\_\_':

    start\_notepad()

##### Excel 导入导出

import os

import uuid

import pandas as pd

from sqlUtils.SqlSession import SqlSession as sql\_session

class DataUtils:

    dict\_column = {"stu\_name": 0, "college": 1, "class\_name": 2, "Java Course": 3,

                   "Python Course": 4, "Rust Course": 5}

    course\_list = ["Java Course", "Python Course", "Rust Course"]

    def \_\_init\_\_(self):

        print("this class can not be created as an instance")

        return

    @classmethod

    def import\_xls(cls, path="C:/Users/33769/Desktop/stu.xlsx"):  # [{}  , {} , {}]

        # path = "C:/Users/33769/Desktop/student.xlsx"

        # //TODO sheet name 多个sheet name

        xls = pd.read\_excel(path)

        data = []

        for (row\_id, row\_info) in xls.iterrows():

            temp = {}

            for (column\_name, index) in DataUtils.dict\_column.items():

                temp[column\_name] = row\_info[index]

            data.append(temp)

        # list(dict)

        # sql insert

        cols = list(DataUtils.dict\_column.keys())

        course\_ids = sql\_session.get\_course\_id(cols[3:])

        course\_id\_dict = {}

        for index in range(len(course\_ids)):

            course\_id\_dict[str(course\_ids[index][1])] = course\_ids[index][0]

        # stu\_id   stu\_name course\_name   score , tuple()

        #  0-=stu\_id   stu\_name ava  cpp rust\_file python

        print(course\_id\_dict)

        # list(dict())

        # course = sql\_session

        # stu\_name': 'ALAN1', 'college': '电子信息工程学院', 'class\_name': '大数据222',

        # 'Java Course': 10, 'Python Course': 70, 'Rust Course': 90}

        for datum in data:

            stu\_id = sql\_session.add\_stu(datum[cols[0]], datum[cols[2]], datum[cols[1]])

            print("stu\_id", stu\_id)

            # stu\_id , course\_id , score

            for key, value in course\_id\_dict.items():

                print(stu\_id)

                print(value)

                print(datum[key])

                sql\_session.insert\_score\_by\_id(int(stu\_id), int(value), datum[key])

        print(data)

        return data

    @staticmethod

    def export\_xls():

        project\_path = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))  # 获取当前工作顶级目录

        print(project\_path)

        output\_file = os.path.join(project\_path, uuid.uuid4().hex + 'stu.xlsx')  # uuid + stu.xlsx

        print(output\_file)

        ret = sql\_session.export\_all()

        length = len(ret)

        course\_len = len(DataUtils.course\_list)

        java\_list = []

        py\_list = []

        rs\_list = []

        for i in range(length):

            course\_name = str(ret[i][4])

            if course\_name == DataUtils.course\_list[0]:

                java\_list.append(ret[i][3])

            elif course\_name == DataUtils.course\_list[1]:

                py\_list.append(ret[i][3])

            elif course\_name == DataUtils.course\_list[2]:

                rs\_list.append(ret[i][3])

        rows = length / course\_len  # row , result , 读取course ，

        data = {}  # export -- 》 {column , []}

        for column\_name, index in DataUtils.dict\_column.items():

            #             每一列进行填充 0 1 2  length

            tmp = []

            for i in range(int(rows)):  # 0 1 2 row

                if index < 3:

                    print(ret[(i - 1) \* course\_len][index])

                    tmp.append(ret[i][index])

                    continue

                # 读取课程score

                break

            if tmp:

                data[column\_name] = tmp

        data[DataUtils.course\_list[0]] = java\_list

        data[DataUtils.course\_list[1]] = py\_list

        data[DataUtils.course\_list[2]] = rs\_list

        print(output\_file)

        for datum in data.values():

            print(datum)

        df = pd.DataFrame(data)

        df.to\_excel(output\_file, index=False)

        return  output\_file

if \_\_name\_\_ == '\_\_main\_\_':

    print(list(DataUtils.dict\_column.keys())[3:])

    DataUtils.import\_xls("C:/Users/33769/Desktop/stu.xlsx")

    DataUtils.export\_xls()

##### SQL

import pymysql

from  PropertiesUtils.ResourceBundle import ResourceBundle as resource\_bundle

class SqlSession:

    # username password database host

    properties = resource\_bundle.get\_bundle()

    username = properties['mysql']['username']

    password = properties['mysql']['password']

    database = properties['mysql']['database']

    host = properties['mysql']['host']

    @classmethod

    def del\_by\_stu\_id(cls, stu\_id):

        cursor = None

        conn = None

        try:

            conn = SqlSession.get\_conn()

            cursor = conn.cursor()

            sql = f"delete from student where id = {stu\_id}"

            cursor.execute(sql)

            conn.commit()

        except:

            if conn is not None:

                conn.rollback()

        finally:

            if conn is not None:

                conn.close()

    @classmethod

    def get\_stu\_id(cls, name, college):

        cursor = None

        conn = None

        try:

            conn = SqlSession.get\_conn()

            cursor = conn.cursor()

            sql = f"insert into student (name , college) values ('{name}','{college}')"

            cursor.execute(sql)

            conn.commit()

            return int(cursor.lastrowid)

        except:

            if conn is not None:

                conn.rollback()

        finally:

            if conn is not None:

                conn.close()

    @classmethod

    def get\_class\_id(cls, class\_name):

        cursor = None

        conn = None

        try:

            conn = SqlSession.get\_conn()

            cursor = conn.cursor()

            sql = f"""SELECT id  from class where class\_name = '{class\_name}'"""

            cursor.execute(sql)

            ret = cursor.fetchone()

            return int(ret[0])

        except:

            if conn is not None:

                conn.rollback()

        finally:

            if conn is not None:

                conn.close()

    @classmethod

    def add\_stu(cls, name="", class\_name="", college=""):

        cursor = None

        conn = None

        try:

            conn = SqlSession.get\_conn()

            cursor = conn.cursor()

            class\_id = SqlSession.get\_class\_id(class\_name)

            print(class\_id)

            pri\_key = SqlSession.get\_stu\_id(name, college)

            sql = f"insert into student\_class\_rela values(NUll, {pri\_key} , {class\_id} ) "

            print(sql)

            cursor.execute(sql)

            conn.commit()

            return pri\_key

        except:

            if conn is not None:

                conn.rollback()

        finally:

            if conn is not None:

                conn.close()

    @classmethod

    def get\_score(cls, name="lartimes2099", page=1, page\_size=8):  # TODO pagesize 分页查询

        cursor = None

        conn = None

        try:

            start\_index = (page - 1) \* page\_size;

            limit = f"limit {start\_index} , {page\_size}"

            conn = SqlSession.get\_conn()

            cursor = conn.cursor()

            sql = f"""

                    SELECT  DISTINCT stu\_name , cl.class\_name  as class\_name

                    , college , sum\_score  , tmp.stu\_id

            FROM  (SELECT

                        s.id  as stu\_id , s.`name` as stu\_name , s.college

                            as college , SUM(sc.score) as sum\_score

                    FROM

                        student s LEFT JOIN score sc on sc.stu\_id = s.id

                    GROUP BY s.id)

            tmp

            join  student\_class\_rela re on tmp.stu\_id = re.stu\_id

            join class cl  on cl.id = re.class\_id

            ORDER BY tmp.sum\_score DESC {limit}

                    """

            cursor.execute(sql)

            data = cursor.fetchall()

            return data

        except:

            if conn is not None:

                conn.rollback()

        finally:

            if conn is not None:

                conn.close()

    @classmethod

    def get\_conn(cls, host=host, user=username,

                 password=password,

                 database=database,

                 port=3306):

        return pymysql.connect(host=host,

                               user=user,

                               password=password,

                               database=database,

                               port=port)

    @classmethod

    def get\_auth(cls, role\_id=1):

        cursor = None

        conn = None

        try:

            conn = SqlSession.get\_conn()

            cursor = conn.cursor()

            sql = f"""

                             SELECT  a.acl\_value

                            FROM role\_auth\_rela ra

                            JOIN    auth a

                            on ra.role\_id = {role\_id} and a.id = ra.auth\_id

                            """

            cursor.execute(sql)

            data = cursor.fetchall()

            return data

        except:

            if conn is not None:

                conn.rollback()

        finally:

            if conn is not None:

                conn.close()

    @classmethod

    def login(cls, username="lartimes", password="2004"):

        cursor = None

        conn = None

        try:

            conn = SqlSession.get\_conn()

            cursor = conn.cursor()

            sql = f"""

                      SELECT u.username ,r.role\_name , r.id

                    FROM

                        user u

                    inner JOIN

                        user\_role\_rela ur

                    on u.username = '{username}' and u.password = '{password}'

                    and u.id = ur.user\_id

                    join

                     role r

                    on r.id = ur.role\_id and r.is\_valid = '1'

                    order by  r.role\_level desc

                    """

            print("login:" + sql)

            cursor.execute(sql)

            data = cursor.fetchone()

            return data

        except:

            if conn is not None:

                conn.rollback()

        finally:

            if conn is not None:

                conn.close()

    @classmethod

    def update\_stu(cls, stu\_id, name, class\_name, college):

        cursor = None

        conn = None

        try:

            new\_cls\_id = SqlSession.get\_class\_id(class\_name)

            conn = SqlSession.get\_conn()

            cursor = conn.cursor()

            sql = f"select class\_id from  student\_class\_rela where stu\_id = {stu\_id}"

            cursor.execute(sql)

            cls\_id = int(cursor.fetchone()[0])

            sql1 = f"""

                update student set name = '{name}' , college = '{college}' where id = {stu\_id} ;

                """

