摘要

此app为小视频集合平台，采取简洁、直观的页面及交互方式让用户有舒适的体验，通过对app界面布局，视频浏览，视频上传，用户的互动等功能的规划来实现崭新的视听环境。

此app通过用户上传的视频来构造视频广场主页，通过用户的点赞、评论、下载量来构造热门视频页面。除此之外，用户可以关注自己感兴趣的用户，并且可以实时查看关注用户的动态。

采用Java语言进行编写，使用MySql和Redis数据库，Spring Boot+ Spring MVC+MyBatis进行后端编写。使用flutter进行前端开发。

关键字： 短视频, flutter, springboot, nginx, ffmpeg, hls

英文摘要

第一章 绪论

* 1. 研究意义与价值

短视频即短片视频，是一种以互联网内容传播方式，在互联网新媒体上传播、时长在5分钟以内的视频，是指在各种平台上播放的、适合在移动状态和短时间休闲状态下观看的、高频推送的内容，几秒到几分不等，内容融合了技能分享、幽默搞怪、时尚潮流、社会热点、街头采访、公益教育、广告创意、商业定制等主题。短视频不同于微电影和直播，不需要具有特定的表达形式和团队配置要求，具有生产流程简单、制作门槛低、参与性强等特点。

随着移动终端的高速发展，人们的日常生活、学习、工作等，越来越依赖于手机。在物质生活过剩的今天，追求精神生活就是很大一部分人对生活的定义，伴随着手机的普及，信息传播的及时性和便利性也得到了极大的提升，这也为短视频的传播提供了渠道。

社会从过去的封建保守到如今的越来越开放进步，人们对于自我表达越发大胆 ，追求标新立异。而短视频正好迎合 了这部分人群的需求。操作的简单，短视频的表现方式比图纸更加直观具有冲击力，比图纸形式更能展现更加生动的内容；由于科技的发展，原本只能靠专业摄像机才能录制的视频，如今用智能手机就能制作出画质的小视频，并且传播方式也只需简单的步骤；在生活紧凑的今天，人们的精神状态是经常的，需要得到有效的缓解，而短视频利用人们碎片化的时间做到随时放松，适用于多种场合。从娱乐的放松性上来说，短视频的性价比是做高的。

1.2 发展概况

21世纪是信息化的时代，随着移动终端的普及和网络的飞速发展，短平快的大流量传播内容逐渐获得各大平台、粉丝以及资本家的青睐。我国的短视频发展大致可以分为三个阶段：2013-2015年，以秒拍、美拍等短视频软件为起点，短视频平台逐渐进入公众视野，短视频这一新形态的传播方式开始被用户接受；2015-2017年，以抖音、为代表的短视频应用获得资本的青睐，各大互联网巨头围绕短视频领域展开争夺，一些如电视、报纸等传统媒体也纷纷加入这场大潮；2017年至今，短视频垂直细分模式全面开启。

从2017年开始，随着短视频热度的飞速提升，视频行业逐渐崛起一批优质的UGC(User Generated Content，用户生成内容)内容制作者。在现如今的短视频产业链中，其上游部分的核心主要包括了UGC、PGC（Professionally Generated Content）在内的大量内容创造者。下游部分主要包括了视频平台和其他分发渠道，其中短视频平台是短视频内容最主要的生产场所，之后在平台内外进行多渠道分发。短视频的覆盖范围正在急速扩张，影响力也越来越大。

1.3 论文结构

本论文先阐述了系统研究研究意义与价值以及发展概况。然后按照软件工程的思想对系统从需求分析一直到软件测试都做了非常全面的设计。

论文第二章主要介绍了本系统所使用的技术，并且分析了这些技术的优势之处。第三章主要从系统的可行性分析、功能和性能等方面进行了详细的需求分析。第四章主要介绍了系统的概要设计，包括系统的功能结构、系统的流程设计以及数据库的概要设计。第五章主要介绍了系统的详细设计与实现过程。第六章主要介绍了系统的测试过程。第七章主要是结论和展望。在本论文的最后列出了实现本系统所参阅的参考文献以及表达了真诚的致谢。

第二章 相关技术

* 1. 开发工具的选择

·2.1.1 后端开发工具的选择

IntelliJ IDEA，是Java编程语言开发的集成环境(Integrated Development Environment，IDE)，IntelliJ在业界被公认为最好的java开发工具，尤其在智能代码助手、代码自动提示、重构、J2EE支持、各类版本工具(git、svn等)、JUnit、CVS整合、代码分析、 创新的GUI设计等方面的功能可以说是超常的。Idea由捷克软件公司JetBrains在2001年1月时推出最初版。其强大的功能几乎可以让用户不用鼠标便达到用户的目的，大大的加快了开发的速度。本app服务端使用idea作为开发IDE。

·2.1.2 前端开发工具的选择

Android Studio，是谷歌推出的一个Android集成开发软件，基于 Intellij IDEA，其强大的功能以及提供的各种插件极大的便利了开发者的开发。本app客户端使用Android Studio作为开发IDE。

* 1. 操作系统的选择

服务端接口部署在阿里云学生云服务器上（Linux），阿里云服务器(Elastic Compute Service, ESC)是一种计算能力可弹性伸缩的计算服务，其管理方式比物理服务器更简单高效，具有安全稳定、数据可靠、性能卓越、弹性伸缩、节约成本、简单易用等特性。服务器的价格昂贵，普通学生无法承受这昂贵的价格，而阿里云学生云服务器不光价格上完全在学生的承担范围之内，在性能上也能够完全满足学生一般的需求。本项目采用前后端完全分离的设计方式，将服务端接口部署在云服务器上，用户只需安装移动端软件就能随时随地使用此app。

