《软件技术基础课程设计》

实验项目：实验 2-2

院　　系：计算机科学与技术系

学　　期：2021 年第 1 学期

班　　级：19 阳明 1 班

学　　号：196002605　196002614

姓　　名：白炳松　　黄睿

指导教师：李纲

日期：2021 年 12 月 29 日

目录

分工安排

白炳松（50%）：播放器主窗体的布局与逻辑功能、笔记功能的实现、关键词生成显示。报告编写部分：概述、系统需求分析、类设计、行为模型设计、原型设计、系统框架设计、播放器模块实现、便笺模块实现、总结

黄睿（50%）：数据库连接、数据库建立、数据表创建、用例分析、语音识别与字幕添加模块、数据模型设计、数据管理模块、个人账户管理与信息修改模块、总结、参考文献

概述

1.1 系统介绍

建立一个多功能本地视频播放系统，主要由四大部分组成，分别是视频播放器的主窗体、富文本便笺、显示账户信息和统计数据的管理界面，以及 openGauss 数据库。

用户可以完成注册登录操作访问数据，并且通过获取激活码，可以得到 VIP 权限；对管理员也设置了权限，区分了不同的使用对象。

在登录后，导入本地视频进行正常播放，同时可以记录笔记，插入截图和图片进行记录，并且可以记录播放视频与笔记的数据等，所有数据都通过 JDBC 连接 openGauss 数据库进行操作。

1.2 运行环境及其开发工具

该系统在 Windows10 操作系统下进行开发，软件开发环境为 Python3.8，开发工具为 PyCharm，数据库为 openGauss。

系统需求分析

2.1 系统主要需求

该系统的主要任务有以下几点：

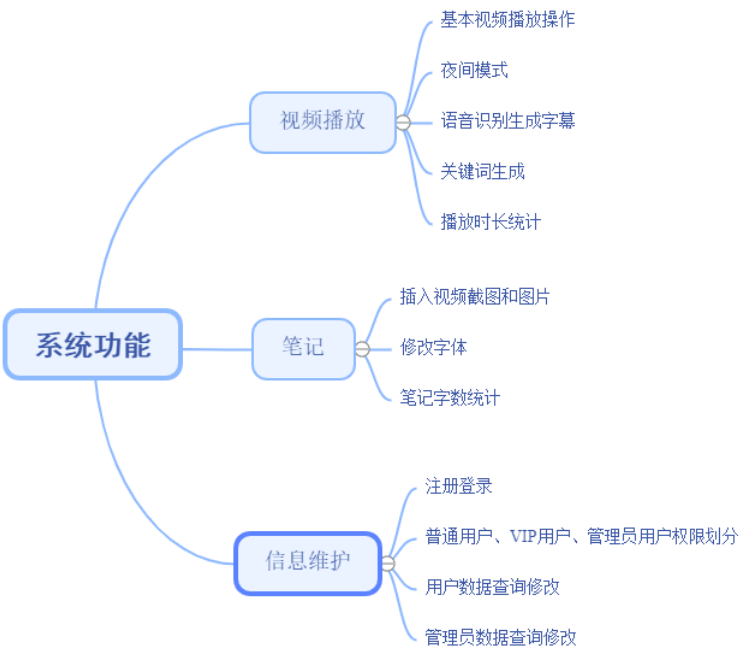
（1）用户注册登录，添加普通用户、VIP 用户以及管理员用户，分别给予不同权限。

（2）本地视频播放器，包括播放暂停、控制进度、控制音量、开启全屏等。

（3）视频同步笔记，可以导入视频截图和外部图片，修改笔记字体等。

（4）视频播放和笔记数据统计，以及账户信息管理与显示。

（5）语音识别生成视频字幕和生成字幕关键词



2.2 系统次要需求

除以上主要需求部分，该系统还应考虑以下性能的实现：

（1）易用性：可视化界面需要人性化的设计，更高的可操作性，降低用户的学习成本，做到即使没有说明书，一般的用户上手后也能进行操作，用户体验良好。

（2）可维护性：系统的可维护性是指当需要对系统的某个模块进行修改时，

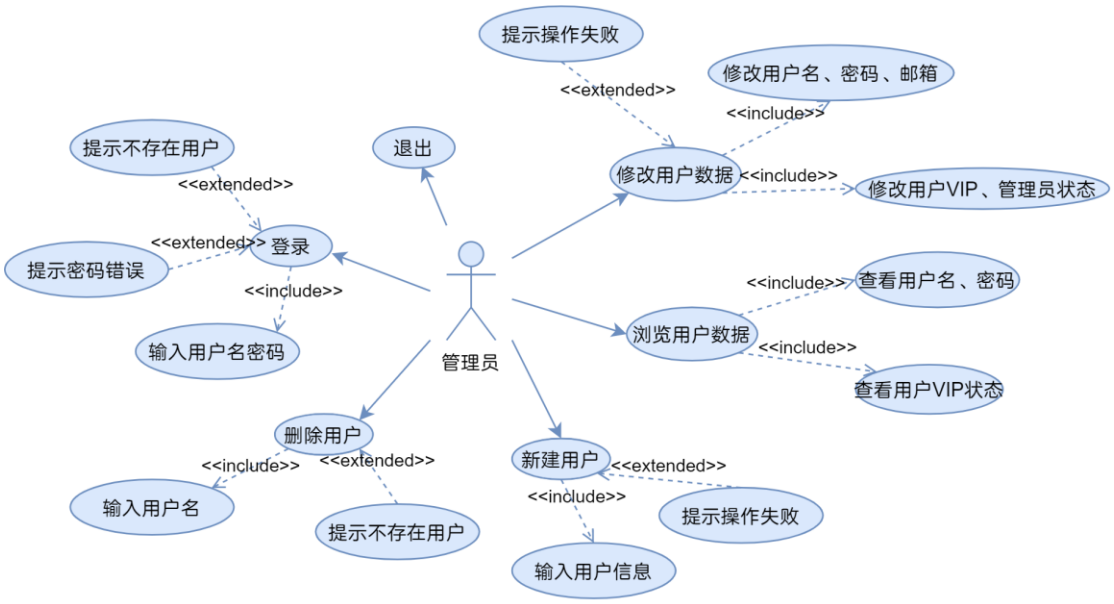
对其他模块的影响和修改的代价。

（3）有效性：系统的有效性是指系统完成计划的任务以及达到预期结果的程度。在该系统中，有效性主要指系统完成账户的注册登录，以及视频播放、笔记等功能的正常使用

（4）时效性：系统的时效性指的是系统保持更新，在功能、页面设计甚至是底层架构等方面都保持和现阶段技术现状和现实需求的紧密结合。针对我们这个系统具体来说，就是保持更新，和用户日益增长的需求保持高度一致，并且能够实现自我创新，创造用户新需求。

用例分析

软件的用例分析，参与者有普通用户、管理员用户。



3.1 管理员的用例分析

3.1.1 登录用例说明

用例名称	登录
参与者	管理员
用例描述	登陆账号
触发事件	点击登录
前置条件	已注册

事件流	基流	1.输入账号和密码 2.提交登录
	分支流	无
	代替流	1.输入账号错误，提示账号未注册 2.输入密码错误，提示密码错误 3.返回登陆界面
结论		登录完成，用例结束
后置条件		显示视频播放界面

3.1.2 退出用例说明

用例名称		退出
参与者		管理员
用例描述		退出账号
触发事件		点击退出账号
前置条件		已登录
事件流	基流	1、退出程序 2、删除登录信息文件
	分支流	无
	代替流	无
结论		退出登录，用例结束
后置条件		无

3.1.3 浏览用户数据说明

用例名称		浏览用户数据
参与者		管理员
用例描述		查看用户信息
触发事件		点击账户管理
前置条件		已登录
事件流	基流	1、从数据库读取数据 2、显示数据
	分支流	无
	代替流	提示无法读取登录信息
结论		显示完毕，用例结束
后置条件		无

3.1.4 修改用户数据说明

用例名称	修改用户数据
参与者	管理员
用例描述	修改用户信息数据
触发事件	1、点击账户管理

		2、点击修改按钮
前置条件		已登录
事件流	基流	1、从文本框读取数据 2、判断数据合法性 3、写入数据库
	分支流	无
	代替流	提示无法读取登录信息
结论		修改完毕，用例结束
后置条件		无

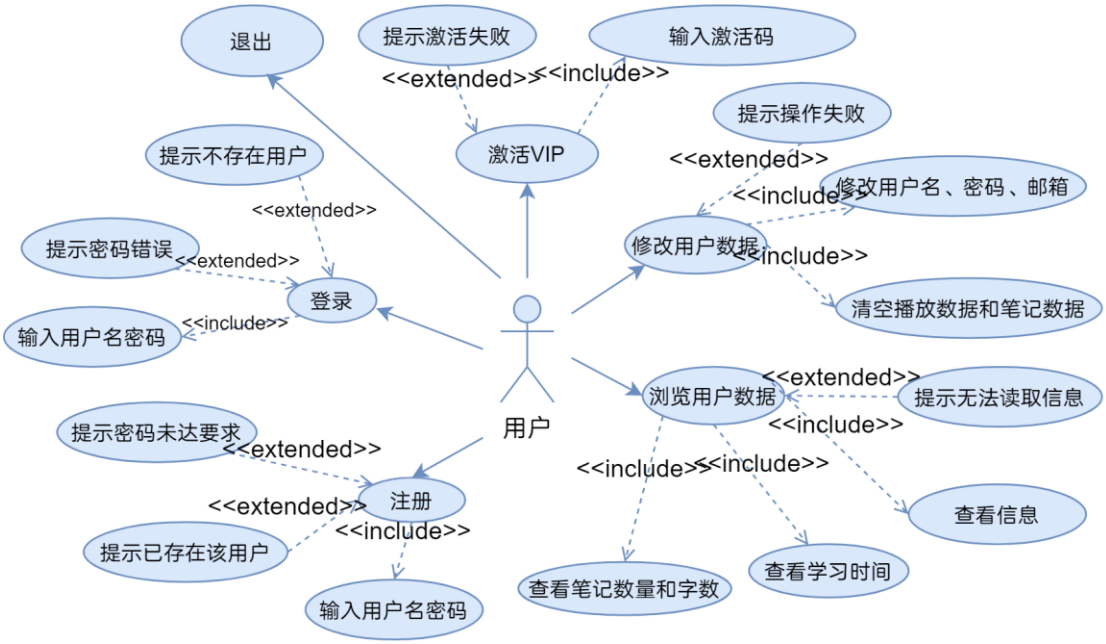
3.1.5 新建用户用例

用例名称		新建用户
参与者		管理员
用例描述		注册
触发事件		1、输入新账号和密码及其他信息 2、点击注册按钮
前置条件		无
事件流	基流	1、从文本框读取数据 2、判断数据合法性 3、写入数据库
	分支流	无
	代替流	提示已经存在该用户
结论		注册完毕，用例结束
后置条件		登录软件，进入视频播放

3.1.6 删除用户用例

用例名称		删除用户
参与者		管理员
用例描述		删除用户
触发事件		1、输入要删除的账号名 2、点击删除
前置条件		无
事件流	基流	1、从文本框读取数据 2、判断数据合法性 3、删除数据库中该条记录
	分支流	无
	代替流	提示不存在该用户
结论		删除完毕，用例结束
后置条件		无

3.2 用户用例分析



3.2.1 登录用例说明

用例名称		登录
参与者		普通用户
用例描述		登陆账号
触发事件		点击登录
前置条件		已注册
事件流	基流	1.输入账号和密码 2.提交登录
	分支流	无
	代替流	1.输入账号错误，提示账号未注册 2.输入密码错误，提示密码错误 3.返回登陆界面
结论		登录完成，用例结束
后置条件		显示视频播放界面

3.2.2 退出用例说明

用例名称		退出
参与者		普通用户
用例描述		退出账号
触发事件		点击退出账号
前置条件		已登录
事件流	基流	1、退出程序 2、删除登录信息文件

	分支流	无
	代替流	无
结论		退出登录，用例结束
后置条件		无

3.2.3 浏览用户数据用例说明

用例名称		浏览用户数据
参与者		普通用户
用例描述		查看用户信息
触发事件		点击账户管理
前置条件		已登录
事件流	基流	1、从数据库读取数据 2、显示数据
	分支流	无
	代替流	提示无法读取登录信息
结论		显示完毕，用例结束
后置条件		无

3.2.4 修改用户信息说明

用例名称		修改用户信息
参与者		普通用户
用例描述		修改用户信息
触发事件		1、点击账户管理 2、点击修改按钮
前置条件		已登录
事件流	基流	1、从文本框读取数据 2、判断数据合法性 3、写入数据库
	分支流	无
	代替流	1、提示无法读取登录信息 2、提示数据非法 3、提示数据库连接失败
结论		修改完毕，用例结束
后置条件		无