            SqlSession.modify\_stu(sql1);

            sql2 = f"""

                update student\_class\_rela set class\_id = {new\_cls\_id} where  class\_id = {cls\_id};

                """

            cursor.execute(sql2)

            conn.commit()

        except:

            if conn is not None:

                conn.rollback()

        finally:

            if conn is not None:

                conn.close()

    @classmethod

    def modify\_stu(cls, sql):

        cursor = None

        conn = None

        try:

            conn = SqlSession.get\_conn()

            cursor = conn.cursor()

            cursor.execute(sql)

            conn.commit()

        except:

            if conn is not None:

                conn.rollback()

        finally:

            if conn is not None:

                conn.close()

    @classmethod

    def check\_score(cls, stu\_id=1):

        cursor = None

        conn = None

        try:

            conn = SqlSession.get\_conn()

            cursor = conn.cursor()

            sql = f"""

            SELECT s.name , cr.` name` , sc.score

            FROM  student s  join  score sc

            on s.id = {stu\_id} and sc.stu\_id = s.id

            join    course cr

            on cr.id  = sc.course\_id

            ORDER BY sc.score ASC

            """

            cursor.execute(sql)

            return cursor.fetchall()

        except:

            if conn is not None:

                conn.rollback()

        finally:

            if conn is not None:

                conn.close()

    @classmethod

    def select\_like(cls, name, class\_name, college):

        cursor = None

        conn = None

        name\_sql = ""

        college\_sql = ""

        class\_sql = ""

        if not name\_sql:

            name\_sql = f"and   s.`name` like '%{name}%' "

        if not class\_sql:

            print("class\_sql =================")

            class\_sql = f"and  c.class\_name like '%{class\_name}%'"

        if not college\_sql:

            college\_sql = f"and s.college like '%{college}%' "

        #   and   s.`name` like '%{name}%'

        #       and s.college like '%{college}%'

        #       and  c.class\_name like '%{class\_name}%'

        try:

            conn = SqlSession.get\_conn()

            cursor = conn.cursor()

            sql = f"""

        select s.name as stu\_name  ,

        c.class\_name as class\_name ,

        s.college as college\_name ,

        sum(sr.score)  as sum\_score ,

        s.id as stu\_id

        FROM  student s

        JOIN    class c

        on 1 = 1

        {name\_sql} {college\_sql} {class\_sql}

        JOIN score  sr

        on  sr.stu\_id = s.id

        GROUP BY s.id , c.class\_name

        limit 0 , 8

              """

            cursor.execute(sql)

            return cursor.fetchall()

        except:

            if conn is not None:

                conn.rollback()

        finally:

            if conn is not None:

                conn.close()

    @classmethod

    def export\_all(cls):

        cursor = None

        conn = None

        try:

            conn = SqlSession.get\_conn()

            cursor = conn.cursor()

            sql = """  select  s.name as stu\_name  ,

                    s.college as college\_name ,

                  c.class\_name as class\_name ,

                  sr.score  as score,

                    cr.` name` as course\_name

                FROM  student s

                 JOIN   student\_class\_rela sc on sc.stu\_id = s.id

                 JOIN  class c ON sc.class\_id = c.id

                 join  score sr on s.id = sr.stu\_id

                 join  course cr on cr.id = sr.course\_id

                 order by  s.name , course\_name ASC"""

            cursor.execute(sql)

            result = cursor.fetchall()

            print(result)

            return result

        except:

            if conn is not None:

                conn.rollback()

        finally:

            if conn is not None:

                conn.close()

    @classmethod

    def get\_course\_id(cls, list\_course: list):  # list(course)

        cursor = None

        conn = None

        try:

            conn = SqlSession.get\_conn()

            cursor = conn.cursor()

            temp = "("

            for e in list\_course:

                temp += '\''

                temp += e

                temp += '\''

                temp += ","

            temp = temp[:-1]

            temp += ")"

            sql = f""" SELECT \* FROM course  where ` name` in {temp} """

            cursor.execute(sql)

            result = cursor.fetchall()

            print(result)

            return result

        except:

            if conn is not None:

                conn.rollback()

        finally:

            if conn is not None:

                conn.close()

    @classmethod

    def insert\_score\_by\_id(cls, stu\_id: int, cr\_id: int, score):

        cursor = None

        conn = None

        try:

            conn = SqlSession.get\_conn()

            cursor = conn.cursor()

            sql = f"""insert into score values (NULL , {stu\_id} , {cr\_id}  , {score})"""

            cursor.execute(sql)

            conn.commit()

        except:

            if conn is not None:

                conn.rollback()

        finally:

            if conn is not None:

                conn.close()

    #  # user , file\_name , obj\_name

    @classmethod

    def insert\_file\_obj(cls, user: str, file\_name: str, obj\_name: str):

        cursor = None

        conn = None

        try:

            conn = SqlSession.get\_conn()

            cursor = conn.cursor()

            sql = f"""

            insert into oss\_path\_obj values (NULL , '{user}' , '{file\_name}'  , '{obj\_name}' )

            """

            cursor.execute(sql)

            conn.commit()

        except:

            if conn is not None:

                conn.rollback()

        finally:

            if conn is not None:

                conn.close()

    @classmethod

    def get\_obj\_path(cls, user: str, file\_name: str):

        cursor = None

        conn = None

        try:

            conn = SqlSession.get\_conn()

            cursor = conn.cursor()

            sql = f"""

              select obj\_name from oss\_path\_obj where username  = '{user}'

              and file\_name = '{file\_name}'

               """

            cursor.execute(sql)

            conn.commit()

            return cursor.fetchone()[0]

        except:

            if conn is not None:

                conn.rollback()

        finally:

            if conn is not None:

                conn.close()

    @classmethod

    def get\_stu\_count(cls):

        cursor = None

        conn = None

        try:

            conn = SqlSession.get\_conn()

            cursor = conn.cursor()

            sql = f"""

               select count(\*) from student

                 """

            cursor.execute(sql)

            return cursor.fetchone()[0]

        except:

            if conn is not None:

                conn.rollback()

        finally:

            if conn is not None:

                conn.close()

    @classmethod

    def get\_oss\_files(cls , username:str="lartimes"):

        cursor = None

        conn = None

        try:

            conn = SqlSession.get\_conn()

            cursor = conn.cursor()

            sql = f"""

                   select  file\_name , obj\_name  from oss\_path\_obj

                   where  username = '{username}'

                     """

            cursor.execute(sql)

            return cursor.fetchall()

        except:

            if conn is not None:

                conn.rollback()

        finally:

            if conn is not None:

                conn.close()

##### system 模块(学生信息)

###### pages-windows.py--- 渲染学生分页查询模块

import dearpygui.dearpygui as dpg

import system.student\_info as stu\_inf

def page\_flush(sender, app\_data, user\_data):

    stu\_inf.stu\_render(None, user\_data)

def pages\_render(page\_num: int, total\_pages: int):

    #   page\_num <= total\_pages

    print("pages\_render", page\_num)

    print(total\_pages)

    page\_up = 1

    page\_down = 1

    pages = []

    if page\_num != 1:

        pages.append(page\_num - 1)

        page\_up = page\_num - 1

    pages.append(page\_num)

    if total\_pages - page\_num >= 1:

        pages.append(page\_num + 1)

        page\_down = page\_num + 1

    with dpg.window(label="pages\_info\_window", pos=(200, 600), width=750, height=100, indent=100, no\_resize=True,

                    no\_title\_bar=True, no\_move=True):

        with dpg.table(label="pages", header\_row=False, row\_background=True, borders\_innerH=True, borders\_outerH=True,

                       borders\_innerV=True, borders\_outerV=True, width=750, height=80):

            for e in range(5):

                dpg.add\_table\_column(width=100)

            with dpg.table\_row(height=40):

                dpg.add\_button(label="page up", width=100, height=40, callback=page\_flush, user\_data=page\_up)

                for page in pages:

                    dpg.add\_button(label=f"{page}", width=100, height=40, callback=page\_flush, user\_data=page)

                dpg.add\_button(label="page down", width=100, height=40, callback=page\_flush, user\_data=page\_down)

###### search\_bar.py –搜索栏

import dearpygui.dearpygui as dpg

import system.student\_info as stu\_info

from sqlUtils.SqlSession import SqlSession as sql\_session

list\_cases = ["search\_name", "search\_class", "search\_college"]

def search\_stu\_info():

    name = str(dpg.get\_value(list\_cases[0]))

    class\_name = str(dpg.get\_value(list\_cases[1]))

    college = str(dpg.get\_value(list\_cases[2]))

    if len(name) or len(class\_name) or len(college):

        data = sql\_session.select\_like(name, class\_name, college)

        stu\_info.stu\_render(data)  # 默认第一页数

    else:

        stu\_info.stu\_render(None)

def search\_render():

    with   dpg.window(label="welcome\_window", pos=(200, 100), width=1000, height=100, no\_move=True, no\_resize=True,

                      no\_title\_bar=True, ):

        with dpg.table(header\_row=False, row\_background=True, borders\_innerH=True, borders\_outerH=True,

                       borders\_innerV=True, borders\_outerV=True, width=750, height=80):

            dpg.add\_table\_column(width=250, )

            dpg.add\_table\_column(width=250, )

            dpg.add\_table\_column(width=250, )

            for i in range(2):

                with dpg.table\_row(height=40):

                    if i:

                        for list\_case in list\_cases:

                            dpg.add\_input\_text(width=180, indent=30, tag=list\_case)

                    else:  # // 0

                        for list\_case in list\_cases:

                            dpg.add\_text(list\_case)

        dpg.add\_button(label="search button", pos=(760, 10), width=200, height=75, callback=search\_stu\_info)