Linux，全称GNU/Linux，是一套免费使用和自由传播的类UNIX操作系统，是一个基于 POSIX和Unix的多用户、多任务、支持多线程和多CPU的操作系统。Linux不仅系统性能稳定，而且是开源软件。其核心防火墙性能高效、配置简单，保证了系统的安全。Linux具有开放源码、没有版权、技术用户多等特点，开放源码意味着任何机构和个人都能自由的查看Linux的底层源码，并且自由的修改和再发布。Linux具有强大的社区支持，在社区里能提供了大数的问题解决方案。

Java是跨平台的语言，Linux对Java提供了优秀的支持。以最少的资源消耗，最方便的操作，完成整个项目的运行。

* 1. 数据库的选择

MySql，MySql是一个关系型数据库管理系统，是最流行的关系型数据库之一。MySql完美支持高性能和可伸缩的网络和嵌入式数据库应用程序。MySql使用的SQL语言是用于访问数据库的最常用标准化语言，具有体积小、速度快、总体拥有成本低、开放源码等特点。

* 1. 编程语言的选择

2.4.1客户端编程语言的选择

Dart，dart是谷歌开发的计算机编程语言，是面向对象、类定义、单继承的语言支持接口(interfaces)、混入(mixins)、抽象类(abstract classes)、具体化泛型(reified generics)、可选类型(optional typing)和sound type system。

Flutter，flutter是谷歌的移动端跨平台UI框架，可以快速的在iOS和Android上构建高质量的原生用户界面。Flutter选择Dart作为开发框架和weight的语言，其可以与现有的代码一起工作。并且Flutter是免费的、开源的。简单来说、Flutter是一款移动应用程序SDK，包含框架、控件和一些工具，可以用一套代码同时构建iOS和Android应用，并且应用上可以达到原生应用一样的性能。

2.4.2 管理端编程语言的选择

JavaScript（js）是一种高级的、解释性语言，是一门基于原型、函数先行、多凡事的语言。JavaScript被世界上绝大多数网站所使用，也被世界主流浏览器（Chrome、IE、FireFox、Safari、Opera）支持。

Vue.js是一款目前火热的JavaScript前端框架，旨在更好的组织与简化Web开发。Vue关注的核心是MVC模式中的视图层，并且，vue.js能更方便的获取数据更新，并通过组件内部特定的方法实现视图与模型的交互。

2.4.3服务端编程语言的选择

Java，Java是一门面向对象的语言，具有简单性、面向对象、分布式、健壮性、安全性、平台独立性与可移植性、多线程、动态性等特点，被广泛运用与企业级Web应用开发和移动应用的开发。

* 1. 技术框架
     1. Spring

Spring框架是Java平台上的一种开源应用框架，提供具有控制反转特性（IOC）的容器，能帮助开发者创建对象以及建立对象之间的依赖关系，这能够降低对象与对象之间的耦合度。

* + 1. Spring Boot

Spring Boot是一个开源的轻量级框架，基于Spring4.0，其不光继承了Spring框架原有的优秀特性，而且还通过简化的配置来进一步简化了Spring应用的整个的搭建和开发过程。另外，Spring Boot框架中还有两个非常重要的策略：开箱即用和约定优于配置。

* + 1. SpringMVC

SpirngMVC是一个MVC集成框架，用来简化基于MVC架构的web应用开发。

* + 1. MyBatis

MyBatis是一个基于Java的持久层框架，支持定制化SQL、存储过程以及高级映射。MyBatis可以使用简单的注解或者XML的方式来配置和映射原生信息，将接口和Java对象映射成数据库中的记录。

* + 1. JSON

JSON是一种轻量级的数据交互格式，采用了完全独立于编程语言的文本格式来存储和表述数据，易于人的阅读理解，同时也易于机器解析和生成，有效的提升网络传输效率。

* + 1. Redis

Redis是一个内存中的数据库，它保存在磁盘上。数据模型是键值，但是支持许多不同类型的值:字符串、列表、集、排序集、散列、流、超对数、位图。Redis通常被称为数据结构服务器。这意味着Redis通过一组命令提供对可变数据结构的访问，这些命令使用带有TCP套接字和简单协议的服务器-客户端模型发送。因此，不同的进程可以以共享的方式查询和修改相同的数据结构。

* + 1. JUnit

Junit是一个开源的Java测试框架，一个编写可重复测试的简单框架。JUnit是一个编写可重复测试的简单框架。它是用于单元测试框架的xUnit体系结构的一个实例。

第三章 系统分析

3.1 系统可行性分析

本系统主要从技术可行性，操作可行性以及社会效益可行性角度出发进行分析。

技术可行性：本系统客户端采用谷歌开发的计算机编程语言Dart进行App开发和谷歌的移动端跨平台框架Flutter进行优化。开发者可以通过 Dart语言开发 App，一套代码同时运行在 iOS 和 Android平台。Flutter主打跨平台、高保真、高性能，Flutter还提供了丰富的组件、接口，开发者可以很快地为 Flutter添加 native扩展，大大的加快了开发速度。同时 Flutter还使用 Native引擎渲染视图，这无疑能为用户提供良好的体验。本系统服务端采用Java语言进行业务处理服务器开发。为了方便开发过程，本系统使用了Spring Boot、SpringMVC和MyBatis开源框架进行系统分层开发，该服务端分为控制层、业务逻辑层和持久层。在使用开发框架的过程中，会使用注解简化配置，如此可以使得在开发过程中将开发的重心放在业务处理上。持久层使用的后台数据库是MySQL和Redis，需要频繁读取的或者有时效性的数据使用具有快速读写特点的Redis进行存储，而需要持久存储的数据则用MySQL进行存储，如此能有效提高数据的读写操作，性能比单一数据库更加优越。在服务端开发过程中和开发完成后，使用开源的JUntil进行单元测试，尽可能的消除系统bug，保证可靠性。