3.2.5 注册用例

用例名称	注册
参与者	普通用户
用例描述	注册

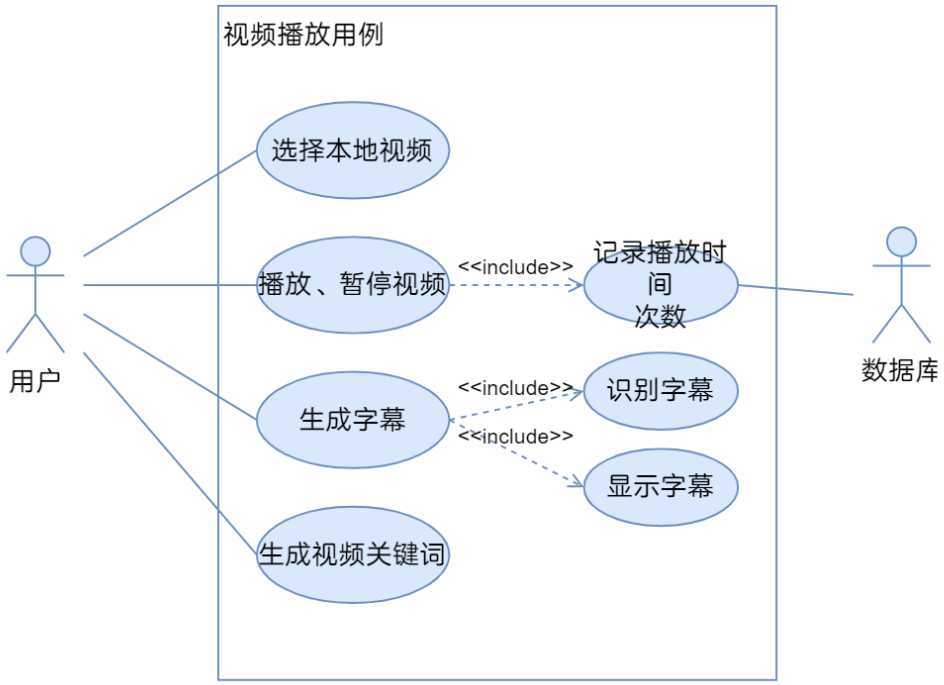
触发事件		1、输入新账号和密码及其他信息 2、点击注册按钮
前置条件		无
事件流	基流	1、从文本框读取数据 2、判断数据合法性 3、写入数据库
	分支流	无
	代替流	1、提示已经存在该用户 2、提示密码未达到要求
结论		注册完毕，用例结束
后置条件		登录软件，进入视频播放

3.2.6 激活 VIP

用例名称		激活 VIP
参与者		普通用户
用例描述		激活 VIP
触发事件		1、输入激活码 2、点击激活按钮
前置条件		无
事件流	基流	1、从文本框读取激活码 2、校验激活码 3、修改数据库 VIP 信息
	分支流	无
	代替流	提示激活码错误
结论		激活完毕，用例结束
后置条件		登录软件，进入视频播放

3.3 视频播放用例分析

3.3.1 用例图



3.3.2 视频播放暂停说明

用例名称		视频播放暂停
参与者		用户、数据库
用例描述		视频播放暂停
触发事件		1、点击播放按钮 2、点击暂停按钮
前置条件		1、用户已登录 2、已经打开视频
事件流	基流	1、从当前进度播放视频 2、记录已播放时间 3、同步进度条 4、暂停视频播放
	分支流	无
	代替流	无
结论		无
后置条件		视频播放完毕

3.3.3 生成字幕说明

用例名称	生成字幕
参与者	用户
用例描述	针对打开的视频进行语音识别生成字幕
触发事件	点击工具栏中的生成字幕选项
前置条件	1、用户已经正常登录

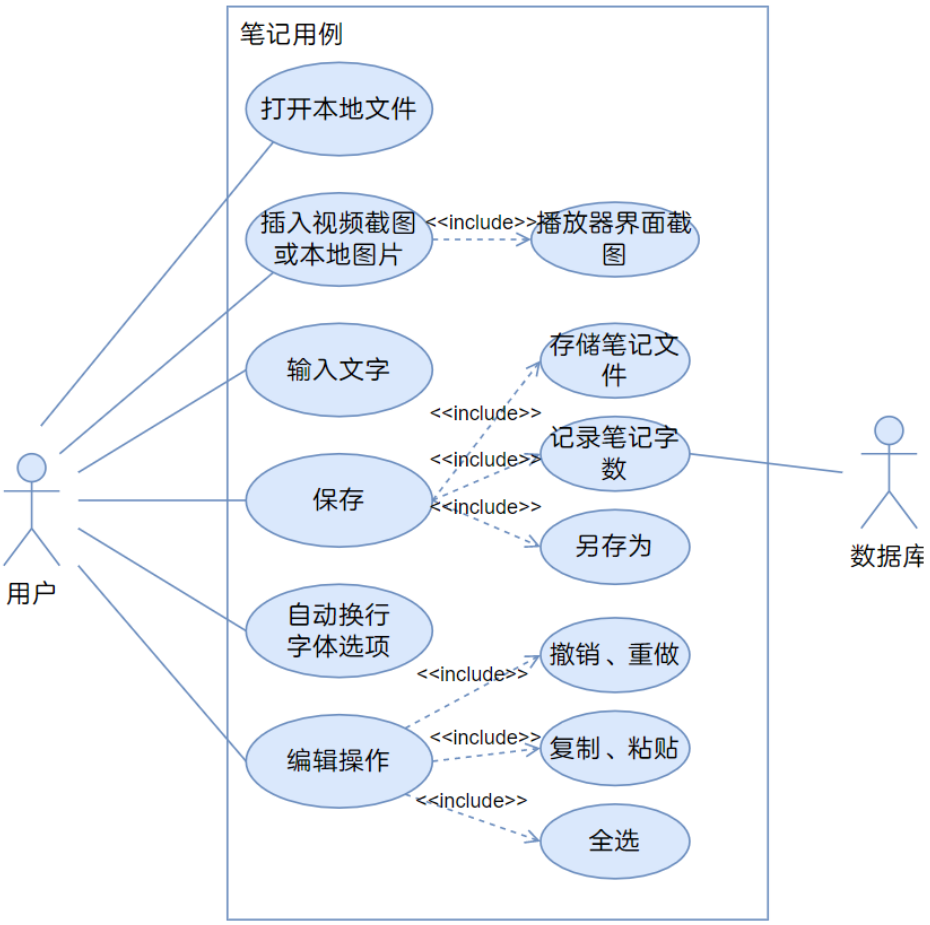
		2、用户已经正常打开视频
事件流	基流	1、提取已打开视频的音频 2、根据语音对音频进行分段 3、使用语音识别 API 转换成文字 4、生成字幕文件
	分支流	无
	代替流	1、音频提取失败 2、API 通信失败 3、识别失败
结论		生成完毕，用例结束
后置条件		生成和视频同名的 SRT 字幕文件

3.3.4 生成关键词用例说明

用例名称		生成关键词
参与者		用户
用例描述		生成关键词
触发事件		点击工具栏中的生成关键词选项
前置条件		1、用户已经正常登录 2、用户已经正常打开视频 3、已经生成字幕
事件流	基流	对语音识别的结果进行关键词提取
	分支流	无
	代替流	无
结论		生成关键词完毕，用例结束
后置条件		

3.4 笔记用例分析

3.4.1 插入图片用例说明



用例名称		插入图片
参与者		用户
用例描述		插入视频截图或本地图片
触发事件		1、在便笺窗口顶部工具栏点击对应选项
前置条件		无
事件流	基流	1、自动对视频播放窗口截屏 2、打开系统文件选择器选择图片 3、在便笺中插入图片
	代替流	无
结论		成功插入图片或截图
后置条件		成功插入图片或截图

3.4.2 保存用例说明

用例名称	保存
参与者	用户

用例描述		保存笔记
触发事件		1、点击工具栏中的生成关键词选项 2、关闭窗口，根据提示确定保存
前置条件		1、用户已经正常登录
事件流	基流	1、统计此次输入的字数 2、将字数存至数据库 3、保存笔记文件
	分支流	无
	代替流	1、不保存文件 2、不保存字数数据 3、另存为文件
结论		结束笔记用例
后置条件		关闭窗口

3.4.3 自动换行、字体选项用例说明

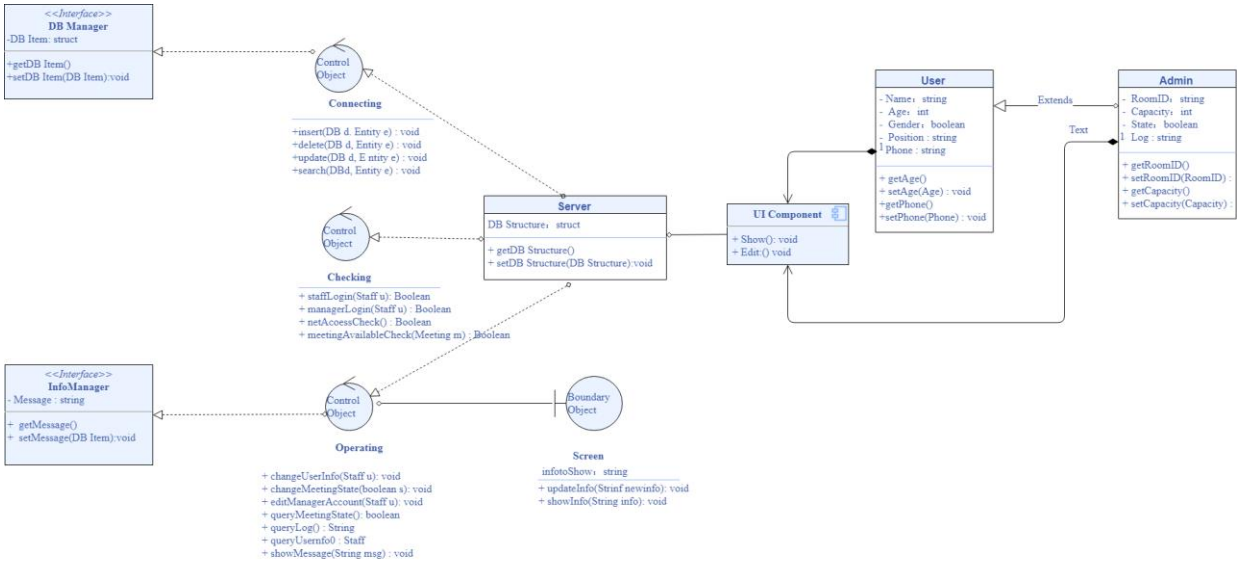
用例名称		自动换行、字体选项
参与者		用户
用例描述		设置页面显示自动换行、修改字体格式
触发事件		1、点击工具栏中的相应选项
前置条件		1、用户已经正常登录
事件流	基流	1、弹出字体选项窗口
	分支流	1、切换自动换行状态
	代替流	无
结论		结束修改自动换行、字体用例
后置条件		无

3.4.4 编辑操作用例说明

用例名称		编辑操作
参与者		用户
用例描述		复制、粘贴等编辑操作
触发事件		1、点击工具栏中的相应选项
前置条件		1、用户已经正常登录 2、已打开便笺窗口
事件流	基流	1、复制粘贴、撤销重做、全选、删除
	分支流	无
	代替流	无
结论		结束文字编辑用例
后置条件		无