###### student\_info,.py –详细信息

import math

import dearpygui.dearpygui as dpg

import system.pages\_window as pages

from sqlUtils.SqlSession import SqlSession as sql\_session

def check\_info(sender, app\_data, user\_data):

    if not dpg.get\_value("check\_info"):

        dpg.delete\_item("check\_info")

    # 模态窗口

    scores = sql\_session.check\_score(int(user\_data))

    length = len(scores)

    if length:

        with dpg.window(label="student info", tag="check\_info", width=800, height=400,

                        modal=True, pos=(200, 200), no\_move=True):

            dpg.add\_text(str(scores[0][0]) + "'s personal scores")

            for i in range(length):

                dpg.add\_text(str(scores[i][1]))

                dpg.add\_text(str(scores[i][2]), indent=100)

list\_tags = ["check\_info", "stu\_info\_tag"]

def stu\_render(data=None, page\_num=1):

    for list\_tag in list\_tags:

        dpg.delete\_item(list\_tag)

    with dpg.window(label="stu\_info ",

                    pos=(200, 200),

                    width=750,

                    height=400,

                    indent=100,

                    no\_resize=True, no\_title\_bar=True,

                    no\_move=True, no\_scrollbar=True, no\_scroll\_with\_mouse=True):

        # 根据数据查询 设置column counts

        # 有什么字段？

        with dpg.table(header\_row=False, row\_background=True, borders\_innerH=True, borders\_outerH=True,

                       borders\_innerV=True, borders\_outerV=True,

                       width=750,

                       height=80,

                       tag="stu\_info\_tag"):

            list\_column = ["stu\_name", "class\_name", "college\_name", "sum\_score", "rank", "auth\_btn"]

            for col in list\_column:

                dpg.add\_table\_column(label=col, width=100)

            # [list(map)]

            if data is None:

                data = sql\_session.get\_score(page=page\_num)

                print(data)

            result = []

            length = len(data)  # 页数

            count = sql\_session.get\_stu\_count()

            total\_pages = math.ceil(count / 8)

            print("total pages " ,total\_pages)

            print(count)

            for i in range(length):

                tmp = {}

                for j in range(5):

                    if data[i] is not None:

                        if list\_column[j] == "rank":

                            tmp[list\_column[-1]] = data[i][j]

                            break

                        tmp[list\_column[j]] = data[i][j]

                result.append(tmp)

            print(result)

            if length:

                for i in range(length + 1):  # size : list\_len

                    with dpg.table\_row(height=40):

                        print(i)

                        if i:  # 真的i ==> rank

                            for col in list\_column:

                                if col == "rank":

                                    dpg.add\_text(str(i))

                                else:

                                    if col == "auth\_btn":

                                        dpg.add\_button(user\_data=str(result[i - 1][col]), label="check stu info",

                                                       callback=check\_info)

                                        continue

                                    dpg.add\_text(str(result[i - 1][col]))

                        else:

                            for j in list\_column:

                                dpg.add\_text(j)

            # pagess

            pages.pages\_render(page\_num, total\_pages) #渲染page window

###### 欢迎标语 welcome.py

import dearpygui.dearpygui as dpg

def welcome\_window():

    with dpg.window(label="welcome\_window", pos=(200, 0), width=1000, height=100, no\_move=True, no\_resize=True,

                    no\_title\_bar=True, ):

        title = dpg.add\_text("welcome to the student management system", indent=20, )

        dpg.set\_item\_indent(title, 250)

        return title

##### User Info模块

###### UserAuth\_window,.py 用户权限菜单模块

import os

import time

import dearpygui.dearpygui as dpg

import thread

import system.student\_info as stu\_info

from excel.ImEXportUtils import DataUtils as data\_utils

from notepad.note\_pad import start\_notepad

from oss.oss import OSS as aliyun\_oss

from sqlUtils.SqlSession import SqlSession as sql\_session

from userInfo.UserOssFiles import render\_files

def open\_rust():

    workdir = os.getcwd()

    workdir = workdir.replace("\\", "/")

    workdir += "/rust\_file/FileSystem.exe"

    print(workdir)

    os.system(f"{workdir}")

def delete\_stu():

    stu\_id = dpg.get\_value("del\_stu:del\_stu\_id")  # 数据库已经设置为cascade atomic原子操作 级联删除

    sql\_session.del\_by\_stu\_id(int(stu\_id))

    dpg.delete\_item("tmp2")

    stu\_info.stu\_render(None)

def update\_stu():

    list\_stu = ["modify\_stu:stu\_id", "modify\_stu:name", "modify\_stu:class\_name", "modify\_stu:college", ]

    stu\_id = int(dpg.get\_value(list\_stu[0]))

    name = str(dpg.get\_value(list\_stu[1]))

    class\_name = str(dpg.get\_value(list\_stu[2]))

    college = str(dpg.get\_value(list\_stu[3]))

    sql\_session.update\_stu(stu\_id, name, class\_name, college)

    dpg.delete\_item("tmp4")

    stu\_info.stu\_render(None)

def modify\_stu():

    if not dpg.get\_value("tmp4"):

        dpg.delete\_item("tmp4")

    print("=======================")

    with dpg.window(label="update student", tag="tmp4", width=800, height=400, modal=True, pos=(200, 200),

                    no\_move=True):

        list\_stu = ["stu\_id", "name", "class\_name", "college", ]

        for tag in list\_stu:

            dpg.delete\_item(tag)

        theme = "modify\_stu"

        with dpg.table(header\_row=False):

            dpg.add\_table\_column()

            dpg.add\_table\_column()

            len = list\_stu.\_\_len\_\_() + 1

            for i in range(len):

                with dpg.table\_row():

                    if i != len - 1:

                        for j in range(0, 2):

                            if j:

                                dpg.add\_input\_text(tag=theme + ":" + list\_stu[i], width=300, height=100)

                            else:

                                dpg.add\_text(list\_stu[i])

                    else:

                        for j in range(0, 2):

                            if j:

                                dpg.add\_button(width=100, height=30, label="cancel",

                                               callback=lambda: dpg.configure\_item("tmp4", show=False))

                            else:

                                dpg.add\_button(label="submit", width=100, height=30, callback=update\_stu)

def del\_stu():

    if not dpg.get\_value("tmp2"):

        dpg.delete\_item("tmp2")

    with dpg.window(label="delete student", tag="tmp2", width=800, height=400, modal=True, pos=(200, 200),

                    no\_move=True):

        theme = "del\_stu"

        list\_tags = [theme + ":del\_stu\_id"]

        for list\_tag in list\_tags:

            dpg.delete\_item(list\_tag)

        with dpg.table(header\_row=False):

            dpg.add\_table\_column()

            for i in range(3):

                with dpg.table\_row():

                    if i:

                        if i == 2:

                            dpg.add\_button(label="confirm delete", callback=delete\_stu)

                            return

                        dpg.add\_input\_text(tag=list\_tags[0], width=300, height=100, indent=100)

                    else:

                        dpg.add\_text("id of the student that you want to delete..")

    stu\_info.stu\_render(None)

def add\_student():

    list\_tags = ["add\_stu:name", "add\_stu:class\_name", "add\_stu:college"]

    args = []

    for list\_tag in list\_tags:

        args.append(str(dpg.get\_value(list\_tag)))

    print(args)

    sql\_session.add\_stu(name=args[0], class\_name=args[1], college=args[2])

    for list\_tag in list\_tags:

        dpg.delete\_item(list\_tag)

    # 删除tmp1 tag

    dpg.delete\_item("tmp1")

    stu\_info.stu\_render(None)

def add\_stu():

    if not dpg.get\_value("tmp1"):

        dpg.delete\_item("tmp1")

    with dpg.window(label="add student", tag="tmp1", width=800, height=400, modal=True, pos=(200, 200), no\_move=True):

        #          stu : name , class\_name college  , head\_teacher

        list\_stu = ["name", "class\_name", "college"]

        for e in list\_stu:

            dpg.delete\_item(e)

        theme = "add\_stu"

        with dpg.table(header\_row=False):

            dpg.add\_table\_column()

            dpg.add\_table\_column()

            len = list\_stu.\_\_len\_\_() + 1

            for i in range(len):

                with dpg.table\_row():

                    if i != len - 1:

                        for j in range(0, 2):

                            if j:

                                dpg.add\_input\_text(tag=theme + ":" + list\_stu[i], width=300, height=100)

                            else:

                                dpg.add\_text(list\_stu[i], )

                    else:

                        for j in range(0, 2):

                            if j:

                                dpg.add\_button(label="cancel", width=100, height=30,

                                               callback=lambda: dpg.configure\_item("tmp1", show=False))

                            else:

                                dpg.add\_button(tag=theme + "submit", label="submit", width=100, height=30,

                                               callback=add\_student)

user\_bucket = "lartimes"

def upload(sender, app\_data):

    file\_dict = dict(app\_data['selections'])

    print(file\_dict.values())

    file\_path = list(file\_dict.values())[0].replace("\\", "/")

    print(file\_path)

    aliyun\_oss.put\_file\_oss(file\_path, user\_bucket)

    render\_files()

def imp\_xlsx(sender, app\_data):

    file\_dict = dict(app\_data['selections'])

    print(file\_dict.values())

    file\_path = list(file\_dict.values())[0].replace("\\", "/")

    print(file\_path + "========excel===========")

    data\_utils.import\_xls(path=file\_path)

    stu\_info.stu\_render()

def import\_excel():

    #     TODO 主要逻辑进行

    dpg.delete\_item("import\_excel")

    with dpg.file\_dialog(directory\_selector=False, show=False, callback=imp\_xlsx, file\_count=3, tag="import\_excel",