操作可行性：本系统客户端为App形式，服务端接口部署在云服务器上，用户只需要下载安装安装包就能随时随地使用本系统。Flutter具有跨平台的特性，只需要在IDE上生成不同格式的安装包，apk或者ios格式，不同的系统安装不同的安装包，就能运行本系统。

社会效益：在如今社会，人们的思想越来越开放，表现自我的方式也越来越多，而短视频集合平台作为一个能发表自己的作品、浏览他人的作品、相互互动的短视频平台，完美的符合了用户的需求。在紧凑的现代社会生活中，人们的压力是巨大的，而短视频集合平台可以让用户能随时随地利用一些碎片时间浏览短视频，来达到让自己放松的效果。因此，小视频集合是巨大的。

3.2 系统需求分析

3.2.1 功能需求分析

本系统分为用户端和管理端。

用户端供普通用户使用，用户在没有登录系统之前仅仅能够浏览推荐广场和热度广场上的视频，而其他的关注用户、发表视频、点赞、评论和下载功能无法使用，注册登录后才能使用这些功能。不光如此，登录之后用户还具有查看自己的历史点赞的视频、自己关注用户、关注用户发表的视频、粉丝、删除发表的视频等功能。

管理员供管理员使用，管理员登录后能够对用户和用户发表的视频进行操作。对用户，管理员具有限制用户登录、重置用户密码、删除用户、浏览用户具体信息等功能，对用户发表的视频，管理员具有审核视频、查看视频具体信息、浏览视频、删除视频等功能。