类设计

类设计如下所示：

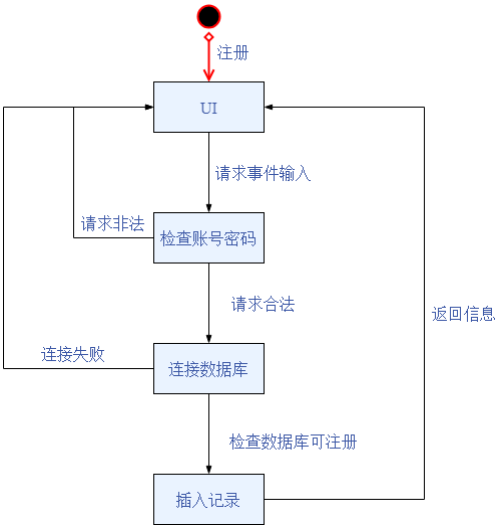


行为模型设计

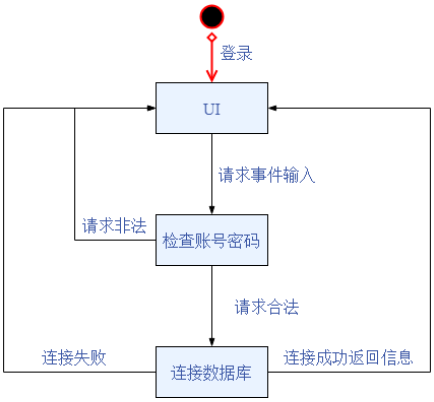
5. 1 关键业务的活动图

主要分为 3 种不同的状态图，其中 UI 是系统初始访问状态，用于提供操作界面和等待用户操作。进行后续操作时进行不同状态的转换。

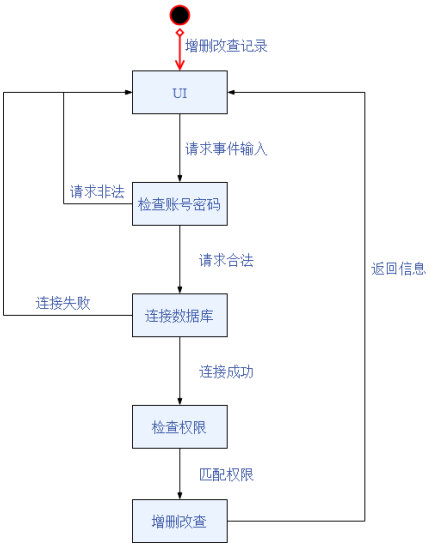
(1)注册：



(2)登录：

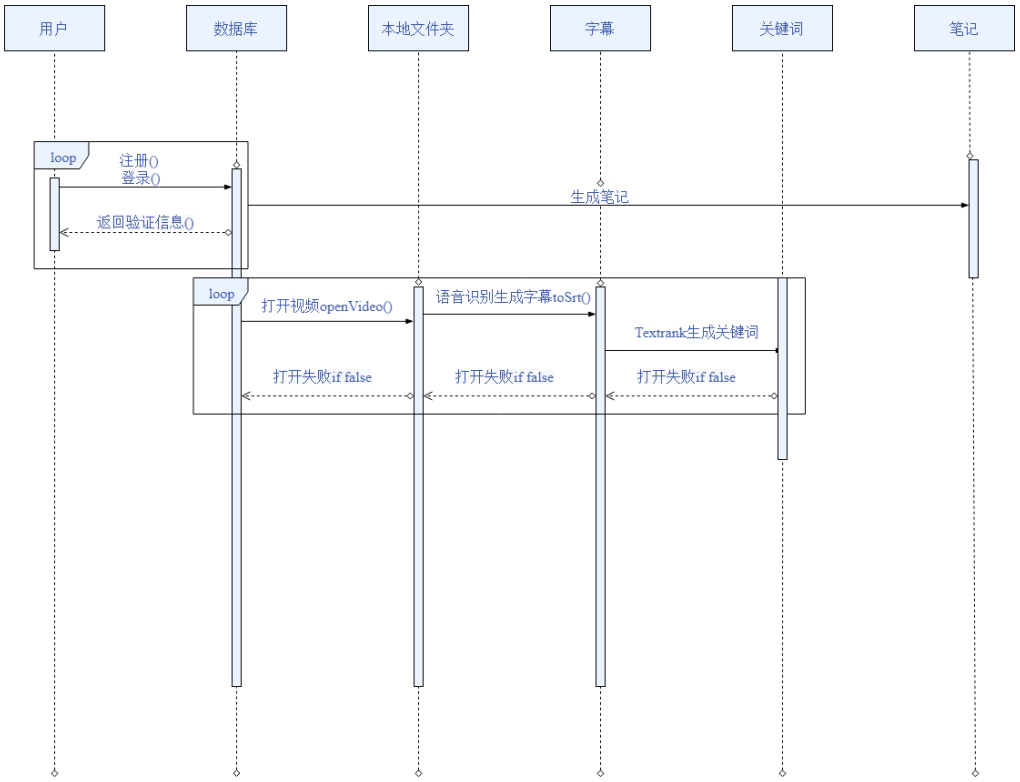


(3)增删改查记录



5.2 关键业务的顺序图

对系统的几个操作（字幕识别与生成、关键词生成）进行顺序图的设计



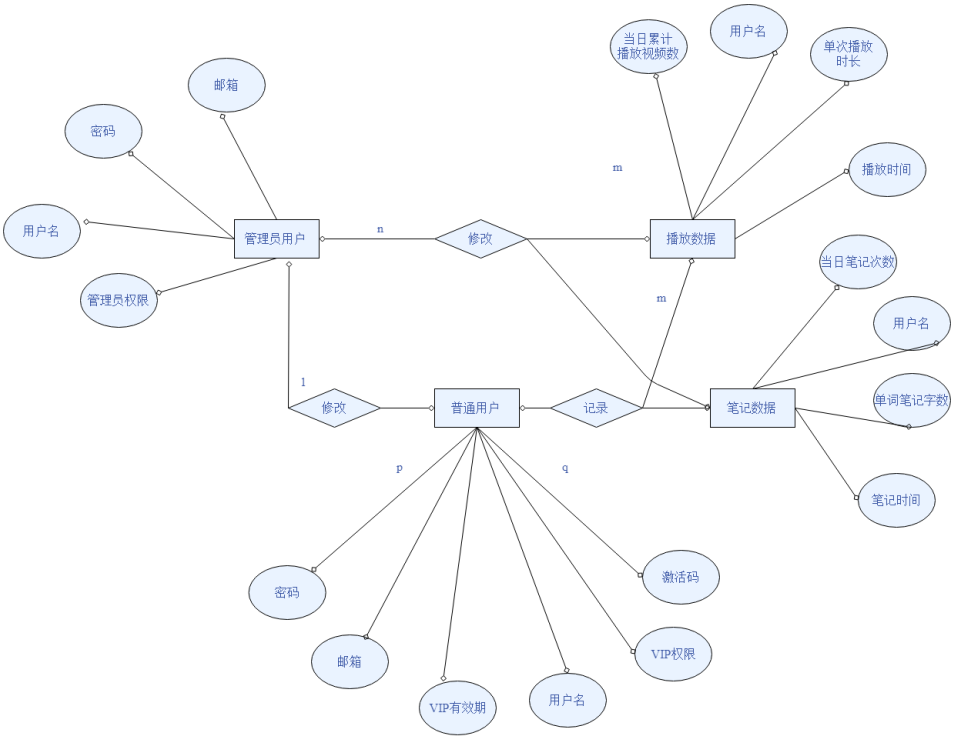
数据模型设计

6.1 实体关联设计

基于以上需求，在本系统中需要持久化存储的对象主要有为：用户信息、播放视频数据记录、笔记数据记录等。

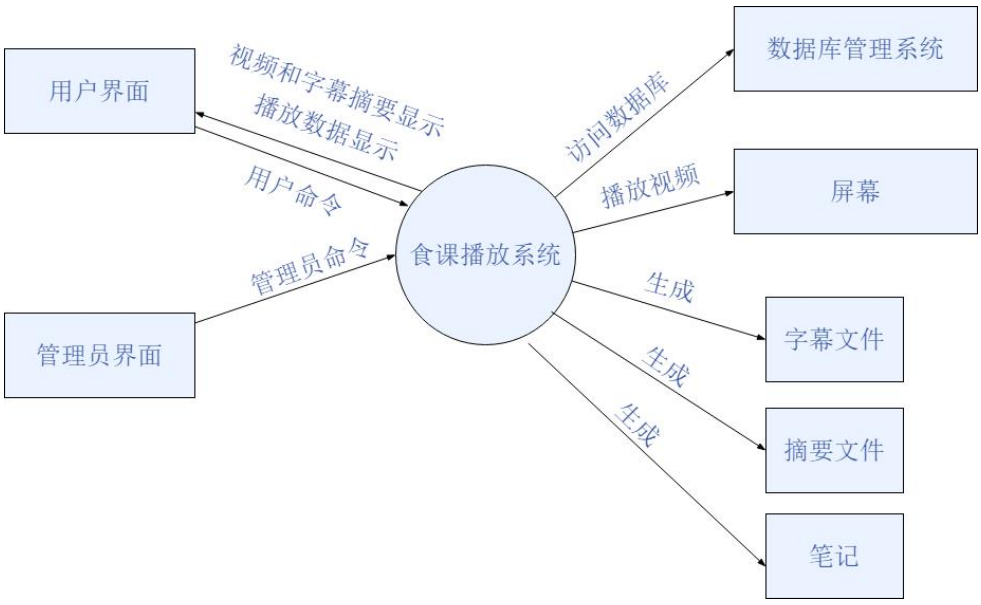
在此，创建主要的实体型：普通用户、管理员用户以及播放数据、笔记数据。还有屏幕、用户硬盘等由于篇幅所限不加入其中。

主要实体的 ER 图如下所示：



6.2 顶层数据流图设计

顶层数据流图如下所示：



对顶级数据流进行进一步的细化，接下来对一级数据流中的外部实体、转换过程、数据流向和数据源进行分析。

6.3 一级数据流图设计

对顶级数据流进行进一步的细化，接下来对一级数据流中的外部实体、转换过程、数据流向和数据源进行分析。

- 外部实体：

和顶级数据流保持一致，此处的外部主要实体型主要四个，分别是普通用户界面、管理员用户界面以及播放数据、笔记数据。系统能够通过 UI 界面很好地与用户进行交互操作。

- 转换过程：

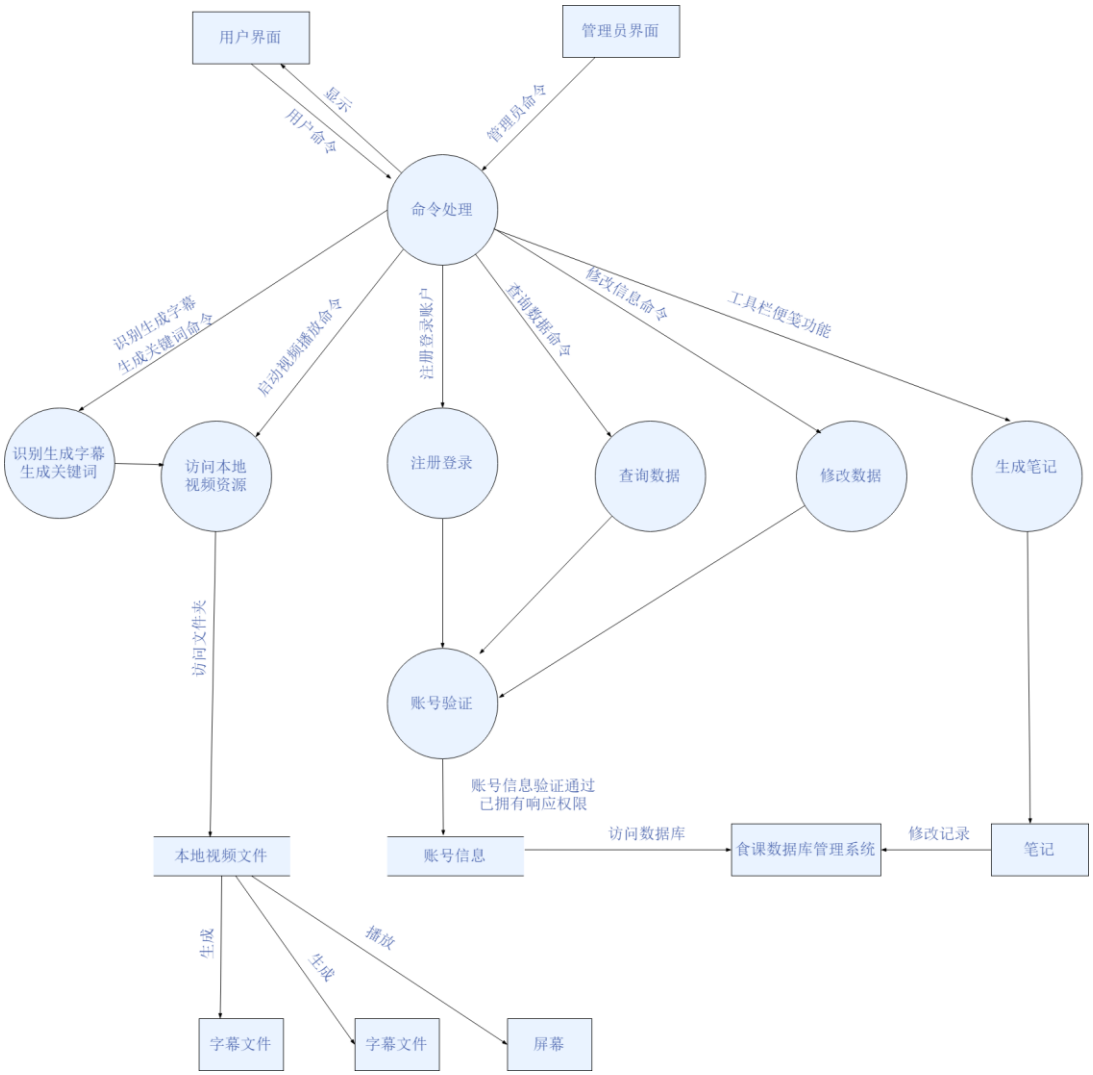
在此处的一级数据流中，转换过程经过一定的变换，进入下一阶段的功能选择，从而逐步达成用户需求。

- 数据流向：

接头指示的方向即为数据的流向。

- 数据源：

此处的数据源包括了在数据库用户信息和播放数据和笔记数据。



对系统建模之后，也就是从概念上以及逻辑上对系统结构进行设计之后，可以把这些概念模型转变成以数据库为基础的实际数据模型，并根据相应的模型和系统分析应用的具体需利用表格把数据库中各个表格相互的联系表达出来。

每张数据表都要有一个主键，主键的内容是没用重复的，所以用来区分各个用户。此外还要根据实际情况来设置数据的类型。对于一些内容较多的字段尽量将数据的长度往大设置，对该视频播放系统的各功能的数据表如下所示。

6.4 数据字典

6.4.1 账户

1) 数据文件

文件名： **users**

文件组成： **user_name 、 user_password 、 user_cdkey 、 is_vip 、 、 vip_endtime、is_admin、play_time、user_email**