                         width=700, height=400):

        dpg.add\_file\_extension(".xlsx", color=(0, 255, 0, 255))

    dpg.show\_item("import\_excel")

def open\_selector():

    #     TODO 主要逻辑进行

    dpg.delete\_item("file\_dialog\_tag")

    with dpg.file\_dialog(directory\_selector=False, show=False, callback=upload, file\_count=3, tag="file\_dialog\_tag",

                         width=700, height=400):

        dpg.add\_file\_extension("", color=(255, 150, 150, 255))

        dpg.add\_file\_extension(".\*")

        dpg.add\_file\_extension(".cpp", color=(255, 255, 0, 255))

        dpg.add\_file\_extension(".h", color=(255, 0, 255, 255))

        dpg.add\_file\_extension(".java", color=(200, 0, 255, 255))

        dpg.add\_file\_extension(".rs", color=(150, 0, 200, 255))

        dpg.add\_file\_extension(".py", color=(0, 255, 0, 255))

    dpg.show\_item("file\_dialog\_tag")

list\_tags = ["user\_auth\_privilege", "upload files", "file\_system\_pack\_up", "make\_note", "import excel", "export excel"]

def callback(sender, app\_data, user\_data, ):

    print("Called on the main thread!")

    print(user\_data)

    if user\_data == list\_tags[1]:

        # TODO

        open\_selector()  # //日志 云存储

    elif user\_data == list\_tags[3]:

        #     文本编辑器， 同样保存即上传。

        start\_notepad()

    elif user\_data == list\_tags[2]:

        t1 = thread.Thread(open\_rust)

        t1.start()

        time.sleep(3)

        t1.kill()  # 调用CMD的话，好像程序就被杀死,此处只进行演示吧。

    #      这个文件整理当然可以做好，包所有整理文件搞好，列一个清单

    #      后面没有实现

    elif user\_data == list\_tags[4]:

        # "import excel" ,"export excel"

        import\_excel()

    elif user\_data == list\_tags[5]:

        output\_file = data\_utils.export\_xls()

        dpg.delete\_item("excel\_output")

        with dpg.window(label="file content", tag="excel\_output", width=800, height=400, modal=True, pos=(200, 200),

                        no\_move=True):

            dpg.add\_text(output\_file)

    elif user\_data == "add student":

        add\_stu()

    elif user\_data == "delete student":

        del\_stu()

    elif user\_data == "modify student info":

        modify\_stu()

def user\_auth(role\_id=1):

    for list\_tag in list\_tags:

        dpg.delete\_item(list\_tag)

    with dpg.window(label="user\_auth\_privilege", pos=(0, 180), no\_resize=True, no\_title\_bar=True, no\_move=True,

                    width=200, height=520, tag="user\_auth\_privilege"):

        if role\_id:

            auth\_tree = sql\_session.get\_auth(role\_id=role\_id)

            if auth\_tree is not None:

                for auth in auth\_tree:

                    length = len(str(auth))

                    temp = str(auth)[2:length - 3]

                    print(temp)

                    list\_tags.append(temp)

                    dpg.add\_button(label=str(temp), width=170, height=50, tag=str(temp), callback=callback,

                                   user\_data=str(temp))

        # 加载权限

        #         sql\_session.

        # //TODO 文件导出， 还有 数据可视化

        for i in range(1, 6):

            # lable , tag ,  user\_data , callback

            dpg.add\_button(label=list\_tags[i],  # PY 实现整理， Rust进行整理，

                           width=170, height=50, tag=list\_tags[i], callback=callback, user\_data=list\_tags[i])

###### UserOssFiles.py 上传下载或日记功能

import os

import dearpygui.dearpygui as dpg

from oss.oss import OSS as aliyun\_oss

# TODO 云存储文件渲染

def render\_files():

    files = aliyun\_oss.get\_files()

    dpg.delete\_item("files\_list\_window")

    dpg.delete\_item("file\_info")

    with dpg.window(label="files\_list\_window ", tag="files\_list\_window", pos=(950, 200), width=300, height=600,

                    no\_resize=True, no\_title\_bar=True, no\_move=True, no\_scrollbar=True, no\_scroll\_with\_mouse=True):

        dpg.add\_text(default\_value="oss cloud files", indent=50)

        with dpg.table(tag="file\_info", header\_row=False, row\_background=True, borders\_innerH=True, borders\_outerH=True,

                       borders\_innerV=True, borders\_outerV=True, width=200, height=80, ):

            dpg.add\_table\_column(label="files")

            dpg.add\_table\_column(label="btn1")

            dpg.add\_table\_column(label="btn2")

            for (file\_name, obj\_name) in files.items():

                with dpg.table\_row():

                    dpg.add\_text(default\_value=file\_name)

                    dpg.add\_button(label="download", callback=download, user\_data=file\_name + ":" + obj\_name)

                    if str(file\_name).lower().endswith(".txt"):

                        dpg.add\_button(label="check contents", callback=check\_info, user\_data=obj\_name)

def check\_info(sender, app\_data, user\_data):

    print(user\_data)

    #     TODO ,模态窗口

    (content, local\_path) = aliyun\_oss.pull\_file\_to\_local(file\_name=user\_data)

    print(content)

    dpg.delete\_item("file\_tag")

    with dpg.window(label="file content", tag="file\_tag", width=800, height=400, modal=True, pos=(200, 200),

                    no\_move=True):

        dpg.add\_text(content)

#     TODO 渲染用户上传文件,

def download(sender, app\_data, user\_data: str):

    names = user\_data.split(":")

    ab\_len = len(names)

    (content, local\_path) = aliyun\_oss.pull\_file\_to\_local(file\_name=names[1], is\_download=True)

    length = len(local\_path.split("/")[-1])

    new\_path = local\_path[0:ab\_len - length - 2:] + names[0]

    print(new\_path)

    try:

        with open(new\_path, "w", encoding="utf8") as fl:

            fl.write(content)

            fl.flush()

    except:

        print("不支持覆盖写入")

        os.rename(local\_path, new\_path)

        print("重命名")

    finally:

        os.remove(local\_path)

    print(length)

    #     弹窗

    dpg.delete\_item("content\_show\_win")

    with dpg.window(label="file content", tag="content\_show\_win", width=800, height=400, modal=True, pos=(200, 200),

                    no\_move=True):

        dpg.add\_text(f"the download file 's path is {new\_path}")

###### User\_window.py 渲染userwindow区域

import socket

import dearpygui.dearpygui as dpg

import redis

import userInfo.userAuthWindow as user\_auth

from sqlUtils.SqlSession import SqlSession as sql\_session

from  PropertiesUtils.ResourceBundle import ResourceBundle as resource\_bundle

properties = resource\_bundle.get\_bundle()

host = properties['redis']['host']

database = int(properties['redis']['database'])

def login():

    name = str(dpg.get\_value("name"))

    pwd = str(dpg.get\_value("pwd"))

    print(name)

    print(pwd)

    is\_checked = bool(dpg.get\_value("thirty\_days\_not\_log"))

    data = sql\_session.login(username=name, password=pwd)

    if data is None:

        print("密码或者账户错误")

        return

    if is\_checked:

        value = ""

        for col in data:

            len = str(col).\_\_len\_\_()

            value += str(col)[0: len]

            value += ","

        host\_ip = socket.gethostbyname(socket.gethostname())

        r = redis.Redis(host=host, port=6379, db=database)

        r.set(host\_ip, value)

    list\_tags = ["exit\_modal", "out\_btn", "texture\_tag", "logout\_modal", "user\_info\_window", "login\_btn", "login\_modal"]

    for list\_tag in list\_tags:

        dpg.delete\_item(list\_tag)

    render\_window(username=data[0], auth=data[1])

    user\_auth.user\_auth(role\_id=int(data[2]))

    dpg.configure\_item("login\_modal", show=False)

def logout():

    dpg.destroy\_context()

def render\_window(username="0", auth="0"):

    logged = None

    try:

        logged = int(username) != 0

    except:

        logged = True

    # 采用redis 进行数据

    # 是否30天免登录

    if logged is not None:

        host\_ip = socket.gethostbyname(socket.gethostname())

        print(host\_ip)

        # 保存信息

        r = redis.Redis(host='docker', port=6379, db=0)

        info = r.get(host\_ip)

        username = "passenger"

        auth = "passenger"  # 最大权限

        id = "0"

        if info is not None:

            print(info)

            info = str(info).split(sep=',')  # name

            # 0 1

            for i in range(3):

                print(info[i])

            username = info[0][2:]

            auth = info[1]

            print(auth)

            id = int(str(info[2]))  # role\_id

    # 加入用户信息window

    with dpg.window(label="user\_info\_window", pos=(0, 0), no\_resize=True, no\_title\_bar=True, no\_move=True,

                    width=200, height=180, tag="user\_info\_window"):

        # image

        width, height, channels, data = dpg.load\_image("./snowFlower.png")

        with dpg.texture\_registry(show=False):

            dpg.add\_static\_texture(width=width, height=height, default\_value=data, tag="texture\_tag")

        max\_size = 50

        scale\_factor = min(max\_size / width, max\_size / height)  # 计算缩放因子以保持宽高比

        new\_width = int(width \* scale\_factor)

        new\_height = int(height \* scale\_factor)

        dpg.add\_image("texture\_tag", width=new\_width, height=new\_height, indent=50)

        # info

        with dpg.table(header\_row=False, pos=(0, 0), tag="user\_info\_tag"):

            dpg.add\_table\_column()

            for i in range(2):

                with dpg.table\_row():

                    if i:

                        dpg.add\_text(f"user auth : {auth}")

                    else:

                        dpg.add\_text(f"username : {username}")

        with dpg.window(label="Delete Files", modal=True, show=False, tag="exit\_modal", no\_title\_bar=True):

            dpg.add\_text("This application will be shutdown.\nThis operation cannot be undone!")

            dpg.add\_separator()

            dpg.add\_checkbox(label="Don't ask me next time")

            with dpg.group(horizontal=True):

                dpg.add\_button(label="OK", width=75, callback=logout)

                dpg.add\_button(label="Cancel", width=75, callback=lambda: dpg.configure\_item("exit\_modal", show=False))

        dpg.add\_button(label="logout", tag="out\_btn", callback=lambda: dpg.configure\_item("exit\_modal", show=True))

        dpg.add\_button(label="login", tag="login\_btn")

        with dpg.popup(dpg.last\_item(), mousebutton=dpg.mvMouseButton\_Left, modal=True, tag="login\_modal"):

            dpg.add\_text("username :")

            dpg.add\_input\_text(tag="name")

            dpg.add\_text("password :")

            dpg.add\_input\_text(tag="pwd", password=True)

            dpg.add\_checkbox(tag="thirty\_days\_not\_log")

            dpg.add\_button(label="Login In", callback=login)

##### main.py 主程序

import dearpygui.dearpygui as dpg

from  oss.oss import OSS as aliyun\_oss

import system.search\_bar as search

import system.student\_info as stu\_info

import system.welcome as welcome

import userInfo.UserWindow as userwin

import userInfo.userAuthWindow as userauth

if \_\_name\_\_ == '\_\_main\_\_':

    dpg.create\_context()

    dpg.configure\_app(manual\_callback\_management=True)

    # 加载字体

    with dpg.font\_registry():

        default\_font = dpg.add\_font("./resource/ChillDuanSans\_Bold.otf", 15)

        title\_font = dpg.add\_font("./resource/XuandongKaishu.otf", 30, )

    dpg.bind\_font(default\_font)

    # 装载容器

    with dpg.window(label="user\_window", pos=(0, 0), width=200, height=700, no\_resize=True, no\_title\_bar=True,

                    no\_move=True, no\_bring\_to\_front\_on\_focus=True):

        # user Info windows

        userwin.render\_window()

        # user auth tree window

        userauth.user\_auth()

    # // 首页

    with dpg.window(label="stu\_info", pos=(200, 0), no\_resize=True, no\_title\_bar=True, no\_move=True, width=1200,

                    height=700, no\_bring\_to\_front\_on\_focus=True):

        title = welcome.welcome\_window()

        dpg.bind\_item\_font(title, title\_font)

        # search bar

        search.search\_render()

        # stu info table 这个应该是一个gloabl table

        stu\_info.stu\_render()

        # other data import operations

    #     TODO : oss -files

        userauth.render\_files()

    dpg.create\_viewport(title='Student Management System', width=1200, height=700,

                        large\_icon="./resource/winter flower.ico",

                        resizable=False)

    dpg.setup\_dearpygui()

    dpg.show\_viewport()

    while dpg.is\_dearpygui\_running():

        jobs = dpg.get\_callback\_queue()  # retrieves and clears queue

        dpg.run\_callbacks(jobs)

        dpg.render\_dearpygui\_frame()

    dpg.start\_dearpygui()

    dpg.destroy\_context()

##### 文件整理

use chrono::{Datelike, Utc};

use flate2::Compression;

use flate2::write::GzEncoder;

use num::{Float, range};

use redis::{Commands, Connection};

use std::collections::hash\_map::DefaultHasher;

use std::env;

use std::fs::{self, File};

use std::hash::{Hash, Hasher};

use std::ops::Add;

use std::os::windows::fs::MetadataExt;

use std::path::{Path, PathBuf};

use std::process::Command;

use std::string::ToString;

use std::thread::sleep;

use std::time::Duration;

fn main() {

    clean\_start();

}

fn clean\_start() {

    // 获取当前工作目录

    let str = env::current\_dir().unwrap().display().to\_string();

    let path = Path::new(&str);

    let root\_path = get\_root\_path(&path);

    let root = Path::new(&root\_path);

    let host\_ip = get\_local\_ip().unwrap();

    println!("{}", root.display().to\_string());

    let client = redis::Client::open("redis://docker").unwrap();

    let mut con = client.get\_connection().unwrap();

    //对该工作目录所有文件进行整理。

    clean\_up(root, &host\_ip, &mut con);

//     开启redis 监听文件，

    count\_loop(host\_ip);

}

fn get\_root\_path(path: &Path) -> String {

    if path.is\_absolute() {

        match path.parent() {

            None => {

                return path.display().to\_string();

            }

            Some(parent) => {

                return get\_root\_path(parent);

            }

        };

    }

    return "/".to\_string();

}

fn get\_sha256\_id(pathes: &Vec<&str>) -> String {

    let mut text = String::new();

    for path in pathes {

        text.push\_str(path)

    }

    println!("{}", text);

    let mut hasher = DefaultHasher::new();

    text.hash(&mut hasher);

    let hash\_result = hasher.finish();

    let hash\_hex = hash\_result.to\_be();

    println!("SHA256 hash of '{}': {}", text, hash\_hex);

    hash\_result.to\_string()

}

fn count\_loop(host\_dirs: String) {

    let host\_ip = get\_local\_ip().unwrap();

    let client = redis::Client::open("redis://docker").unwrap();

    let mut con = client.get\_connection().unwrap();

    let mut count\_times = 25;

    loop {

        let dirs: String = con.get(&host\_dirs).unwrap();

        for dir in dirs.split(",") {

            let key = format!("{host\_ip}:{dir}");

            let files: String = con.get(&key).unwrap();

            let files = files.split(",");

            if files.count() < 10 {

                sleep(Duration::from\_secs(5));

                break;

            }

            //

            print!("放入 host：dirkey");

            let files: String = con.getset(&key, "").unwrap(); //读写分离

            let mut pathes: Vec<&str> = vec![];

            for (index, file) in files.split(",").enumerate() {

                pathes[index] = file;

            }

            pathes.iter().for\_each(|f| println!("{}", f));

            let sha256 = get\_sha256\_id(&pathes);

            tar\_multi\_files(&pathes, sha256);

            sleep(Duration::from\_secs(5));

        }

        count\_times -= 1;

        if count\_times < 0 {

            break;

        }

    }

}

fn clean\_up(root: &Path, host\_ip: &str, con: &mut Connection) {

    let host\_dirs = String::from(host\_ip).add(":directories");

    let rd = match fs::read\_dir(root) {

        Ok(rd) => { rd }

        Err(\_) => { return; }

    };

    for entry in rd {

        let entry = entry.unwrap();

        // entry.metadata()TODO redis 存储磁盘记录

        let path = entry.path(); //path

        let path = path.as\_path();

        // 对路径进行操作，例如打印文件名

        if path.is\_dir() {

            con.set(&host\_dirs, &path.display().to\_string()).unwrap\_or(0);

            clean\_up(&Path::new(&String::from(path.display().to\_string())), host\_ip, con);

        } else if path.is\_file() {

            println!("{}", path.display().to\_string());

            if is\_sys\_file(path) {

                continue;

            } else if can\_be\_deleted(path) {

                fs::remove\_file(path).expect("TODO: panic message");

            } else {//但不可删除

                if path.is\_absolute() {

                    match path.metadata() {

                        Ok(meta) => {

                            let bytes = Float::floor(meta.file\_size() as f64 / 1024f64) as i32;

                            println!("{}", bytes);

                            if bytes > 16 {

                                let pathes = vec![path.to\_str().unwrap()];

                                let sha256 = get\_sha256\_id(&pathes);

                                println!("{}", sha256);

                                let source = tar\_multi\_files(&pathes, sha256);

                                let mut parent = Path::new(&source).parent().unwrap().display().to\_string();

                                println!("{}", parent);

                                //生成链接

                                let mut name = String::new();

                                for x in pathes {

                                    name.push\_str(x);

                                    if name.len() > 10 {

                                        break;

                                    }

                                }

                                parent.push\_str(&name);

                                parent.push\_str(".link");

                                println!("{}", name);

                                let dest = PathBuf::from(parent);

                                let source = Path::new(&source);

                                create\_shortcut(&source, &dest);

                            } else if get\_gap\_days(path) { //如果最近访问时间超出2^4天数

                                let mut parent\_path = path.parent().unwrap().display().to\_string();

                                let key = parent\_path.clone();

                                // hostip:directories  : dir, dir,,

                                let mut len = con.strlen(&host\_dirs).unwrap\_or(0);

                                if len != 0 {

                                    parent\_path = String::from(",").add(&parent\_path);

                                } else {

                                    len = 1;

                                }

                                con.setrange(&host\_dirs, len - 1, parent\_path).unwrap\_or(0);

                                let file\_key = String::from(host\_ip).add(&key);

                                println!("{}", file\_key);

                                let mut child\_path = path.display().to\_string();

                                let mut len = con.strlen(&file\_key).unwrap\_or(0);

                                if len != 0 {

                                    child\_path = String::from(",").add(&child\_path);

                                } else {

                                    len = 1;

                                }

                                //

                                con.setrange(&file\_key, len - 1, child\_path).unwrap\_or(0);

                            }

                        }

                        Err(\_) => {}

                    }

                }

            }

        }

    }

}

//这里获取文件上一次访问的ms数，有bug，这个java ， python都测试了，

//所以最终采用dir {path} /t:c 命令行读取上一次访问时间

fn get\_gap\_days(path: &Path) -> bool {

    // dir stu.xlsx /t:c

    // let path = PathBuf::from(r"C:\Users\33769\Desktop\stu.xlsx");

    let path = path.display().to\_string();

    let mut date = Option::Some(String::from("NONE"));

    let output = Command::new("cmd")