系统的用例图如图3.1所示，系统的功能模块如同3.2所示。

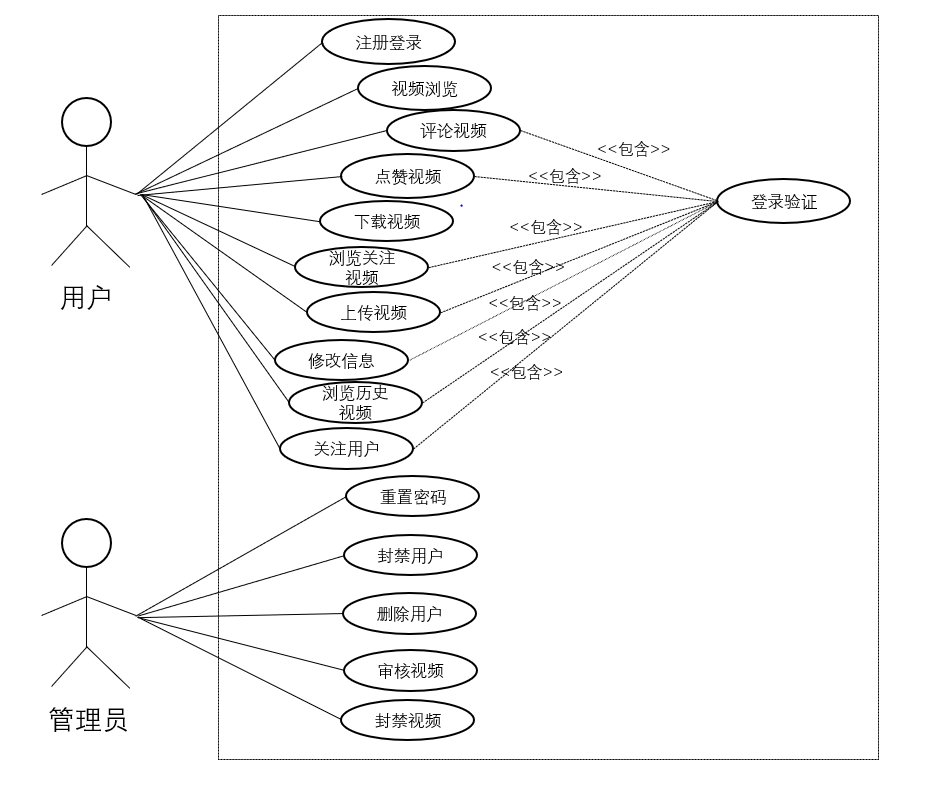


图3.1 短视频集合平台用例图

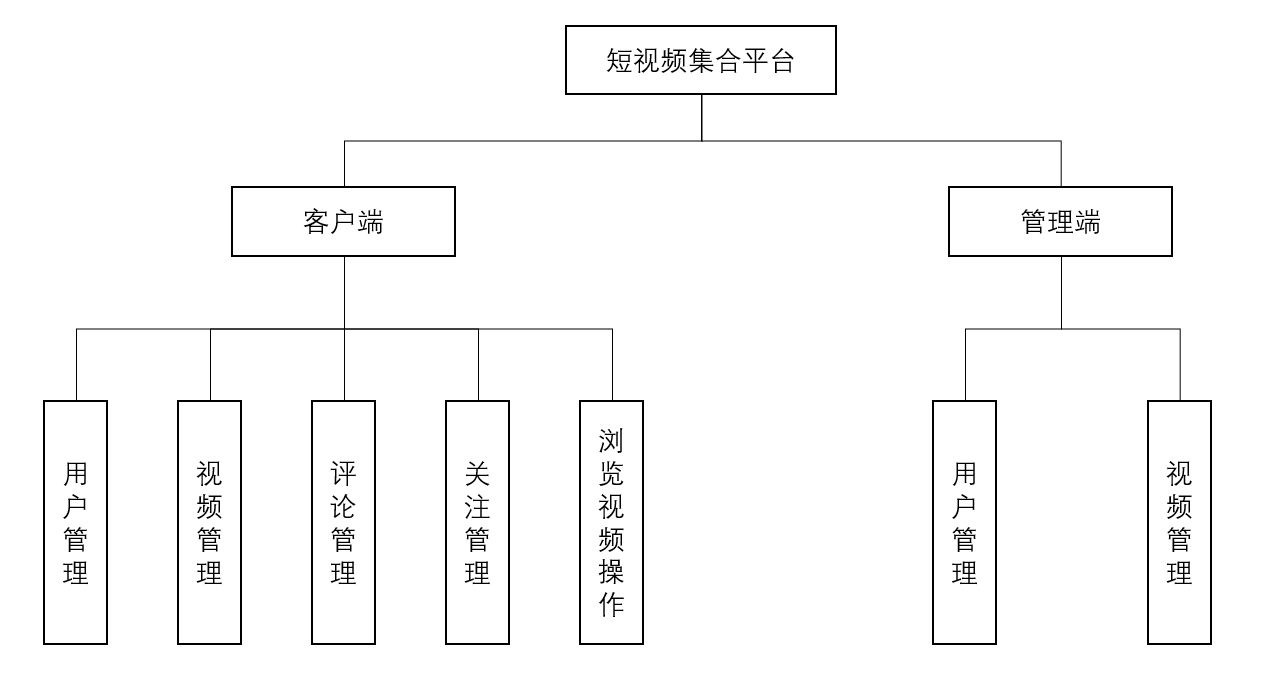


图3.2 短视频集合平台功能模块图

3.2.2 性能需求分析

客户端内部数据缓存不超过5MB，页面加载速度低于1s，视频平均加载速度低于3s，服务端接口请求的响应速度少于1s。管理端API请求的平均响应时间少于1s。

3.2.3 其他需求分析

客户端访问云服务器时，域名必须合法，保证与服务器之间交互的数据安全。保证用户在客户端发布的视频必需健康且不违法。在用户评论区域，要保证不得出现反面、侮辱和敏感等不良评论。在用户点赞区域，禁止出现恶意刷赞的行为。要营造一个安全健康的上网环境。除此之外，在服务端还应保护用户信息安全和数据库数据安全，要定时备份数据库的数据，以防止黑客、自然灾害和突发事故造成数据的丢失，导致无法挽回的可怕后果。

3.2.4 本章总结

本章主要从可行性分析和需求分析的角度对该系统进行了全面的系统分析。通过从系统的可行性分析，操作可行性分析以及社会效益角度阐述了系统的研究价值和社会意义。通过对系统的功能、性能和其他需求分析，可以初步了解到系统的大致方向。

第四章 系统概要设计

4.1 功能结构设计

该系统分为移动端的客户端，web端的管理端以及服务端。

客户端为为用户提供用户管理、视频管理等功能。系统的客户端功能结构如图4.1所示。

用户管理包括用户的注册、登录、用户关注管理、用户粉丝管理、用户视频管理等功能。视频管理包括对视频广场管理、视频的热度管理、视频的浏览、视频的发布、视频的评论点赞下载等功能。

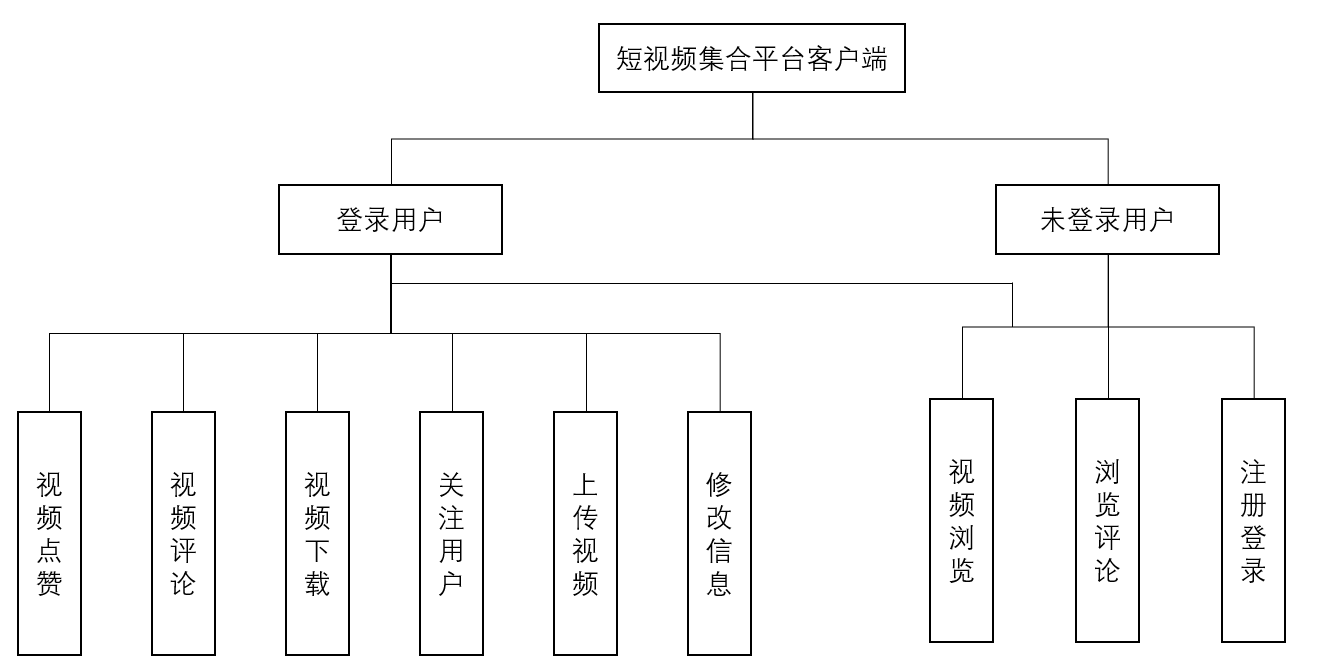
在客户端用户可分为游客和登录用户，游客指为未登录用户，仅可体验客户端部分功能，可以进行视频浏览，视频评论的浏览以及视频信息的浏览，而登录用户则可以体验客户端的所有功能。