数据项：user_name 数据类型：varchar；数据长度：20

数据项：user_password 数据类型：varchar；数据长度：16

数据项：user_cdkey 数据类型：char；数据长度：8

数据项：is_vip 数据类型：bool；数据长度：0

数据项：vip_endtime 数据类型：timestamp；数据长度：0

数据项：is_admin 数据类型：bool；数据长度：0

数据项：play_time 数据类型：int4；数据长度：32

数据项：user_email 数据类型：varchar；数据长度：255

2) 数据流定义

账户注册

名称：注册

简述：用户注册账户信息

数据来源：用户

数据去向：**users 表**

数据组成：user_name、user_password、user_cdkey、is_vip、、vip_endtime、is_admin

平均流量：大约 1000 次/周

高峰流量：大约 200 次/天

账户登录

名称：登录

简述：用户登录以开放对应功能

数据来源：**users 表**

数据去向：客户端

数据组成：user_name、user_password、user_cdkey、is_vip、、vip_endtime、is_admin

平均流量：大约 2000 次/周

高峰流量：大约 300 次/天

6.4.2 播放数据

1) 数据文件

文件名：**userdata**

文件组成：**user_name、play_date、playtime_once、play_count**

数据项：**user_name** 数据类型：varchar(变长字符)；数据长度：20

数据项：**play_date** 数据类型：timestamp(时间戳)；数据长度：0

数据项：**playtime_once** 数据类型：int4(整型)；数据长度：32

数据项：**playtime_count** 数据类型：int4(整型)；数据长度：32

2) 数据流定义

播放数据记录

名称：播放记录

说明：记录当前登录用户的播放数据记录

数据来源：用户

数据去向：Eatecplayer 数据库中的 Userdata 表

数据组成：user_name、play_date、playtime_once、play_count

平均流量：大约 500 次/周

高峰流量：大约 100 次/天

播放数据查询

名称：播放查询

说明：查询当前登录用户的播放数据记录

数据来源：Userdata 表

数据去向：用户界面

数据组成：user_name、play_date、playtime_once、play_count

平均流量：大约 200 次/周

高峰流量：大约 50 次/天

6.4.3 笔记数据

数据文件

文件名：**Notedata**

文件组成：**user_name、note_date、note_count、word_count**

数据项：**user_name** 数据类型：varchar(变长字符)；数据长度：20

数据项：**note_date** 数据类型：timestamp(时间戳)；数据长度：0

数据项：**note_count** 数据类型：int4(整型)；数据长度：32

数据项：**word_count** 数据类型：int4(整型)；数据长度：32

数据流定义

笔记数据记录

名称：笔记记录

说明：记录当前登录用户的播放数据记录

数据来源：用户

数据去向：Eatecplayer 数据库中的 Notedata 表

数据组成：user_name、play_date、playtime_once、play_count

平均流量：大约 500 次/周

高峰流量：大约 100 次/天

笔记数据查询

名称：笔记查询

说明：查询登录用户的播放数据记录

数据来源：Eatecplayer 数据库中的 Notedata

数据去向：用户界面

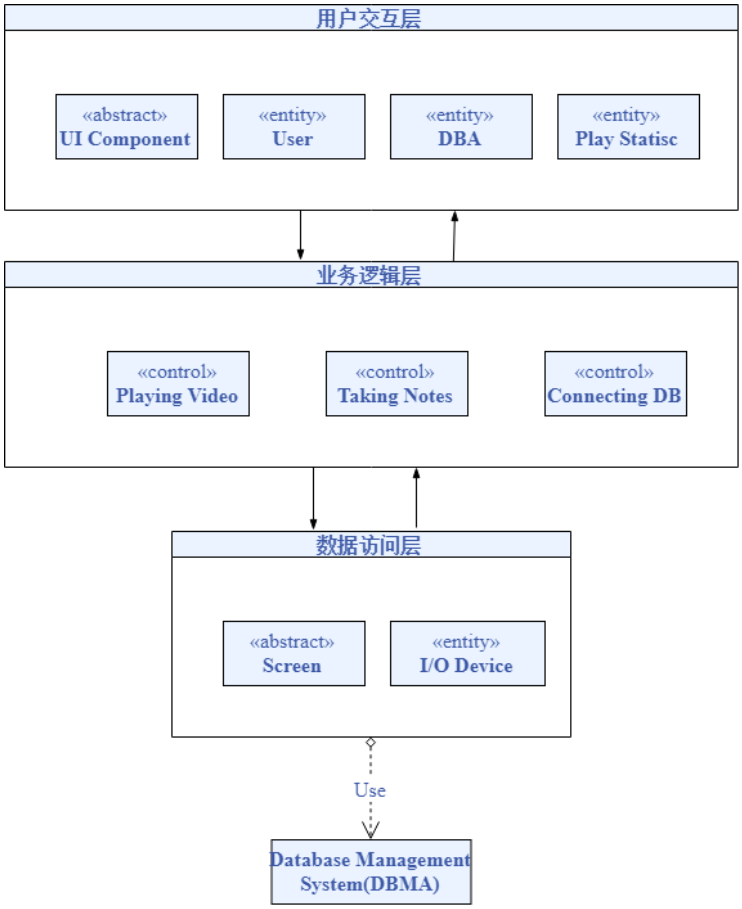
数据组成：user_name、play_date、playtime_once、play_count

平均流量：大约 200 次/周

高峰流量：大约 50 次/天

系统框架设计

根据上述分析，给出系统框架图：



（1）交互层

用户交互层在整个系统起到了比较重要的作用，系统涉及到了多个实体，例如 UI 界面、普通用户、数据库管理员、数据等。对整个框架来说，交互层是贯穿于整个软件使用的过程的。

（2）业务逻辑层

框架的业务逻辑层包括了视频播放、笔记、统计数据、注册登录账号等。

（3）数据访问层

本层主要是处理如何访问数据库中数据，包括输入输出设备与设备屏幕。

（4）数据库管理系统

用于管理数据库，可以被数据访问层访问并使用。

系统体系架构

8.1 系统整体架构

本系统主要采用 C/S 结构，该结构即客户端/服务器结构，它能充分发挥客户端 PC 的处理能力，很多工作可以在客户端处理后再提交给服务器，因此客户端响应速度快。

具体表现在以下两点：

（1） 应用服务器运行数据负荷较轻。

数据库应用由两部分组成，即客户应用程序和数据库服务器程序。二者可分别称为前台程序与后台程序。运行数据库服务器程序的机器即应用服务器。服务器程序启动时，就随时等待响应客户程序发来的请求；用户在需要对数据库中的数据进行任何操作时，客户程序就自动地寻找服务器程序，并向其发出请求，服务器程序根据预定的规则作出应答，送回结果，因此应用服务器运行数据负荷较轻。

（2） 数据的储存管理功能较为透明。

在数据库应用中，数据的储存管理功能，是由服务器程序和客户应用程序分别独立进行的，并且通常把那些不同的（不管是已知还是未知的）前台应用所不能违反的规则，在服务器程序中集中实现。所有这些，对于工作在前台程序上的最终用户，是“透明”的，他们无需过问（通常也无法干涉）背后的过程，就可以完成自己的一切工作。

软件实现与测试

9.1 软件运行流程

食课教学视频播放器，打开软件后进入开始界面，如图 5-11 所示

图 9-1 开始界面图

其中右上角状态栏自左到右的四个按钮分别代表【切换夜/日间模式】、【最小化】、【最大化】、【关闭】。软件根据当前时间判断所在时间为日间，点击【切换夜/日间模式】，可以根据用户个人喜好，手动切换至夜间模式，更加个性化与人性化，如图所示

图 9-2 夜间模式状态图

已注册用户可以输入【账号】【密码】进行登录操作，如图所示

图 9-13 用户登录输入状态图

若用户输入账号密码有误，则会弹窗提示重新输入，如图所示

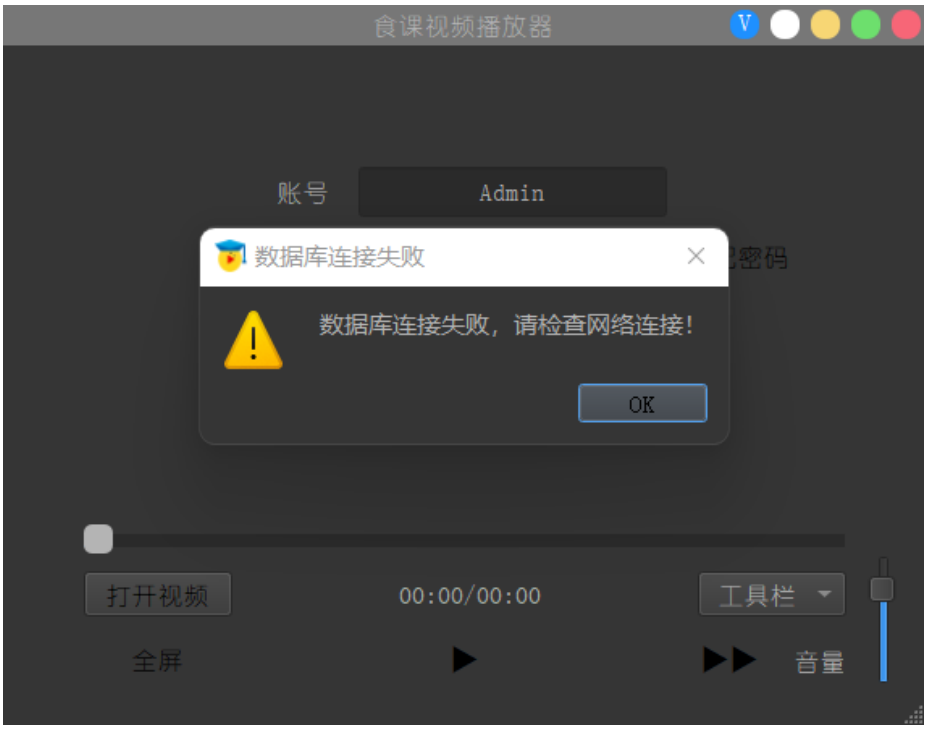


图 9-14 登录失败弹窗图

若用户忘记密码，可以点击右侧【忘记密码】找回密码，如图所示

图 9-15 找回密码界面图

若用户输入账号和密码正确，则会提示登录成功，如图所示

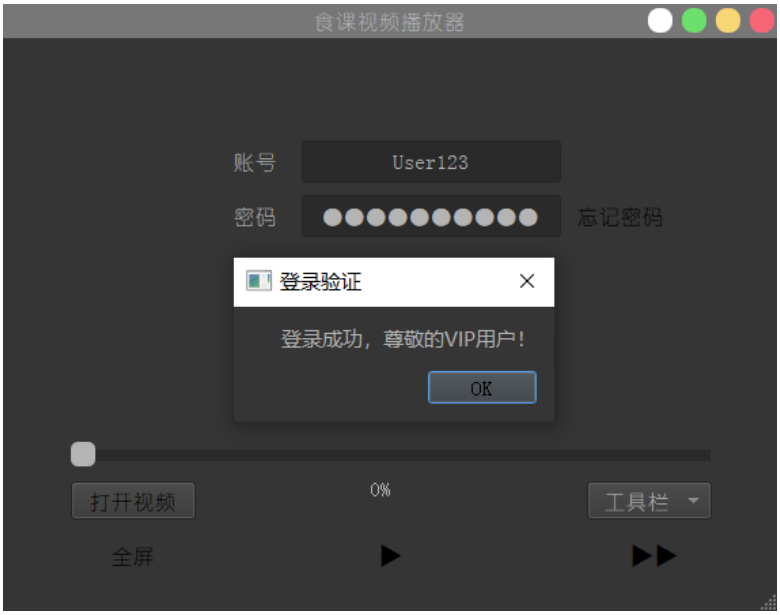


图 9-16 登录成功弹窗图

点击【打开视频】按钮，需要登录后再选择即将观看的视频文件，如图所示：

9.2 数据库建立

使用的数据库

本实验使用的数据库是由华为开发的全面友好开放，携手伙伴共同打造的企业级开源关系型数据库 openGauss。openGauss 相比于其他开源数据库主要有以下几个主要特点：高性能、高可用、高安全、易运维、全开放。

数据库运行环境

实验中用到的 openGauss 需要运行在 Linux 系统上，因此我们搭建了一个 Linux 虚拟机用于运行 openGauss。该虚拟机使用 openEuler（EulerOS）操作系统。

EulerOS 是华为自主研发的服务器操作系统，能够满足客户从传统 IT 基础设施到云计算服务的需求。EulerOS 对 ARM64 架构提供全栈支持，打造完善的从芯片到应用的一体化生态系统。EulerOS 以 Linux 稳定系统内核为基础，支持鲲鹏处理器和容器虚拟化技术，是一个面向企业级的通用服务器架构平台。

数据库创建和数据表设计

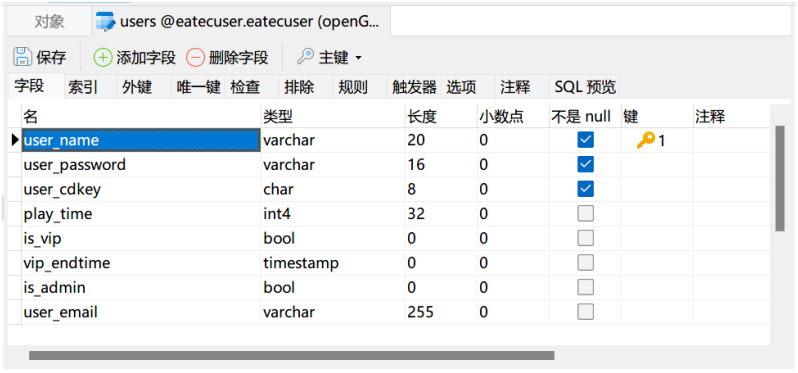
为实现用户注册登录等功能，我们建立了一个名称为 eatecuser 的数据库，拥有者为 dbuser。

为存储用户的基本信息、学习数据、笔记数据，我们在 eatecuser 数据库中建立了 3 张基本表，分别为 users、userdata、notedata，并在软件设计过程中不断