        .arg("/c")

        .arg(format!("dir {path} /t:c"))

        .output()

        .unwrap\_or\_else(|e| {

            panic!("Failed to execute process: {}", e);

        });

    let stdout = String::from\_utf8\_lossy(&output.stdout);

    let out = stdout.to\_string();

    for (index, value) in out.lines().enumerate() {

        if index == 5 {

            date = Option::Some(String::from(&value[0..10]));

        }

    }

    if date.is\_none() {

        false;

    }

    let date = match date {

        None => { "2024/01/07".to\_string() }

        Some(time) => {

            println!("{}", time);

            if time.eq("NONE") {

                return false;

            }

            time

        }

    };

    let last\_year = (&date[0..4]).parse::<isize>().unwrap();

    let last\_month = (&date[5..7]).parse::<isize>().unwrap();

    let last\_day = (&date[8..10]).parse::<isize>().unwrap();

    let now\_time = Utc::now();

    let year: isize = now\_time.year() as isize;

    let month: isize = now\_time.month() as isize;

    let day: isize = now\_time.day() as isize;

    if year - last\_year > 2 {

        return true;

    }

    if last\_month > month {

        return true;

    }

    let month\_gap = match year - last\_year {

        1 => { 12 - last\_month + month }

        0 => { month - last\_month }

        \_ => { 0 }

    };

    if month\_gap > 0 {

        return true;

    }

    day - last\_day > 15

}

fn tar\_multi\_files(pathes: &Vec<&str>, sha256: String) -> String {

    let mut parent\_path = String::from("");

    if pathes.len() > 0 {

        let child\_path = PathBuf::from(pathes[0]);

        parent\_path = child\_path

            .parent()

            .unwrap().to\_str()

            .unwrap().to\_string();

        parent\_path.push\_str("\\");

        println!("{}", &parent\_path);

    }

    parent\_path.push\_str(&sha256);

    let mut sha256 = parent\_path;

    sha256.push\_str(".tar.gz");

    let tar\_gz = File::create(&sha256).unwrap();

    let enc = GzEncoder::new(tar\_gz, Compression::default());

    let mut tar = tar::Builder::new(enc);

    for a in pathes.iter() {

        let mut file = fs::File::open(a).unwrap();

        let path\_buf = PathBuf::from(a);

        let name = path\_buf.file\_name().unwrap().to\_str().unwrap();

        tar.append\_file(name, &mut file).ok();

    }

    tar.finish().ok();

    sha256

}

fn is\_sys\_file(path: &Path) -> bool {

    let sys\_types = vec![".dll", ".sys", ".ini", ".cfg", ".ttf", ".otf", ".json",

                         ".xml", ".db", ".sqllite", ".sql", ".dmp"];

    for x in sys\_types {

        let ret = path.ends\_with(x);

        if ret {

            return true;

        }

    }

    return false;

}

fn can\_be\_deleted(path: &Path) -> bool {

    let del\_types = vec![".tmp ", ".temp", ".thum", ".bak", ".msi", ".log"

                         , ".err", ".tmp", ".jpg", ".png"];

    for x in del\_types {

        let ret = path.ends\_with(x);

        if ret {

            return true;

        }

    }

    return false;

}

fn create\_shortcut(source\_file: &Path, shortcut\_file: &Path) {

    let source\_path = source\_file.display().to\_string();

    let shortcut\_path = shortcut\_file.display().to\_string();

    // 执行 CMD 命令

    let output = Command::new("cmd")

        .arg("/c")  // 使用 /c 参数来执行后续的命令

        .arg(format!("mklink {shortcut\_path} {source\_path}")) // 执行 dir 命令，列出当前目录下的文件和文件夹

        .output()

        .unwrap\_or\_else(|e| {

            panic!("Failed to execute process: {}", e);

        });

}

fn get\_local\_ip() -> Result<String, Box<dyn std::error::Error>> {

    // 使用网络库来获取本机的IP地址

    let socket = std::net::UdpSocket::bind("0.0.0.0:0").unwrap();

    socket.connect("8.8.8.8:80").unwrap();

    let local\_ip = socket.local\_addr().unwrap().ip();

    Ok(local\_ip.to\_string())

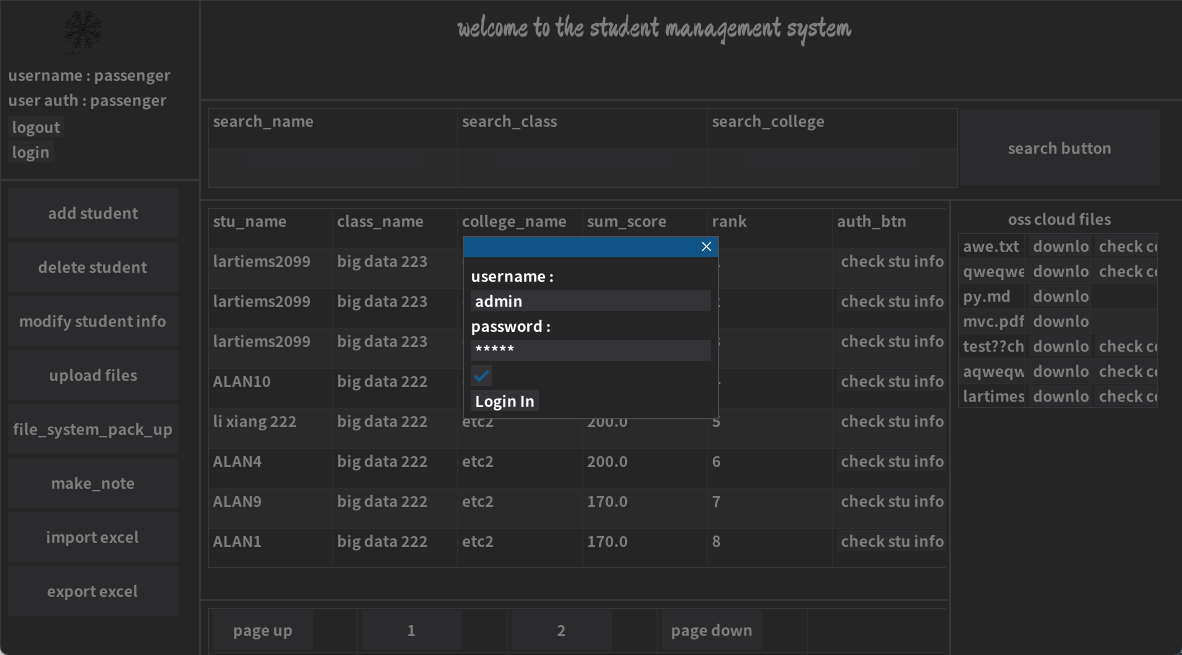
}

## 实验结果及分析

#### 运行截图

##### User模块

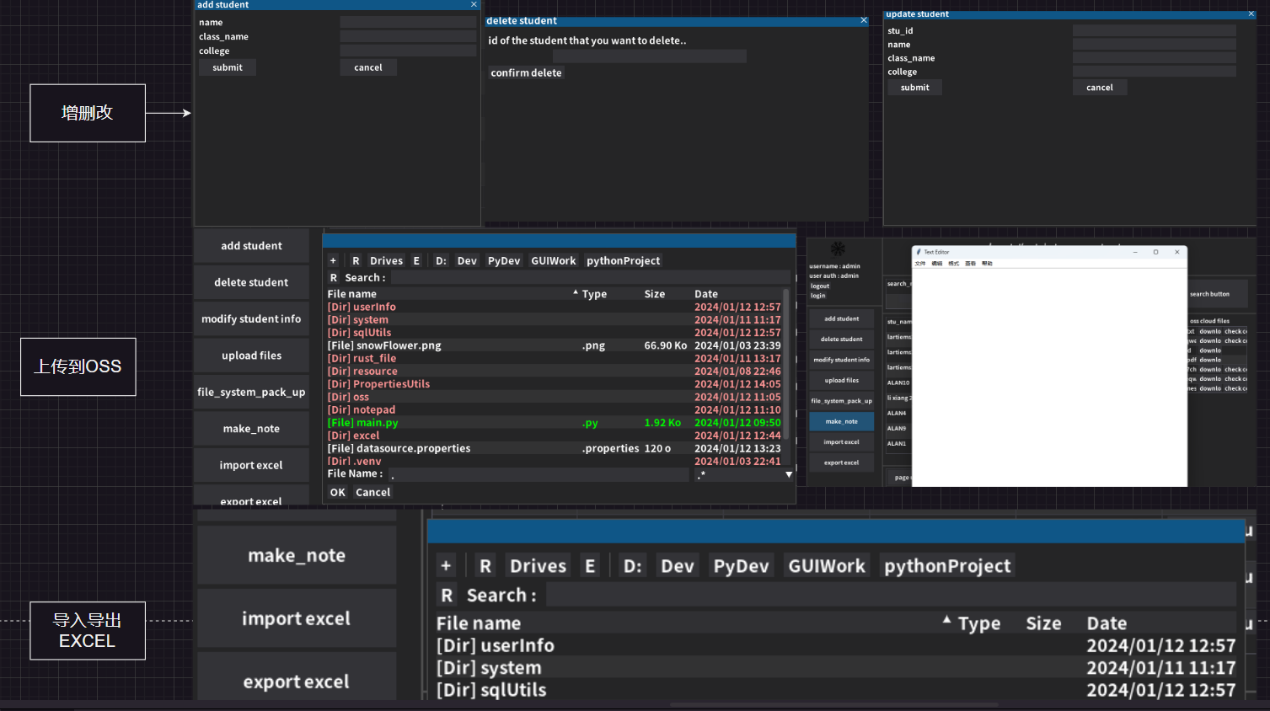
登录模块：匹配数据库，并且加载权限，重新渲染权限菜单，并且以IP+user 作为key作为缓存