图 4.1 短视频集合平台客户端功能结构图

在管理端，管理员可以对用户和视频进行管理。系统管理端的功能架构如图4.2所示。

管理员登录后，在视频审核模块，管理员能够对用户上传的视频进行审核，来判定视频是否违规或可发布。在用户管理模块，管理员可以浏览搜索所有的用户信息，还能对用户进行重置密码，限制用户登录、删除用户所有信息等功能。在视频管理模块，管理员可以审核视频、浏览所有的视频信息和封禁视频的操作。

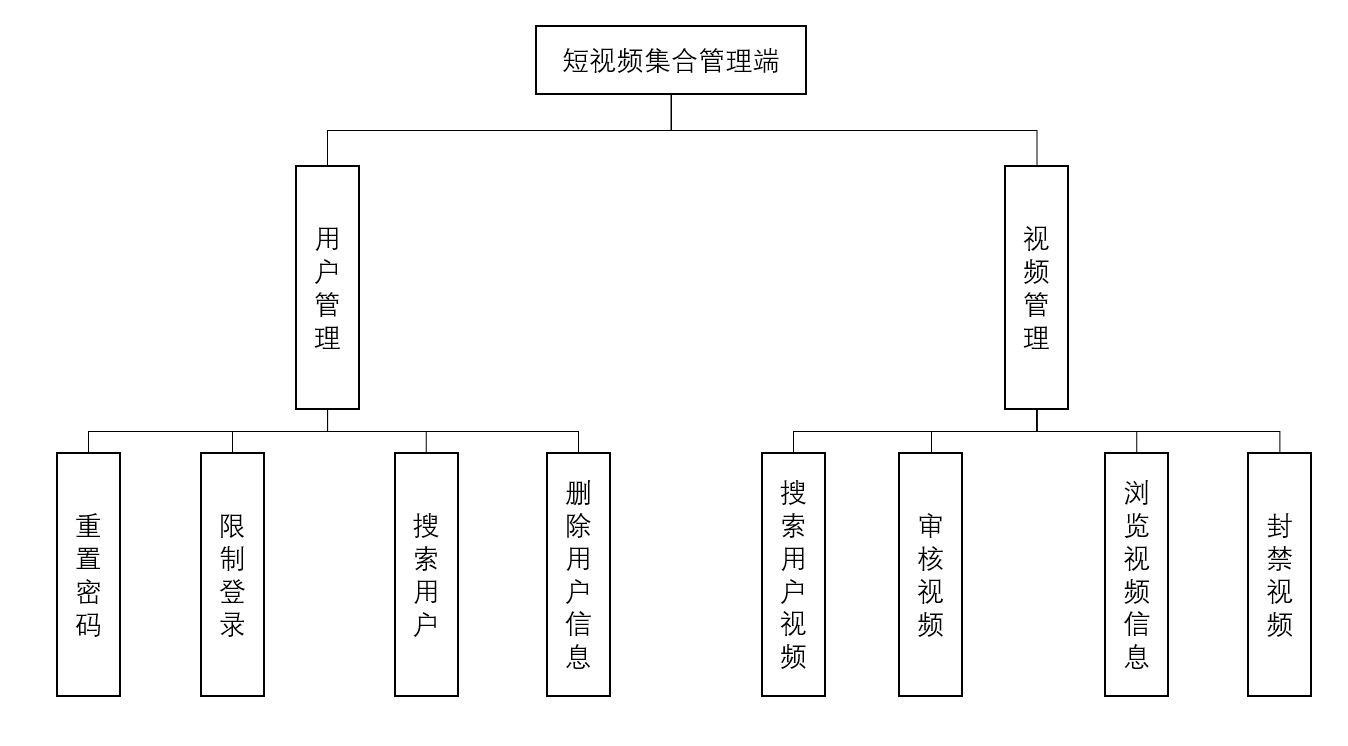


图 4.2 短视频集合平台管理端功能结构图

4.2 系统的流程设计

4.2.1 客户端的流程设计

用户可以根据自己的手机系统下载合适的安装包，安装完成后第一次进入的身份为游客身份，即未登录状态，在登录界面注册登录后，可以体验客户端的所有功能。用户登录具有时效性，后台设置为7天有效登录时间，超过7天则需要重新登录。

用户进入客户端后可以在首页的视频广场查看所有用户发布的视频，每十个视频进行了一次分页，继续下划则可以加载出之后十个视频，根据视频的封面选择自己感兴趣的视频，点击后进入视频信息的详细页面观看视频，不光如此，在视频的详细页面，用户点击爱心图标会有点赞动画特效。点击评论图标后会出现小弹窗，在小弹窗里能浏览其他用户对视频的评论，还能添加自己的评论。用户点击分享小图标后，会进行对视频的下载，下载后视频保存在相册中。点击视频作者头像后进入用户界面，能浏览作者的信息、关注或取消关注作者、浏览作者的历史发布的视频。在推荐广场上方可以选择进入视频热度广场，系统会根据用户对视频的点赞数量、评论数量、下载数量对视频进行排序，每一小时会更新一次排序结果，用户能在视频热度广场查看到最受欢迎的视频。

在关注页面上，登录用户能够查看到自己关注的用户所发布的视频

用户进入视频编辑页面后，可以选择喜爱的视频进行发布分享，在此页面，要发布的视频的必选的，用户还可自行选择视频封面和视频描述进行编辑。此系统为短视频集合平台，主要面向短视频，所以对视频的大小进行了限制，只能上传小于15MB的视频。选择号视频后点击发布按钮，会出现上传指示条，指示条完成后进入视频广场即可查看到发布的视频。