修改优化这些基本表。此外，我们还使用了 Navicat、DataStudio 软件来辅助设计数据库。

关系模式设计

（1）表 users



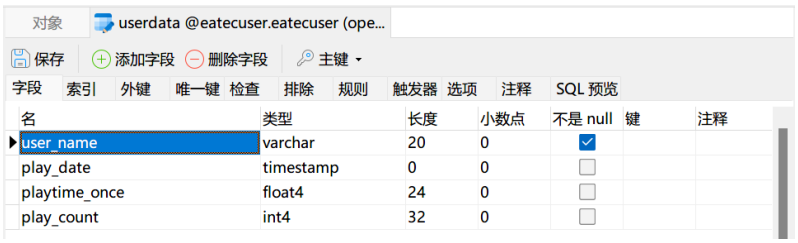
名	类型	长度	小数点	不是 null	键	注释
user_name	varchar	20	0	☑	1	
user_password	varchar	16	0	☑		
user_cdkey	char	8	0	☑		
play_time	int4	32	0	☐		
is_vip	bool	0	0	☐		
vip_endtime	timestamp	0	0	☐		
is_admin	bool	0	0	☐		
user_email	varchar	255	0	☐		

User_name：账号名；
User_password：账号密码；
User_cdkey：账号 VIP 激活码；
Play_time：该账号播放视频时间的总和；
Is_vip、is_admin：是否为 vip 或管理员；
Vip_endtime：vip 有效期；
User_email：账户邮箱；

建表语句：

```
CREATE TABLE users
(
user_name varchar(20) PRIMARY KEY,
user_password varchar(16) not null,
user_cdkey char(8) not null,
play_time int,
is_vip boolean default FALSE,
vip_endtime date,
is_admin boolean default FALSE,
user_email varchar(255)
)
```

（2）表 userdata



名	类型	长度	小数点	不是 null	键	注释
user_name	varchar	20	0	☑		
play_date	timestamp	0	0	☐		
playtime_once	float4	24	0	☐		
play_count	int4	32	0	☐		

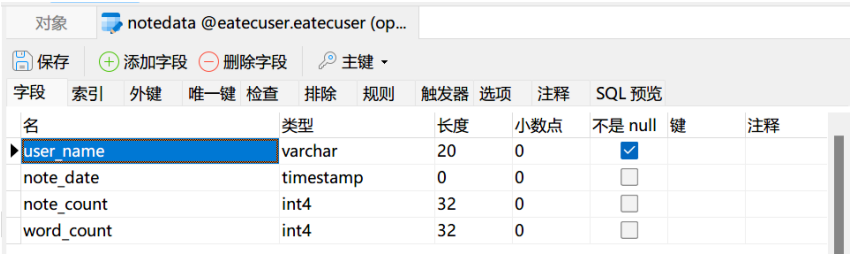
user_name：账户名
play_date：播放日期
playtime_once：播放时间

play_count：播放次数

建表语句：

```
CREATE TABLE userdata
(
user_name varchar(20),
play_date date,
playtime_once int default 0,
play_count int default 0,
FOREIGN KEY (user_name) REFERENCES users(user_name)
)
```

（3）表 notedata



user_name：账户名
note_date：笔记日期
note_count：笔记数量
word_count：笔记字数

建表语句：

```
CREATE TABLE notedata
(
user_name varchar(20),
note_date date,
note_count int default 0,
word_count int default 0,
FOREIGN KEY (user_name) REFERENCES users(user_name)
)
```

9.3 食课视频播放器模块实现

9.3.1 主窗体布局

实现思路

分别建立 GUI3.py 文件作为主窗体界面布局文件，再在 Eatecplayer.py 文件中导入该文件，编写使用该布局的主窗体类 myMainWindow，并编写__main__方法对窗体类进行实例化与展示。

流程

① 使用 QtDesigner 生成.ui 文件
② 使用 PyUIC 转换.ui 文件为 GUI3.py 文件
③ 修改并编写 GUI3.py 和 Eatecplayer.py 的 myMainWindow 类

运行结果

启动界面，食课教学视频播放器，打开软件后进入开始界面，其中右上角状态栏自左到右的四个按钮分别代表【切换夜/日间模式】、【最小化】、【最大化】、【关闭】。软件根据当前时间判断所在时间为日间，点击【切换夜/日间模式】，可以根据用户个人喜好，手动切换至夜间模式，更加个性化与人性化，如图所示：

9.3.2 注册与登录

实现思路

① 分别编写 logup()和 login()方法负责注册与登录的功能，分别与对应按钮使用信号槽函数 clicked.connect 进行绑定例如

```
self.btn_login.clicked.connect(self.login)
self.btn_rigis.clicked.connect(self.logup)
```

② 使用 self.account_text 接受账号，self.password_text 接受密码
③ 注册和登录需要连接 openGauss 数据库，我们使用 JDBC 进行连接。首先设置 URL、USER、PASSWORD，然后利用 Try 尝试创建连接
④ 若连接成功，在登录时执行 SQL 查询语句，判断当前获得的账号密码与数据库中的记录是否匹配

```
curs.execute("SELECT * FROM eatecuser.users WHERE
user_name='{}'".format(self.account_text.text()))
```

若为注册，首先判断输入是否为空，不为空，则查询

```
curs.execute("SELECT * FROM eatecuser.users WHERE
user_name='{}'".format(self.account_text.text()))
```

并且判断当前用户是否已存在，若不存在则同账户一起生成随机激活码写入到 eatecuser 数据库的 users 表中

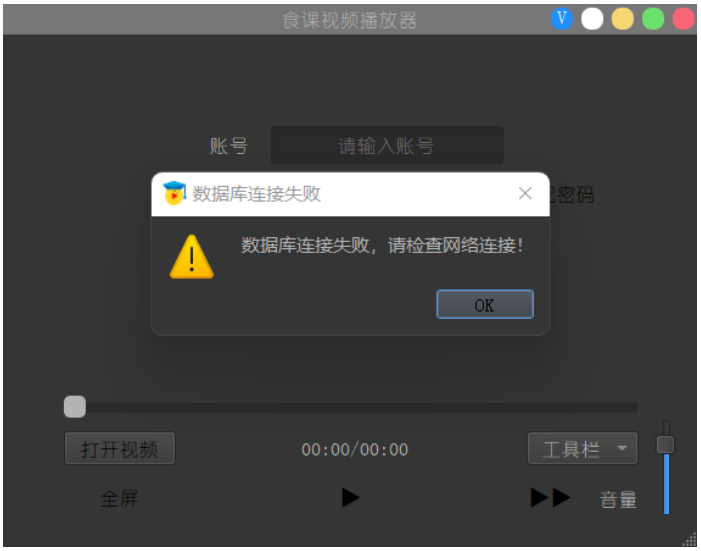
```
# 数据依次为 用户名 密码 cdkey 播放时间（秒） isvip vip到期日期 isadmin
curs.execute("INSERT INTO
eatecuser.users(user_name,user_password,user_cdkey) VALUES
('{}','{}','{}')".format(self.account_text.text(),
self.password_text.text
```

⑤ 若连接失败，则进行异常处理，使用 QMessageBox.warning()弹窗报错

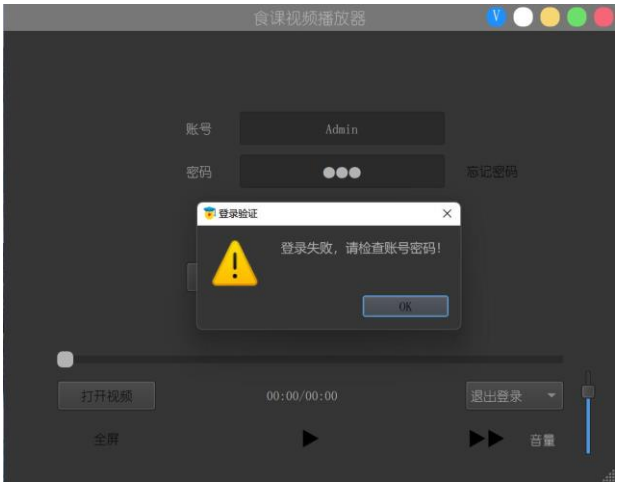
核心代码

运行结果

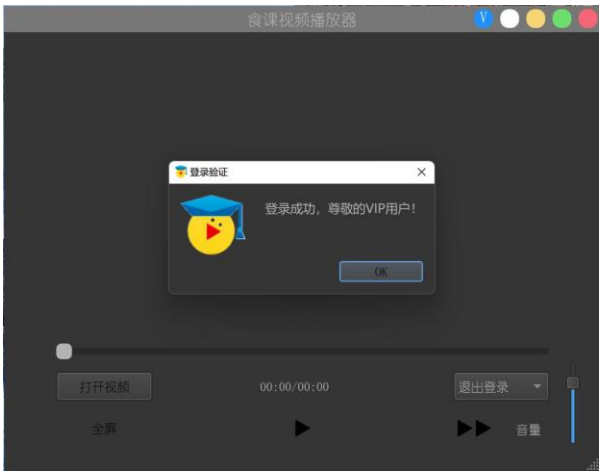
注册或登录，数据库连接失败，如图所示：



登录，账户密码错误，如图所示：

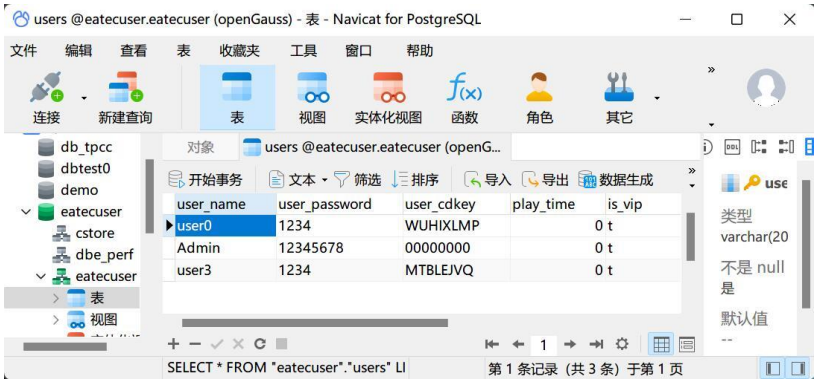


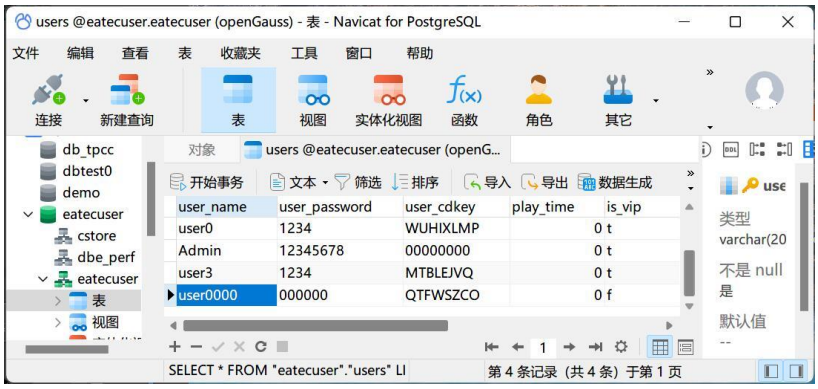
登录，账户密码正确，如图所示：



注册成功，如图所示：

注册前后数据表记录对比：





9.3.3 VIP 激活

实现思路

① 生成激活码：在注册时，随账号一起生成 8 位随机字母的激活码，如下所示".join(random.sample("ABCDEFGHIJKLMNOPQRSTUVWXYZ", 8))

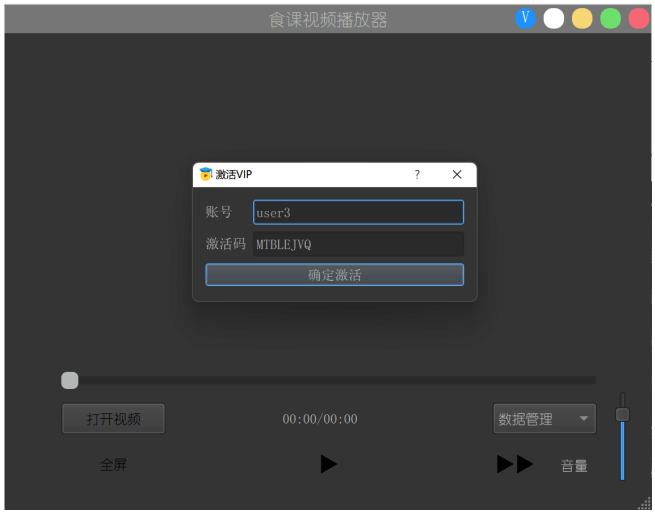
② 验证激活码：主窗体添加按钮，使用信号槽函数连接激活验证码函数。验证激活码时弹窗让用户输入激活码，系统需要连接数据库，根据该账号的用户名，查询 `eatecuser` 数据库的 `users` 表中的 `user_cdkey` 属性的值，若匹配，则置 `is_vip` 为 `True`，否则为 `False`