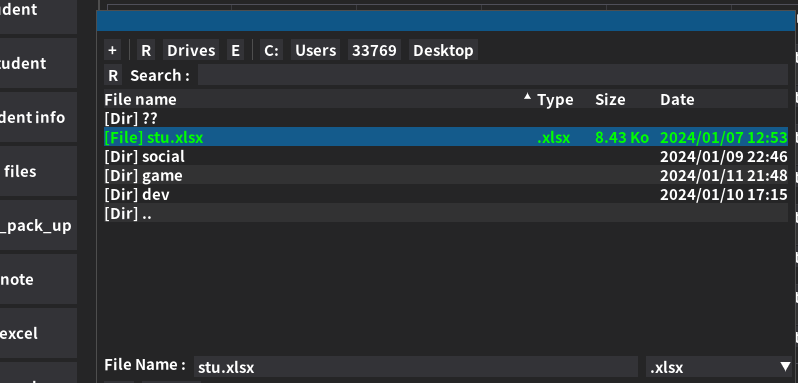


##### 权限菜单模块

权限菜单模块，以模态窗口或者新窗口实现响应。RBAC User<-->Role<-->Auth , 不同用户不同权限

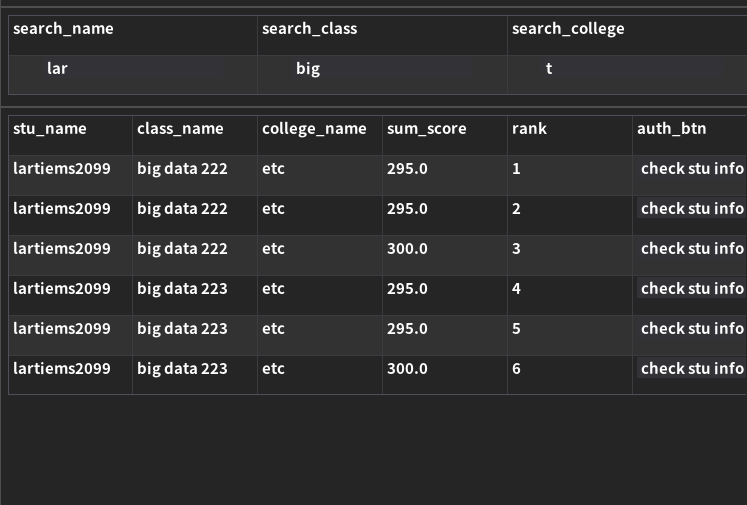
目前支持的权限： 增删改查，上传下载文件，日记等基础文本云存储，导入导出学生信息Excel。

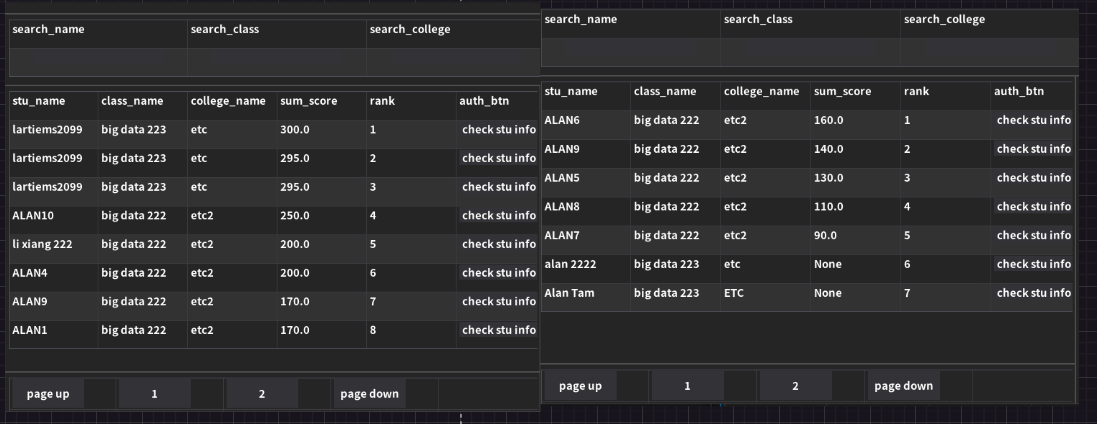


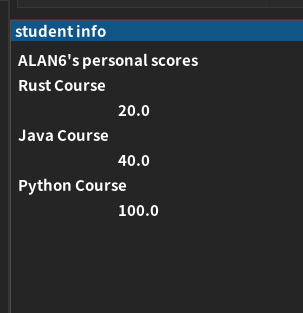


##### 学生信息模块

默认降序排序，分页查询与基于学生名字、班级、学院模糊查询，以及查询细信息功能都完备。

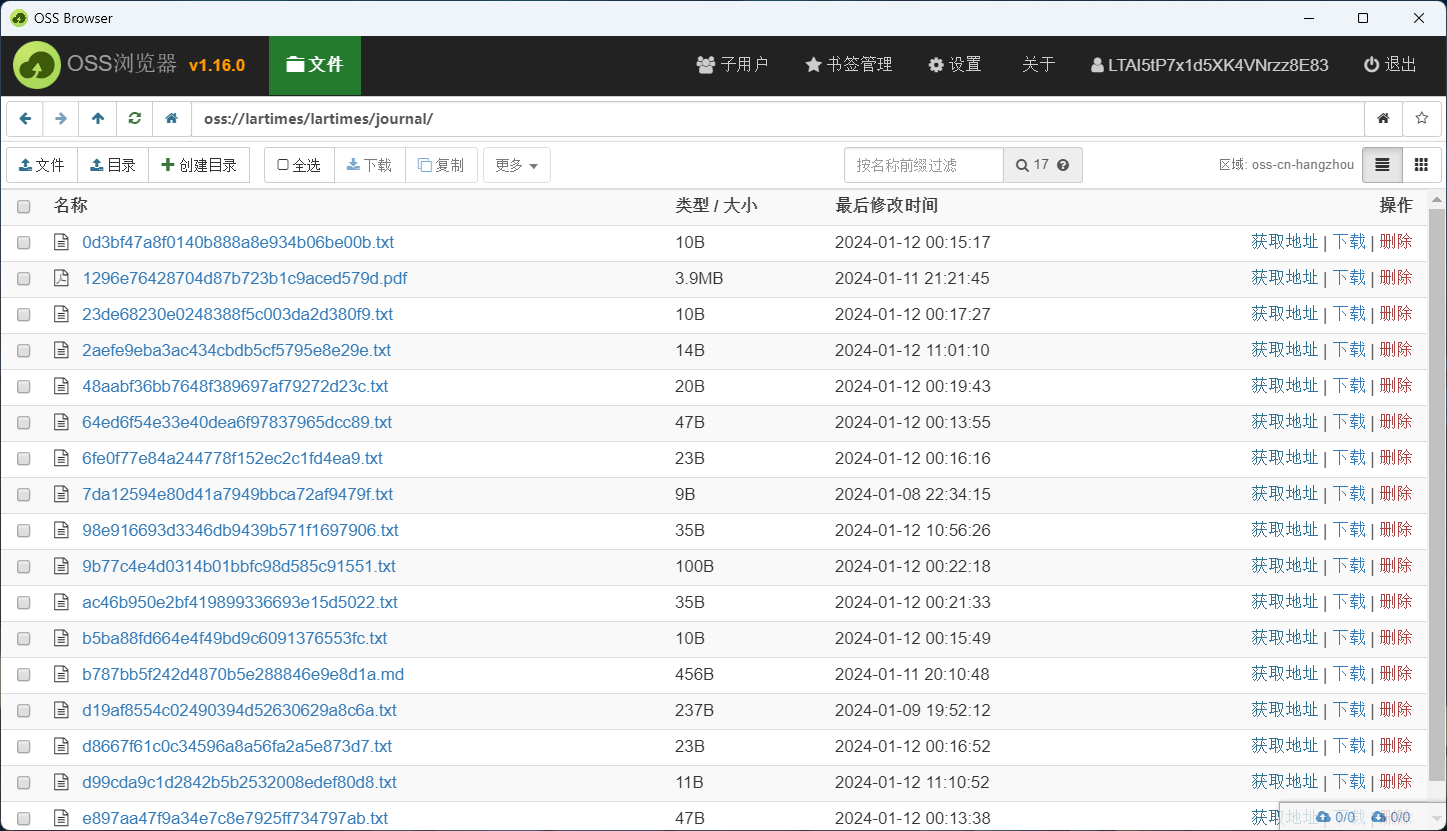
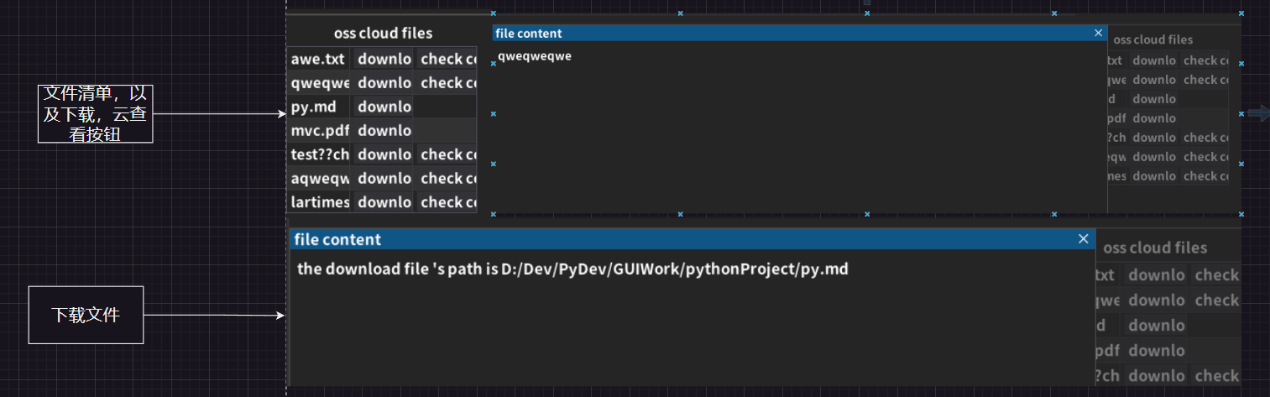