在个人信息界面，未登录用户点击头像后可进行注册登录操作，登录用户点击头像后会进入信息编辑页面，用户可编辑自己的个人信息，此信息将会展示在用户界面，供其他用户观看。除此之外，在此页面上登录用户还有我的点赞、我的视频、我的关注、我的粉丝、退出登录五个功能块。在我的点赞模块，用户点击后会进入历史点赞的页面，在次页面上用户能够看到以往点赞的视频。点击进入我的视频页后，用户能查看以往发布的视频，并拥有删除视频的权限。在进入我的关注和我的粉丝模块后，用户能查看到自己关注的用户以及关注自己的用户，并能够进入用户的主页查看用户的信息以及发布的视频。点击退出后系统户清楚所有的用户缓存信息。图4.3为短视频集合平台的客户端流程。

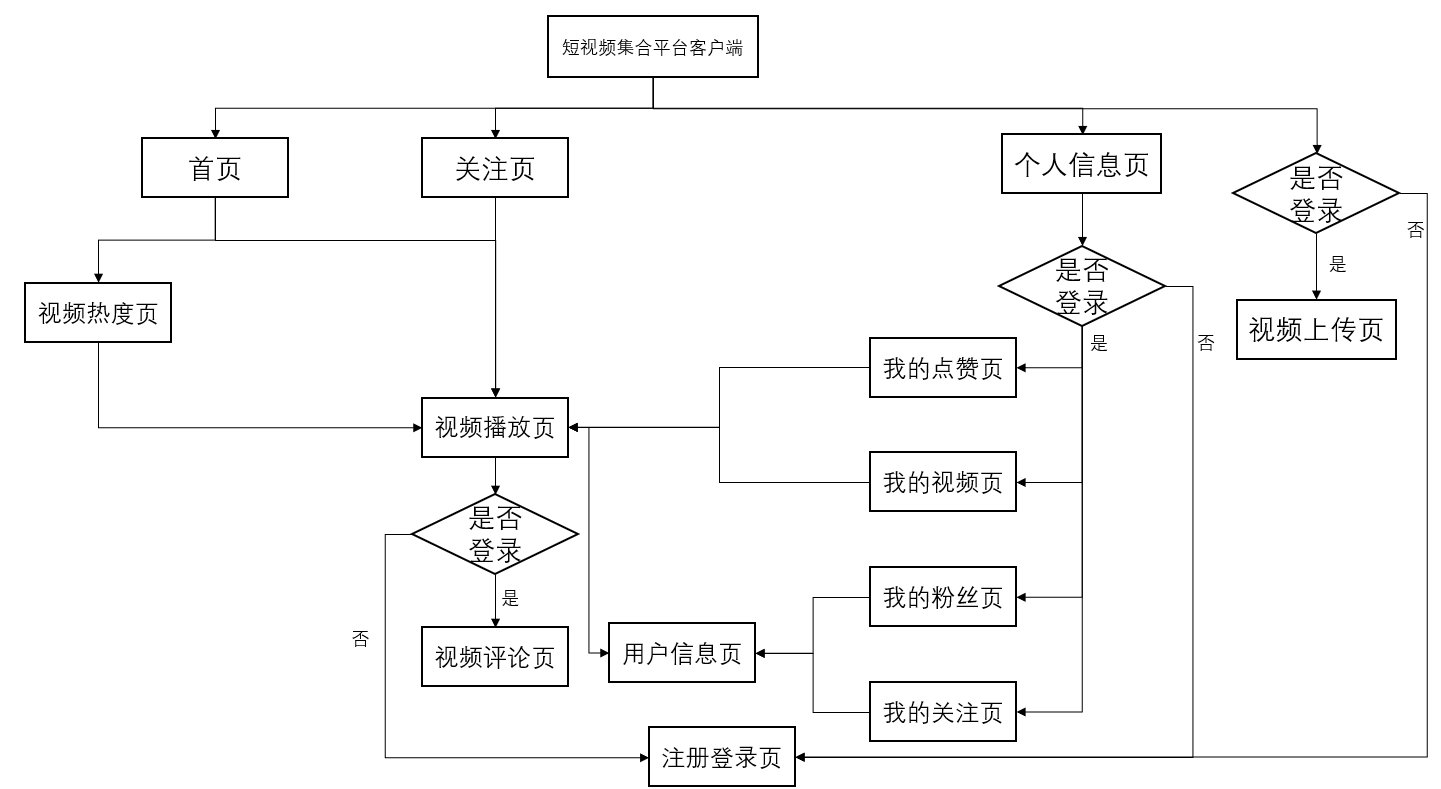


图4.3 短视频集合平台客户端系统流程

4.2.2 管理端的流程设计

在管理端，管理员用管理员账号登录后，可以对用户和用户发布的视频进行管理操作。

在管理员对视频的管理模块中，管理员能够浏览所有用户发布的视频以及视频的相关信息，并能对视频进行封禁操作。

在对用户的管理模块中，管理员能够浏览所有用户的具体信息，能够按用户名搜索用户，还能够进行用户的密码重置，限制用户的登录，删除用户及其所有的信息。

4.3 系统页面设计

本节依次从视频广场页，关注用户视频列表页，视频播放页，视频编辑页，个人中心页，用户信息页，视频播放页来介绍系统的页面设计。

对于用户来说，程序的首页时给予用户第一印象最深的页面，同时也是最为直观的感觉程序体验的好坏。所以程序的首页应该做的简洁明了，色彩柔和，不能让用户有不好的体验。首页主要展示了底部导航栏、顶部导航栏以及视频列表。视频列表块主要存放视频，视频分两列存放，在每个视频块中分上下两部分，上部分为视频封面部分，占了绝大部分，下部分块分左右两部分，左边部分为用户头像，右边部分为视频的点赞数，程序首页设计如图4.5所示。滑动顶部导航栏进入视频热度广场，主要布局和首页的视频广场相似。滑动底部的导航栏进入关注用户视频列表页，主要布局和首页视频广场相似。