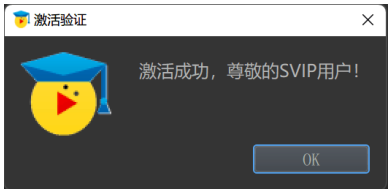
```
def validate_code(self):
    url = 'jdbc:postgresql://192.168.195.129:26000/eatecuser'
    user = 'dbuser'
    password = 'Bigdata@123'
    conn = jaydebeapi.connect('org.postgresql.Driver', url, [user, password], 'D:\Download\postgresql-42.3.1.jar')
    curs = conn.cursor()
    selectsql = "SELECT * FROM eatecuser.users WHERE user_name='{}'".format(self.le_acc.text())
    # curs.execute('set search_path to eatecuser')
    curs.execute(selectsql)
    data = curs.fetchone()
    print(data)
    if data:# 如果查询返回值非空
        if data[4]:
            QMessageBox.warning(self, "提示", "您已经是VIP用户，无需再激活。")
            curs.close()
            conn.close()
            print("数据库已关闭")
            return
    else:
        QMessageBox.warning(self, "激活验证", "激活失败，请检查账号名！")
        curs.close()
        conn.close()
        print("数据库已关闭")
        return
    if data:  # 如果查询返回值非空
        if data[2] == self.le_code.text():
            self.isVIP = True
            curs.execute("UPDATE eatecuser.users SET is_vip=TRUE WHERE user_name='{}'".format(self.le_acc.text()))
    if self.isVIP:
        QMessageBox.about(self, "激活验证", "激活成功，尊敬的SVIP用户！")
    else:
        QMessageBox.warning(self, "激活验证", "激活失败，请检查账号激活码！")
    curs.close()
    conn.close()
    print("数据库已关闭")
```

```
if data:
    QMessageBox.warning(self, "登录验证", "账号已存在，请使用其它用户名。")
else:
    ran_str = ''.join(random.sample("ABCDEFGHIJKLMNOPQRSTUVWXYZ", 8))  # 产生随机激活码
```

运行结果

验证激活码，如图所示：





在 users 表中的数据改变如下：

	用户名	密码	激活码	播放时间	VIP
1	user0	1234	WUHIXLMP	0	True
2	Admin	12345678	00000000	0	True
3	user3	1234	MTBLEJVQ	0	False

	用户名	密码	激活码	播放时间	VIP
1	user0	1234	WUHIXLMP	0	True
2	Admin	12345678	00000000	0	True
3	user3	1234	MTBLEJVQ	0	True

9.3.4 视频播放模块

实现思路

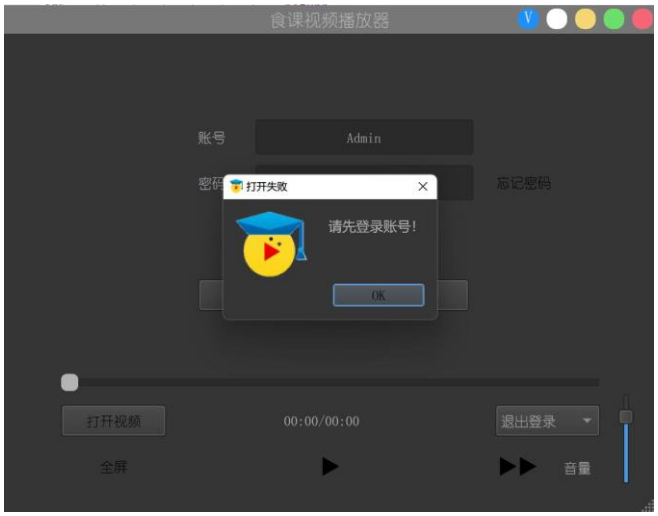
① 使用 PyQt5.QtMultimedia 的 QMediaPlayer, QMediaContent 包，用于视频播放
② 使用 QMediaContent(QFileDialog.getOpenFileUrl(self, '打开视频', filter="MP4 文件(*.mp4);;")[0])，从文件夹中打开视频
③ 创建 self.player = QMediaPlayer(None, QMediaPlayer.VideoSurface)用于视频播放
④ 编写音量控制、进度条控制与时间显示、播放\暂停、全屏等辅助视频播放的组件方法。
⑤ 其中音量控制、进度条控制主要使用了 Qslider 组件的信号槽函数，实现拖动滑条改变音量和视频播放进度。

⑥ 播放\暂停、全屏主要关注函数响应的状态变量的判断与改变。例如，播放暂停响应时需要保证视频已经打开，就需要引入 self.isFileOpen 判断，然后利用 self.isOpen 判断视频正在播放还是关闭，再使用 if else 语句分支分别响应 self.player.pause()或者 self.player.play()。全屏实现与之同理

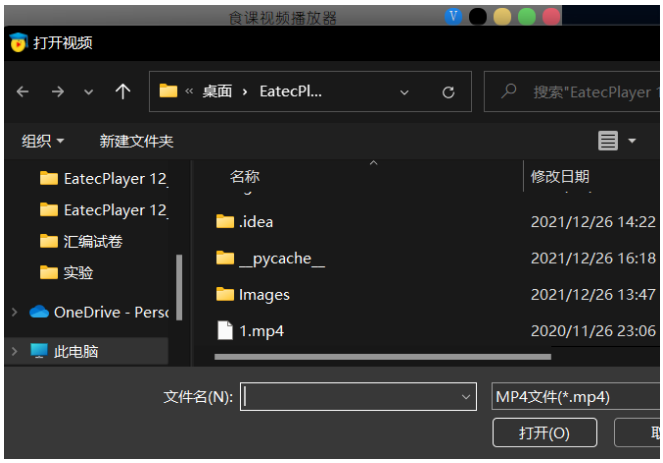
```
openVideoFile
```

运行结果

未登录状态，无法打开视频，如图所示：



登录状态，打开视频，如图所示：



拖动进度条，如图所示：

拖动音量条，调节音量：

双击或点击【全屏】按钮进行全屏播放

9.3.5 播放数据统计

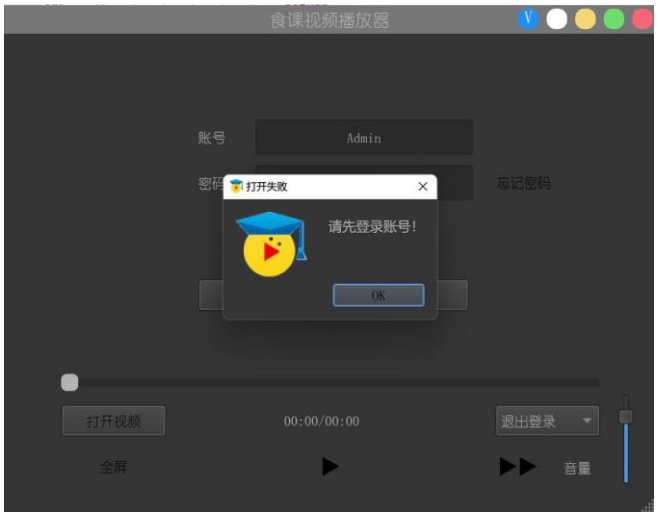
实现思路

① 启动播放器播放视频时，置当次视频播放时长 self.totalPlayTime=0
② 每当点击【播放/暂停】按钮时，使用 time.time()函数记录当前时间，并计算与上次点击该按钮的耗时，并累加得到 self.totalPlayTime
③ 在登录状态下，每次关闭播放器时，弹窗提示是否记录本次视频播放时长数据，若是，则连接数据库，将 self.totalPlayTime 连同用户名和日期一并写入 userdata 表中

```
curs.execute("INSERT INTO eatecuser.userdata(user_name,play_date,playtime_once) VALUES ('{}','{}',{})"
             .format(self.loginUser, timeStamp, str(round(self.totalPlayTime/60.0,2))))
```

运行结果

未登录状态，无法打开视频，如图所示：



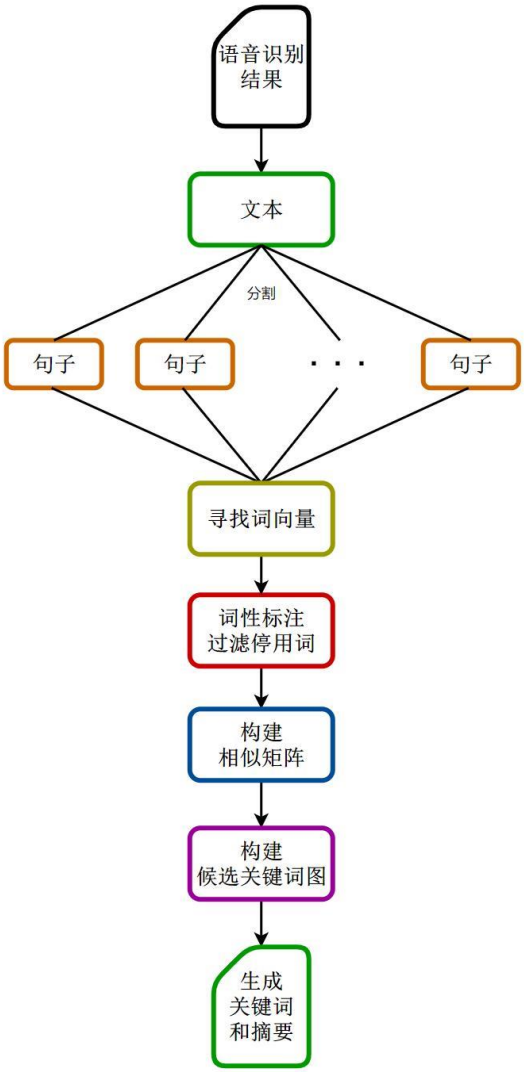
9.3.6 语音识别生成字幕和关键词生成

实现思路

首先，从视频提取音频，然后再利用百度 API 进行语音识别，生成 srt 字幕文件，再将字幕文件通过 ffmpeg 的视频文件水印添加到视频中。

生成关键词可以使用 TextRank 算法。该模块主要通过 TextRank 算法实现。TextRank 算法是一种用于文本的基于图的排序算法。其基本思想来源于谷歌的 PageRank 算法， 通过把文本分割成若干组成单元（单词、句子）并建立图模型，利用投票机制对文本中的重要成分进行排序， 仅利用单篇文档本身的信息即可实现关键词提取、文摘。和 LDA、HMM 等模型不同，简洁有效的 TextRank 算法不需要事先对多篇文档进行学习训练，所以适合用在本系统上实现快速生成关键词。

TextRank 算法是利用局部词汇之间关系（共现窗口）对后续关键词进行排序，直接从文本本身抽取；基于 TextRank 的自动文摘属于自动摘录，通过选取文本中重要度较高的句子形成文摘。使用该模块时，模块会自动读取语音识别后的纯文本文件或对自动生成的字幕进行删除无关信息的处理并转换为纯文本。随后模块将自动进行计算并找出文本中的关键词和摘要。该模块的主要工作流程如下：

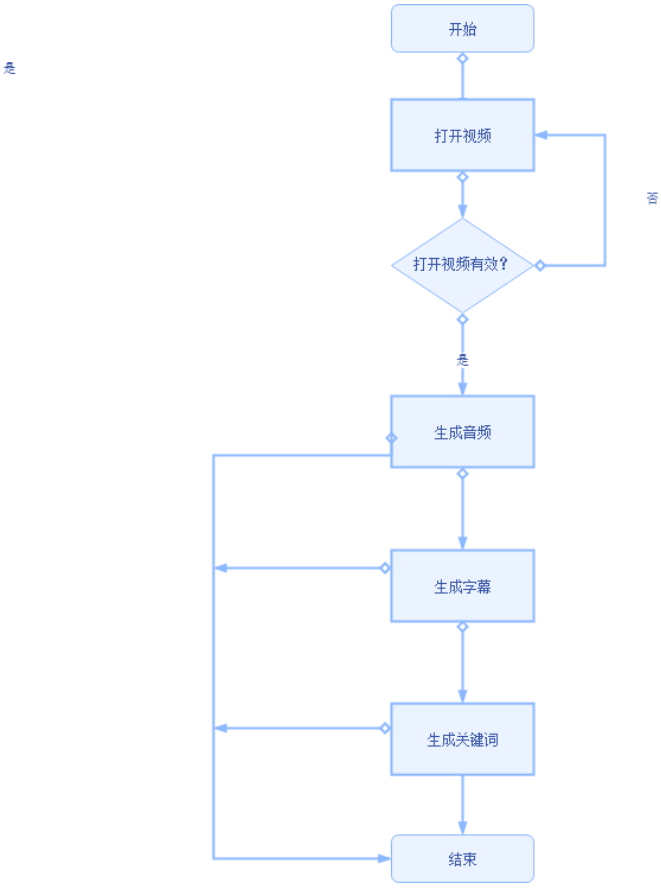


纯文本根据标点符号被分割成多个独立的句子。对于每个句子，进行分词和词性标注处理，并过滤掉停用词，只保留指定词性的词语，如名词、动词、形容词，保留下来的词语即为候选关键词。构建候选关键词图 G = (V,E)，其中 V 为节点集，由（2）生成的候选关键词组成，然后采用共现关系（co-occurrence）构造任意两点之间的边，两个节点之间存在边且仅当它们对应的词汇在长度为 K 的窗口中共现，K 表示窗口大小，即最多共现 K 个单词。根据上面公式，迭代

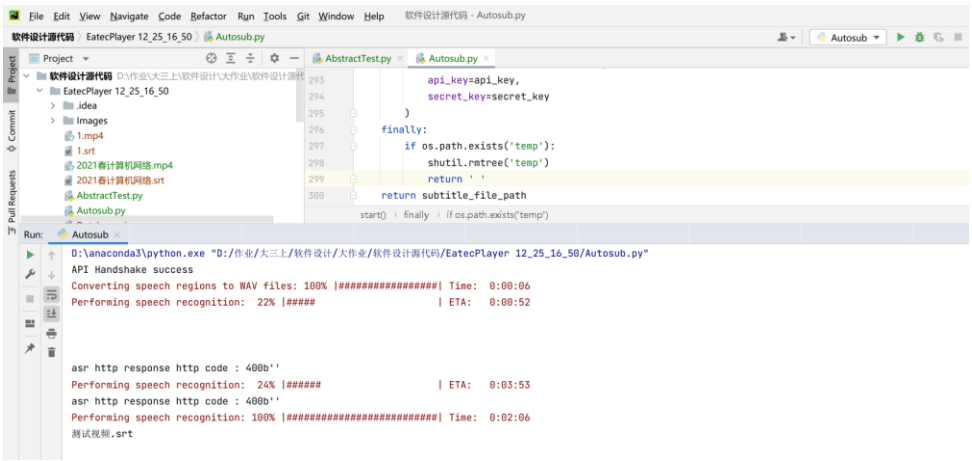
传播各节点的权重，直至收敛。对节点权重进行倒序排序，从而得到最重要的 T 个单词，作为候选关键词，并在原始文本中进行标记，若形成相邻词组，则组合成多词关键词。