 模糊查询、分页查询、查看详细信息

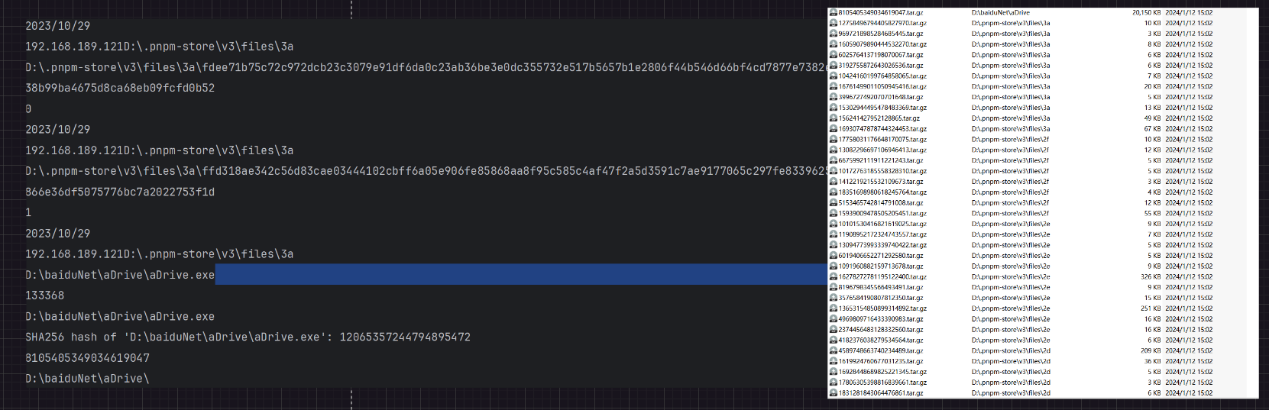
##### OSS云存储模块

采用用户 name+ uuid+sha256作为上传文件唯一标识



##### 文件整理模块

这一块，本来打算导出rust供python直接调用，但是需要导出所有函数，结构体。所以采用CMD硬编码调用，但是此功能不完善，没有图形化供使用者选择删除。但是CMD调用EXE文件（此处应该不由程序线程控制了，默认获取该工作目录顶级磁盘进行清理），尽管采用子线程调用，强制kill该线程也停止不了其运行。此处将控制台log作为展示。



#### 遇到的问题解决方案

##### GUI

主体GUI采用Dearpygui ，没有什么原因，主要因为名字。官方文档没有查到对已经渲染的window等组件数据进行修改渲染，最后只能采用下策，删除指定window以及其所有tag，也就是重新加载这个window对象。

//例子

if not dpg.get\_value("tmp4"):

        dpg.delete\_item("tmp4")

    print("=======================")

    with dpg.window(label="update student", tag="tmp4", width=800, height=400, modal=True, pos=(200, 200),

                    no\_move=True):

        list\_stu = ["stu\_id", "name", "class\_name", "college", ]

        for tag in list\_stu:

            dpg.delete\_item(tag)

##### Excel导出数据排版的转变

由于表的设计，返回所有信息的话，返回字段是这样的：id ， name ， college ， course\_name , course\_score。我们需要转变其为(id , name , college , class .. , score1,score2 , score3 ....)

 project\_path = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))  # 获取当前工作顶级目录

        print(project\_path)

        output\_file = os.path.join(project\_path, uuid.uuid4().hex + 'stu.xlsx')  # uuid + stu.xlsx

        print(output\_file)

        ret = sql\_session.export\_all()

        length = len(ret)

        course\_len = len(DataUtils.course\_list)

        java\_list = []

        py\_list = []

        rs\_list = []

        for i in range(length):

            course\_name = str(ret[i][4])

            if course\_name == DataUtils.course\_list[0]:

                java\_list.append(ret[i][3])

            elif course\_name == DataUtils.course\_list[1]:

                py\_list.append(ret[i][3])

            elif course\_name == DataUtils.course\_list[2]:

                rs\_list.append(ret[i][3])

        rows = length / course\_len  # row , result , 读取course ，

        data = {}  # export -- 》 {column , []}

        for column\_name, index in DataUtils.dict\_column.items():

            #             每一列进行填充 0 1 2  length

            tmp = []

            for i in range(int(rows)):  # 0 1 2 row

                if index < 3:

                    print(ret[(i - 1) \* course\_len][index])

                    tmp.append(ret[i][index])

                    continue

                # 读取课程score

                break

            if tmp:

                data[column\_name] = tmp

        data[DataUtils.course\_list[0]] = java\_list

        data[DataUtils.course\_list[1]] = py\_list

        data[DataUtils.course\_list[2]] = rs\_list

        print(output\_file)

        for datum in data.values():

            print(datum)

        df = pd.DataFrame(data)

        df.to\_excel(output\_file, index=False)

        return  output\_file

##### OSS云存储如何实现唯一标识，与任意文件上传与拉取。 还有同时支持云查看与下载于一个方法，采用异常try-except

 @classmethod

    def pull\_file\_to\_local(cls, file\_name: str, username: str = "lartimes", is\_download: bool = False, ):

        bucket = oss2.Bucket(OSS.auth, OSS.endpoint, OSS.username)

        # file\_name TODO 根据file——name 获取OSS filePath （obj\_name）

        local\_path = OSS.cwd + "/" + uuid.uuid4().hex + "." + file\_name.split(".")[1]

        bucket.get\_object\_to\_file(file\_name, local\_path)

        content = ""

        try:

            with open(local\_path, "r", encoding="utf8") as fl:

                result = fl.readlines()

                for line in result:

                    content += line

        except:

            print("不持支该格式直接读入")

        print(content)

        if not is\_\_download:

            os.remove(local\_path)

        return (content, local\_path)

##### 文件整理模块获取文件metadata最近访问时间，Rust获取时间的ms数出现bug，已经过java ，py测试，ms相差甚大，最终采用硬编码调用CMD从响应数据获取日期。

 let output = Command::new("cmd")

        .arg("/c")

        .arg(format!("dir {path} /t:c"))

        .output()

        .unwrap\_or\_else(|e| {

            panic!("Failed to execute process: {}", e);

        });

    let stdout = String::from\_utf8\_lossy(&output.stdout);

    let out = stdout.to\_string();

    for (index, value) in out.lines().enumerate() {

        if index == 5 {

            date = Option::Some(String::from(&value[0..10]));

        }

}

//…….

 let last\_year = (&date[0..4]).parse::<isize>().unwrap();

    let last\_month = (&date[5..7]).parse::<isize>().unwrap();

    let last\_day = (&date[8..10]).parse::<isize>().unwrap();

    let now\_time = Utc::now();

    let year: isize = now\_time.year() as isize;

    let month: isize = now\_time.month() as isize;

    let day: isize = now\_time.day() as isize;

//….

# 结束语（心得体会）

加深了Python基础语法的理解与应用。同时体会到python这类语言的缺点。 ：与缩进 即是优势也是却失，这必然带来了大工程1的维护成本，大工程几乎没有python写的，python即使声称面向对象，但个人认为其适合快速开发，面向过程的类型。同时语法简单，学习成本低，适合初学者。

# 课设总结

熟悉了py的基础部分，体会到了其优缺点。大家从头设计共同实现提高了编程能力。培养了很大兴趣。实现这个管理系统，同时做了一些小工具，小工具可以视为我们做程序的初衷，更可能去将程序实践化，将其投入以及运用到现实生活中以带来便捷性。

# 参考文献（资料）

* [Dear PyGui’s Documentation — Dear PyGui documentation](https://dearpygui.readthedocs.io/en/latest/index.html) :
  + <https://dearpygui.readthedocs.io/en/latest/index.html>
* [如何为Python SDK配置访问凭证\_对象存储(OSS)-阿里云帮助中心 (aliyun.com)](https://help.aliyun.com/zh/oss/developer-reference/python-configuration-access-credentials?spm=a2c4g.11186623.0.i5)
  + <https://help.aliyun.com/zh/oss/developer-reference/python-configuration-access-credentials?spm=a2c4g.11186623.0.i5>
* [Python\_对象存储(OSS)-阿里云帮助中心 (aliyun.com)](https://help.aliyun.com/zh/oss/developer-reference/python/?spm=a2c4g.11186623.0.0.1f2d623afMEZKD)
  + <https://help.aliyun.com/zh/oss/developer-reference/python/?spm=a2c4g.11186623.0.0.1f2d623afMEZKD>
* [IO tools (text, CSV, HDF5, …) — pandas 2.1.4 documentation (pydata.org)](https://pandas.pydata.org/pandas-docs/stable/user_guide/io.html#io-excel)
  + https://pandas.pydata.org/pandas-docs/stable/user\_guide/io.html#io-excel