点击视频列表块的视频块收进入视频播放页，在视频播放页主屏播放的时视频，在视频的左下部有个视频信息块，在视频右部有个视频互动块，如图4.6所示。在视频信息块分为上下两部分，最上部为视频作者用户名，下部为视频作者描述。在视频互动模块上下结构分为四个部分，最上方的是作者头像块，点击可进入作者主页查看用户的具体信息以及发布的视频，在头像下会根据用户是否关注了作者来判断是否显一个关注按钮，在头像块下方是一个安心点赞按钮、评论按钮、下载按钮，点击评论按钮，会出现一个小弹窗，小弹窗从底部向上弹出，占总屏幕的50%，如图4.7所示。

在视频编辑页可以选择上传的视频，也可选择封面和视频描述进行上传。如图4.8所示。

在个人中心页，点击头像可依据是否已登录来判断转跳到登录页面或者个人信息编辑页。在此页面为上下结构，上部分主要展示了用户的个人信息页面，下部分是一些用户操作，包括查看个人点赞、个人视频、个人关注、个人粉丝和退出登录，如图4.9所示。

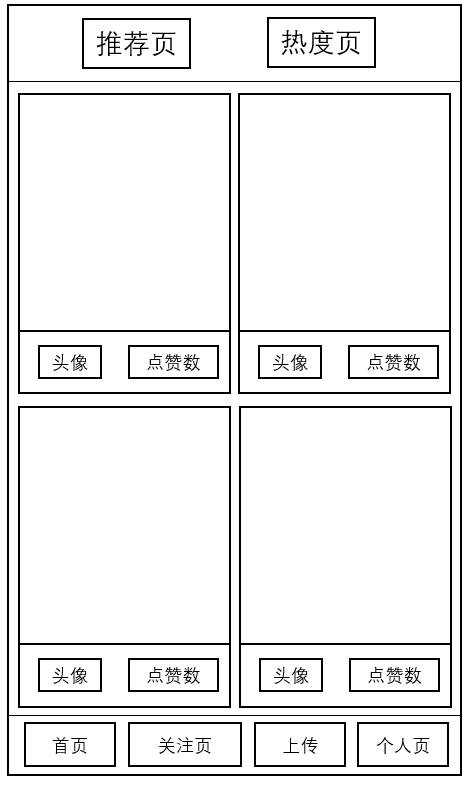


图4.5 短视频集合平台客户端首页



图4.6 短视频集合平台客户端视频播放页

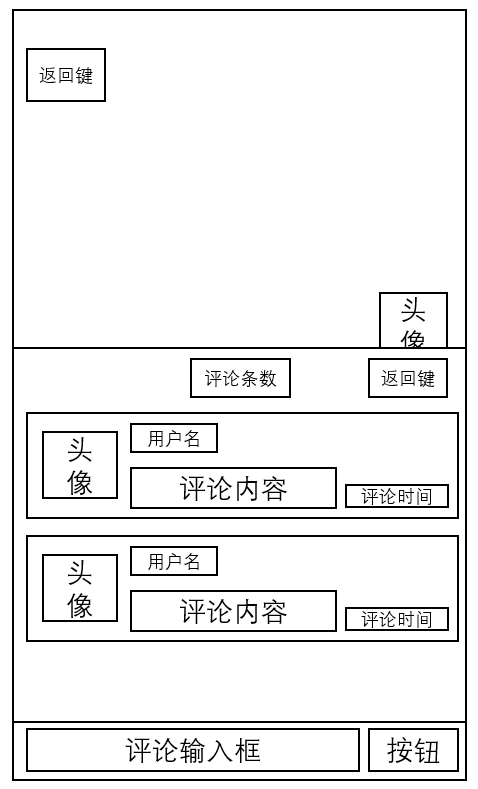


图4.7 短视频集合平台客户端视频评论页

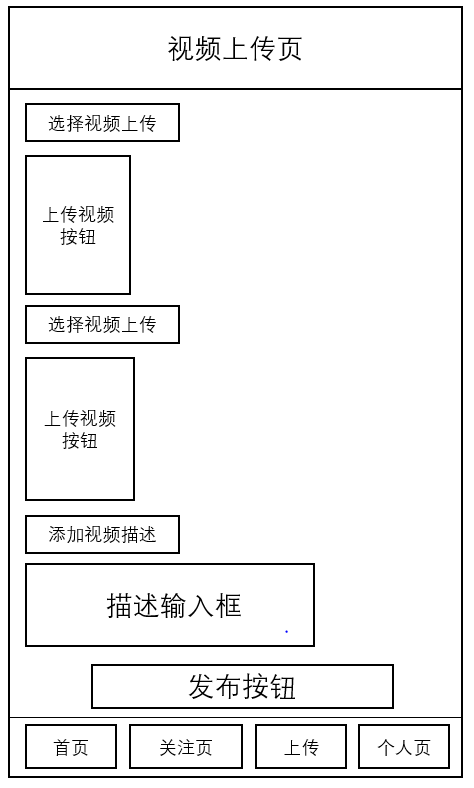


图4.8 短视频集合平台客户端视频上传页

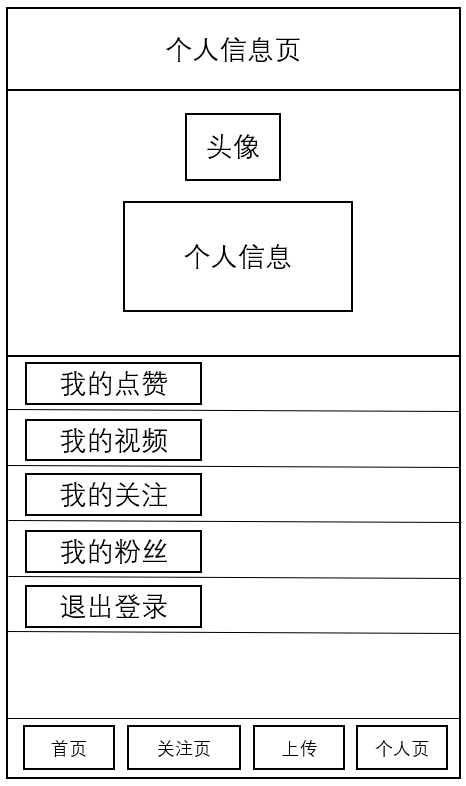


图4.9 短视频集合平台客户端个人信息页