虽然使用 TextRank 摘要算法具有不依赖标注数据、计算快速等优点，但是迭代计算时没有考虑句子的语义相似度，没有考虑典型的摘要句特征信息，生成的摘要句可能会出现冗余而导致文摘不够全面。

功能的总流程图如下所示：



生成字幕如下所示：



AbstractTest.py × Autosub.py × 测试视频.srt ×

```
1
00:00:00,256 --> 00:00:01,024
最近有网友

2
00:00:01,280 --> 00:00:03,072
询问苏二七战斗机在进行

3
00:00:03,328 --> 00:00:04,096
眼镜蛇机动

4
00:00:04,608 --> 00:00:06,144
眼看他寄身不断上扬

5
00:00:06,400 --> 00:00:07,168
变成后面

6
00:00:07,424 --> 00:00:08,704
去为什么不会后宫颁
```

添加字幕如下所示：

生成关键词如下所示：

9.4 食课便笺模块实现

9.4.1 窗体布局

实现思路

在 EatecPlayer.py 中创建 NotebookWindow 类继承主窗体 myMainWindow 类，在类初始化方法__init__(self)设置按钮等布局，并且设置窗口标题等

运行结果

点击右侧【工具栏】，选择【食课便笺】，打开软件后进入开始界面，其中左上角状态栏自左到右的五个按钮分别为【文件】、【编辑】、【格式】、【查看】和【插图】，如下所示

9.4.2 插图模块

实现思路

① 视频截图采用的是先利用 Spy++软件获取视频播放器窗口的句柄、标题多功能视频播放器和类 Qt5152QWindowIcon
② 再根据这些信息设置 hwnd= win32gui.FindWindow("Qt5152QWindowIcon", u'多功能视频播放器')
③ 设 screen = QApplication.primaryScreen()和
img = screen.grabWindow(hwnd).toImage()即可得到截图。
④ 设置文件名为当前时间，存入图片文件夹中
⑤ 使用 QTextEdit 的 textCursor()方法获取光标，再使用当前获取的光标的 inserImage()插入刚才的截图

⑥ 插入图片与打开视频相近，用 QFileDialog.getOpenFileName(self.centralwidget, "打开图片", "", "*.jpg;;*.png;;All Files(*)")，后面同样利用光标进行插图。

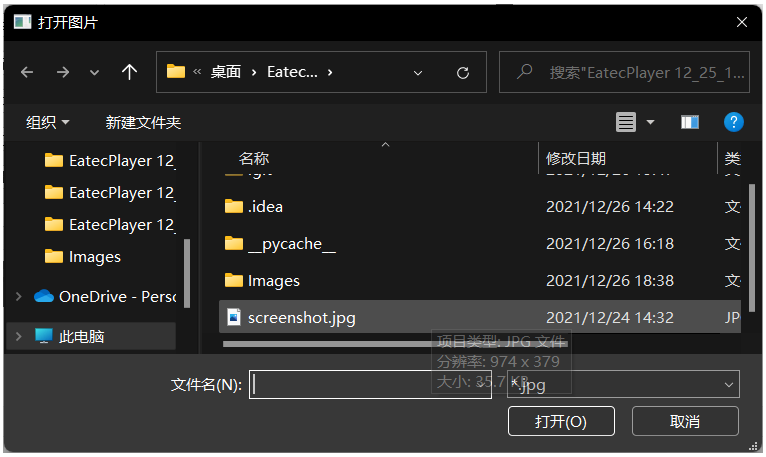
功能描述

① 快捷键“Ctrl+I”或点击“插图-插入视频截图”可将当前播放界面加入到当前笔记中
② 快捷键“Ctrl+Shift+I”或点击“插图-插入图片”可打开文件夹导入图片到笔记中

运行结果

视频截图插入，如图所示：

导入外部图片：



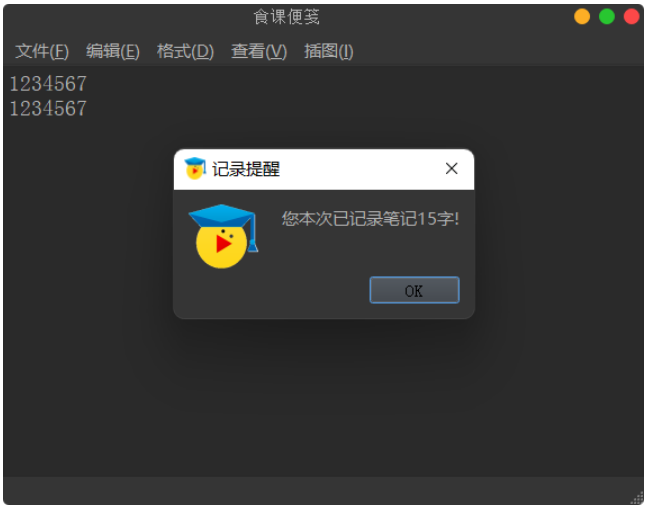
9.4.3 笔记数据统计

实现思路

① 启动播放器时，置当次笔记次数 dailyNoteCount=0
② 每当关闭笔记窗口时，调用 takeNoteRecord(self)方法统计笔记数据
③ 统计数据分为当次打开播放器的笔记次数和笔记字数，分别用 dailyNoteCount、dailywordCount 记录。每天每次调用 takeNoteRecord(self)时加 1
④ dailyNoteCount=self.plainTextEdit.toPlainText().__len__()即可得到字数
⑤ 连接数据库，找到用户当天的笔记数据
curs.execute("SELECT * FROM eatecuser.notedata WHERE user_name='{}' AND note_date='{}'"
⑥ 若没有找到数据，则插入
curs.execute("INSERT INTO eatecuser.notedata VALUES ('{}','{}',{},{})"
.format(self.loginUser, timeStamp, str(self.dailyNoteCount), str(self.dailywordCount)))
⑦ 若找到则更新当天数据
curs.execute("INSERT INTO eatecuser.notedata VALUES ('{}','{}',{},{})"
.format(self.loginUser, timeStamp, str(self.dailyNoteCount), str(self.dailywordCount)))

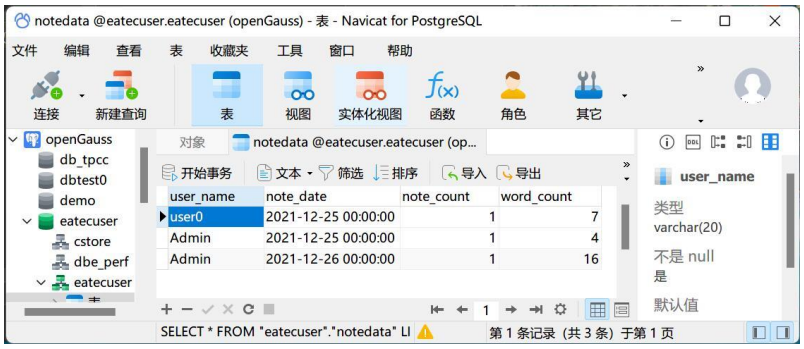
运行结果

关闭笔记的弹窗显示，如图所示：



数据库中相关记录查看，如图所示：

	日期	笔记数量	字数统计
1	2021-12-25 ...	1	4
2	2021-12-26 ...	1	16



9.5 数据管理

9.5.1 窗体布局

实现思路

在项目文件夹下新建一个 Database.ui 文件，使用 Qt 设计师软件绘制界面布局、编辑控件属性，编辑完成后使用 pyuic 工具将 Database.ui 转换成 Ui_Database.py 窗体，其中包含 Ui_MainWindow 窗体类，实现窗口显示。

流程

① 使用 QtDesigner 生成.ui 文件

② 使用 PyUIC 转换.ui 文件为 Ui_Database.py 文件
③ 编写 Database.py 继承 Ui_Database.py 中的 Ui_MainWindow 类

运行结果

启动食课教学视频播放器，登录管理员账户后在右下角工具栏选择数据管理启动界面。界面布局如下：

中间最大的区域会以表格的形式显示数据库基本表中的内容；左侧分组框包含 8 个功能按钮，其中“连接数据库”、“刷新数据”、“退出”按钮点击立即生效实现对应功能，其余按钮点击后会将上方数据库操作栏切换到对应模式，“执行”按钮会显示当前模式；上方数据库操作栏包含多个文本输入控件用于输入操作数据，“执行”按钮点击后从输入控件获取数据，执行对应的数据库操作。右侧数据统计栏显示数据库中的人数和播放时长。

9.5.2 连接数据库

实现思路

点击“b_view_data”按钮触发“view_data”函数，“view_data”函数进行一系列操作将数据显示在 table_widget 表格控件上。

流程

① 点击“b_view_data”按钮触发“view_data”函数
② “view_data”函数调用“connectdb”函数连接数据库并对所有数据查询
③ 将结果保存在一个二维数组 data 中
④ 使用 data 数据创建并实例化一个 QStandardItemModel 数据模型
⑤ 设置数据模型的表头
⑥ 设置 table_widget 的数据模型为上述数据模型

核心代码

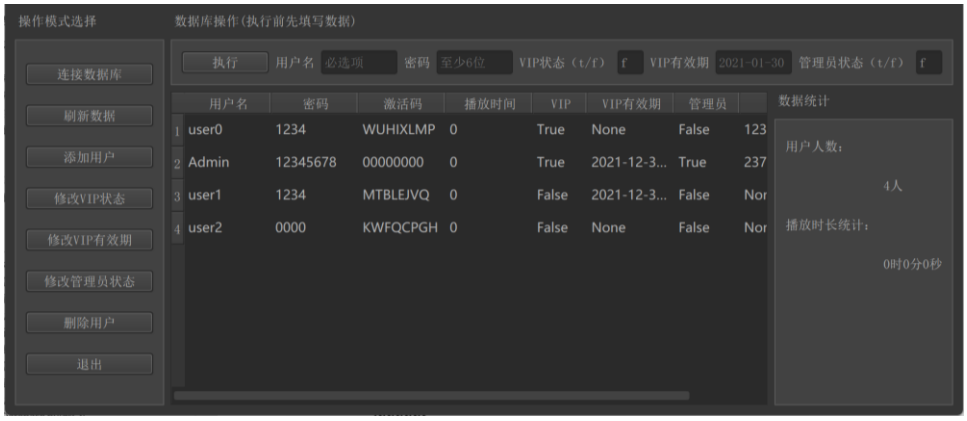
```
class DBMS(QMainWindow,Ui_MainWindow):

    def connectdb(self):
            url = 'jdbc:postgresql://192.168.195.129:26000/eatecuser'
            user = 'dbuser'
            password = 'Bigdata@123'
            # password = 'Gauss#3demo'
            print("正在连接数据库，请稍后...")
            conn = jaydebeapi.connect('org.postgresql.Driver', url, [user, password], '.\p
ostgresql-42.3.1.jar')
            print("数据库已连接")
            self.dbopen = True
            return conn

    # 浏览数据
    def view_data(self):
        try:
            timecount=0
            db = self.connectdb()
            curs = db.cursor()
            curs.execute("SELECT * FROM eatecuser.users")
            data = curs.fetchall()
            model = QStandardItemModel()
            model.setHorizontalHeaderItem(0,QStandardItem("用户名"))
            model.setHorizontalHeaderItem(1, QStandardItem("密码"))
            model.setHorizontalHeaderItem(2, QStandardItem("激活码"))
            model.setHorizontalHeaderItem(3, QStandardItem("播放时间"))
            model.setHorizontalHeaderItem(4, QStandardItem("VIP"))
            model.setHorizontalHeaderItem(5, QStandardItem("VIP 有效期"))
            model.setHorizontalHeaderItem(6, QStandardItem("管理员"))
            model.setHorizontalHeaderItem(7, QStandardItem("邮箱"))
            for i in range(0,len(data)):
                for j in range(0,len(data[i])):
                    model.setItem(i,j,QStandardItem(str(data[i][j])))
                if data[i][3]:
                    timecount=timecount+data[i][3]
            self.table_widget.setModel(model)
            self.l_usercount.setText(str(len(data))+"人")
            self.l_timecount.setText(str(timecount//3600)+"时"+str(timecount%60)+"分
"+str(timecount%3600)+"秒")
            db.close()
            self.dbopen=False
        except Exception:
            QMessageBox.warning(self, "数据库连接失败", "数据库连接失败，请检查网络连接！")
```

运行结果

点击“连接数据库”按钮后，界面如图所示：



9.5.3 数据增删改查

实现思路

点击操作模式栏内“添加用户”、“修改 VIP 状态”、“修改 VIP 有效期”、“修改管理员状态”、“删除用户”按钮后使用一个整型变量 functionMode 记录按钮对应的模式，并修改上方“执行”按钮的文本为对应模式的描述文本，并将除本模式需要使用的文本输入框以外的输入框设置为不可用状态。点击“数据库操作栏”中的按钮，从输入框获取数据，判断 functionMode 记录的模式，使用游标的 execute 函数执行 SQL 语句。

流程（以删除用户为例）

① 启动本功能页面
② 点击左侧连接数据库显示数据
③ 点击删除用户按钮
④ 将 

按钮文字设置为 

⑤ functionMode 变为 5
⑥ 将除 name_modifytext 以外的输入框可用属性设置为 False
⑦ 在 name_modifytext 输入“user2”
⑧ 点击“删除用户按钮”
⑨ 再次执行“view_data”函数以刷新数据

```
class DBMS(QMainWindow,Ui_MainWindow):
    temp = ''
    dbopen = False
    functionMode = 0
    # 执行数据库操作
    def run(self):
        try:
            db = self.connectdb()
            curs = db.cursor()
            nametext = self.name_modifytext.text()
```

```
            pwtext = self.pw_modifytext.text()
            viptext = self.vip_modifytext.text()
            datetext = self.date_modifytext.text()
            admintext = self.adm_modifytext.text()

            if self.functionMode==0:
                QMessageBox.warning(self,"提示","请先选择左侧功能")
            else:
                if nametext == '':
                    QMessageBox.warning(self, "警告", "用户名无法为空")
                    return  # 如果用户名为空不再往下执行
                # 增
                if self.functionMode==1:
                    if pwtext=='':
                        QMessageBox.warning(self, "警告", "密码无法为空")
                        return
                    else:
                        ran_str = ''.join(random.sample("ABCDEFGHIJKLMNOPQRSTUVWXYZ", 8))
# 产生随机激活码
                        curs.execute("SELECT * FROM eatecuser.users WHERE user_name='{}'"
                                     .format(nametext))
                        temp = curs.fetchone()
                        if temp:#判断用户是否重名
                            QMessageBox.warning(self,"警告","已存在该用户")
                        else:
                            curs.execute("INSERT INTO eatecuser.users(user_name,user_password,user_cdkey) VALUES ('{}','{}','{}')"
                                         .format(nametext, pwtext, ran_str))
                            QMessageBox.about(self,"","新增用户成功")
                # 改 vipstate
                elif self.functionMode==2:
                    try:
                        curs.execute("UPDATE eatecuser.users"
                                     "SET is_vip='{}'"
                                     "WHERE user_name='{}'".format(viptext,nametext))
                        succ=True
                    except:
                        succ=False
                        QMessageBox.about(self, "", "修改失败")
                    if succ:
                        QMessageBox.about(self, "", "修改成功")
                # 改 VIP 有效期
                elif self.functionMode==3:
                    ifdateright=False
                    # 判断日期是否合法
                    try:
                        time.strptime(datetext,"%Y-%m-%d")
                        ifdateright = True
                    except:
                        ifdateright = False
                    if not ifdateright:
                        QMessageBox.warning(self,"警告","日期格式有误")
                    else:
                        try:
                            curs.execute("UPDATE eatecuser.users SET vip_endtime='{}' WHERE user_name='{}'".format(datetext, nametext))
                        except:
                            QMessageBox.warning(self,"","修改失败")
                # 改管理员
                elif self.functionMode==4:
                    try:
                        curs.execute("UPDATE eatecuser.users SET is_admin='{}' WHERE user_name='{}'".format(admintext, nametext))
                    except:
                        QMessageBox.warning(self,"","修改失败")
                # 删
                elif self.functionMode==5:
                    try:
                        curs.execute("DELETE FROM eatecuser.users WHERE user_name='{}'".format(nametext)
```

```
    )
                    except:
                        QMessageBox.warning(self,"","删除失败")
                db.close()
                self.refresh()
        except Exception:
            QMessageBox.warning(self, "数据库连接失败", "数据库连接失败，请检查网络连接！")
    # 删除一行数据
    # 刷新
    def refresh(self):
        self.view_data()
    # 添加一行数据行
    def add_row_data(self):
        self.functionMode=1
        self.b_run.setText("添加用户")
        self.pw_modifytext.setEnabled(True)
        self.vip_modifytext.setEnabled(True)
        self.date_modifytext.setEnabled(True)
        self.adm_modifytext.setEnabled(True)
    # 删除一行数据
    def del_row_data(self):
        self.functionMode=5
        self.b_run.setText("删除用户")
        self.pw_modifytext.setEnabled(False)
        self.vip_modifytext.setEnabled(False)
        self.date_modifytext.setEnabled(False)
        self.adm_modifytext.setEnabled(False)
    # 修改 VIP 状态
    def altervip(self):
        self.functionMode = 2
        self.b_run.setText("修改 VIP 状态")
        self.pw_modifytext.setEnabled(False)
        self.vip_modifytext.setEnabled(True)
        self.date_modifytext.setEnabled(False)
        self.adm_modifytext.setEnabled(False)
    # 修改 VIP 有效期
    def altervipdate(self):
        self.functionMode = 3
        self.b_run.setText("修改有效期")
        self.pw_modifytext.setEnabled(False)
        self.vip_modifytext.setEnabled(False)
        self.date_modifytext.setEnabled(True)
        self.adm_modifytext.setEnabled(False)
    # 修改管理员状态
    def alteradmin(self):
        self.functionMode = 4
        self.b_run.setText("修改管理员状态")
        self.pw_modifytext.setEnabled(False)
        self.vip_modifytext.setEnabled(False)
        self.date_modifytext.setEnabled(False)
        self.adm_modifytext.setEnabled(True)

```

运行结果（以删除用户为例）

如图所示：


操作模式选择
数据库操作(执行前先填写数据)
连接数据库
刷新数据
添加用户
修改VIP状态
修改VIP有效期
修改管理员状态
删除用户
退出
删除用户
用户名
user2
密码
至少6位
VIP状态（t/f）
f
VIP有效期
2021-01-30
管理员状态（t/f）
f
用户名 密码 激活码 播放时间 VIP VIP有效期 管理员
1 user0 1234 WUHIXLMP 0 True None False 123
2 Admin 12345678 00000000 0 True 2021-12-3... True 237
3 user1 1234 MTBLEJVQ 0 False 2021-12-3... False Nor
数据统计
用户人数：
3人
播放时长统计：
0时0分0秒

9.6 个人账户管理和信息修改

9.6.1 窗体布局

实现思路

在项目文件夹下新建一个 MyAccount.ui 文件，使用 Qt 设计师软件绘制界面布局、编辑控件属性，编辑完成后使用 pyuic 工具将 MyAccount.ui 转换成 Ui_MyAccount.py 窗体，其中包含 Ui_MainWindow 窗体类，实现窗口显示。

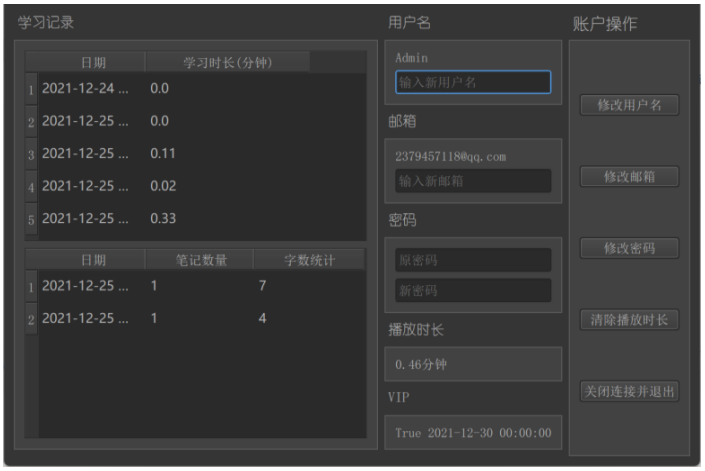
流程

① 使用 QtDesigner 生成.ui 文件
② 使用 PyUIC 转换.ui 文件为 Ui_ MyAccount.py 文件
③ 编写 MyAccount.py 继承 Ui_ MyAccount.py 中的 Ui_MainWindow 类
④ 启动该子程序时子程序先项目文件夹读取 un.txt 文件中的用户名信息
⑤ 使用读取到的信息去数据库查询该账户的数据
⑥ 显示查询到的数据

运行结果

启动食课教学视频播放器，登录账户后在右下角工具栏选择个人账户管理启动界面。界面布局如下：

左侧两个表分别展示每次播放视频的时长和每次保存笔记的数量和字数统计。中间栏展示用户名、邮箱、播放时长、VIP 信息，右侧放置账户信息修改的操作。

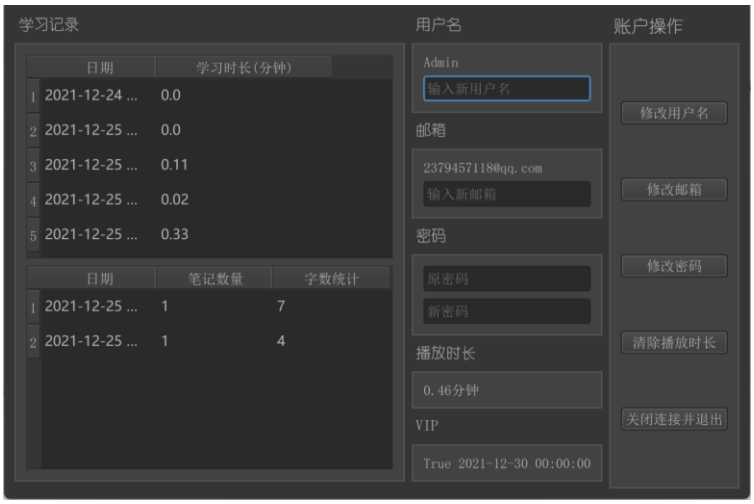


9.6.2 学习记录展示

实现思路

窗体左侧垂直排列两个表格控件，分别为他们设置两个数据模型。如果数据库连接成功则分别查询 userdata 基本表和 notedata 基本表，将数据遍历写入数据模型，最终在两个表格控件中显示。

运行结果

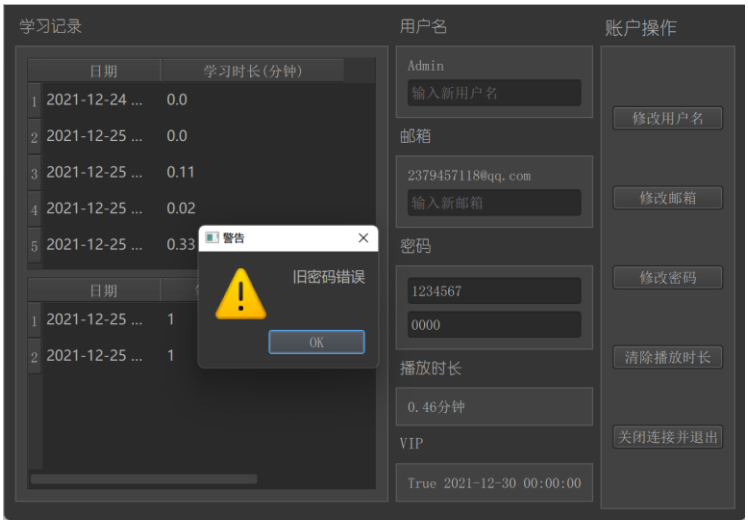


9.6.3 用户信息修改

实现思路

点击最右侧信息修改按钮后，程序从中间的文本输入框获取输入的新信息，并查询一次数据库来判断输入的新信息是否合法（如相同用户名是否已存在、旧密码是否输入正确），如果合法则执行 execute 函数执行 SQL 语句修改数据库中的信息。

运行结果


学习记录
日期
学习时长(分钟)
1 2021-12-24 ... 0.0
2 2021-12-25 ... 0.0
3 2021-12-25 ... 0.11
4 2021-12-25 ... 0.02
5 2021-12-25 ... 0.33
日期
1 2021-12-25 ... 1
2 2021-12-25 ... 1
警告
旧密码错误
OK
用户名
Admin
输入新用户名
邮箱
2379457118@qq.com
输入新邮箱
密码
1234567
0000
播放时长
0.46分钟
VIP
True 2021-12-30 00:00:00
账户操作
修改用户名
修改邮箱
修改密码
清除播放时长
关闭连接并退出

总结

实现这个视频播放器管理系统看似简单，动手开始做才发现有各种各样的问题。数据库的相关内容：例如 E-R 图中实体的确定，数据流图的设计，数据库中各个字段数据类型如何设置，甚至通过 JDBC 连接数据库都出现了一些问题，我们花了一些时间去处理。而软件的核心功能，视频播放器也由于含有许多组件，需要编写许多代码去实现功能。其中我认为花费许多时间的有视频的进度条，尽管它的逻辑看似简单，但由于 Qslider 这个组件我不太熟悉，导致在实现的时候没有头绪，我在翻阅大量资料之后才明白如何利用它进行进度调节。

截止这个学期，我们学习了许多原理，例如面向对象程序设计、数据结构、数据库、软件工程等等，虽然原理比较清楚了，但到了实践还是觉得光知道原理是远远不够的。

“纸上得来终觉浅，绝知此事要躬行。”因为在实践过程中仍然出现一些问题，这些问题在接触 openGauss 和 PyQt5 的相关操作之前是无法预料到的，因此需要我们有一定的临场应变的 debug 能力，无论是分析 bug 的原因还是在论坛网站中搜集资料寻找解决方法，都是作为一个程序员或者工程师的基本素质，这些是我为完成本次期末作业得到的训练

参考文献

[1] (美)罗杰 S.普莱斯曼．软件工程实践者的研究方法(原书第 8 版・本科教学版)．北京：机械工业出版社，2016.12
[2] 王珊，萨师煊．数据库系统概论．北京：高等教育出版社，2014.9