4.4 数据库概念结构设计

短视频集合平台的后台数据设计主要包括用户表、用户详细信息表、用户关注表、用户评论信息表、视频信息表以及视频点赞表的设计。下面展示数据库概念设计的详细过程。

（1）用户这个实体主要包括用户id、用户名、用户手机号、密码、注册时间、最近更新时间、最近登录时间、最近登录设备和用户状态这些属性。图4.11为用户实体属性图。

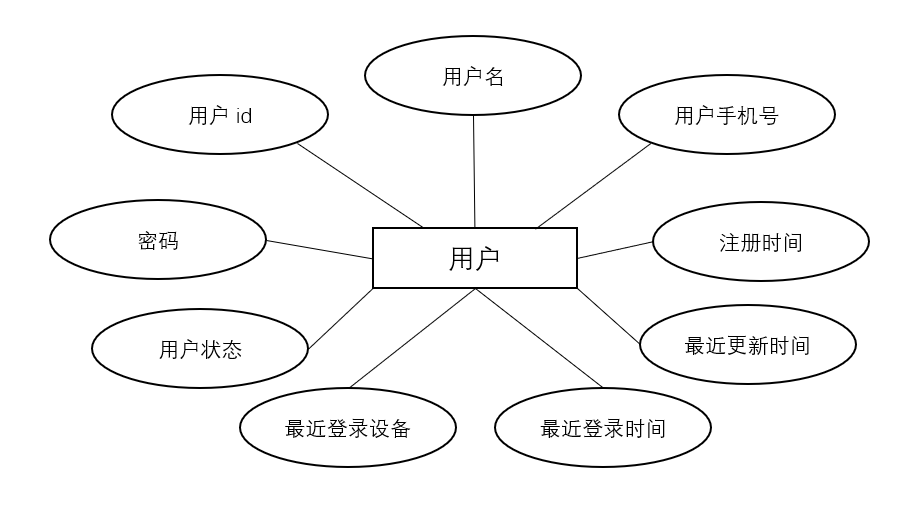


图4.11 用户实体属性图

（2）用户详细信息这个实体主要包括用户信息id、用户id、用户头像、用户名、用户性别、用户年龄、用户地址、用户简介、用户手机号、用户粉丝数以及用户关注数这些属性。图4.12为用户详细信息实体属性图。

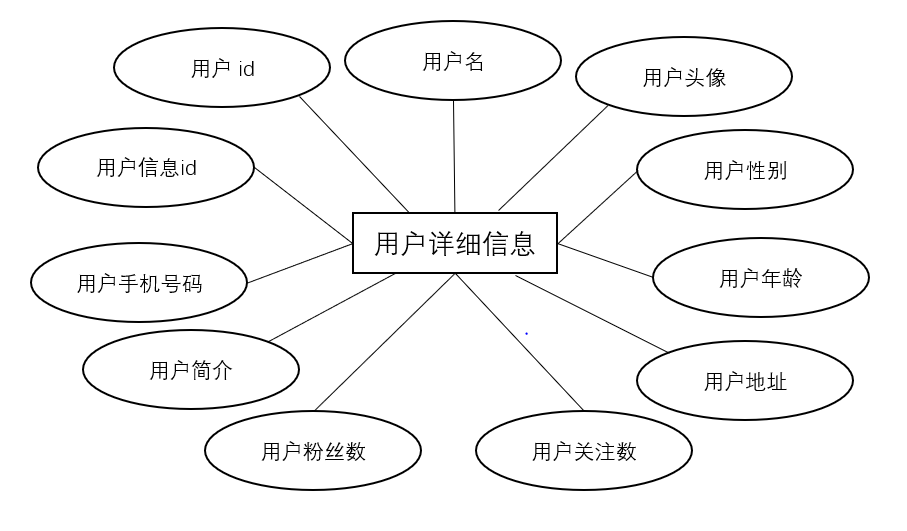


图4.12 用户详细信息实体属性图

（3）用户关注这个实体主要包括关注信息id，用户id，粉丝id，关注时间这些属性。图4.13为用户关注实体属性图。

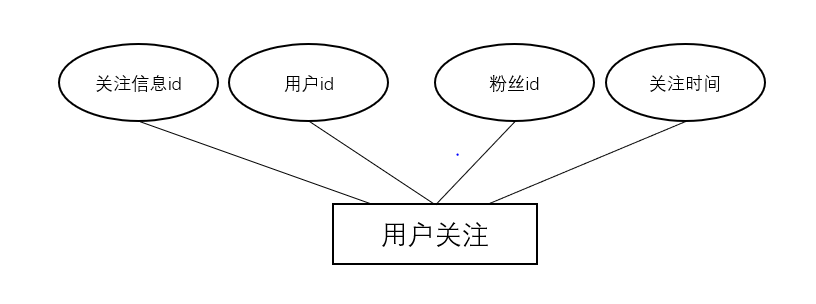


图4.13 用户关注实体属性图

（4）用户评论信息这个实体主要包括用户评论id，视频作者id，视频id，评论用户id，评论内容和评论时间这些属性。图4.14为用户评论信息实体属性图。



图4.14 用户评论信息实体属性图

（5）视频信息这个实体主要包括视频id，视频作者id，视频创建时间，视频地址，视频封面地址，视频描述，视频喜欢数，视频下载数，视频评论数，视频热度以及视频状态这些属性。图4.15为视频信息实体属性表。

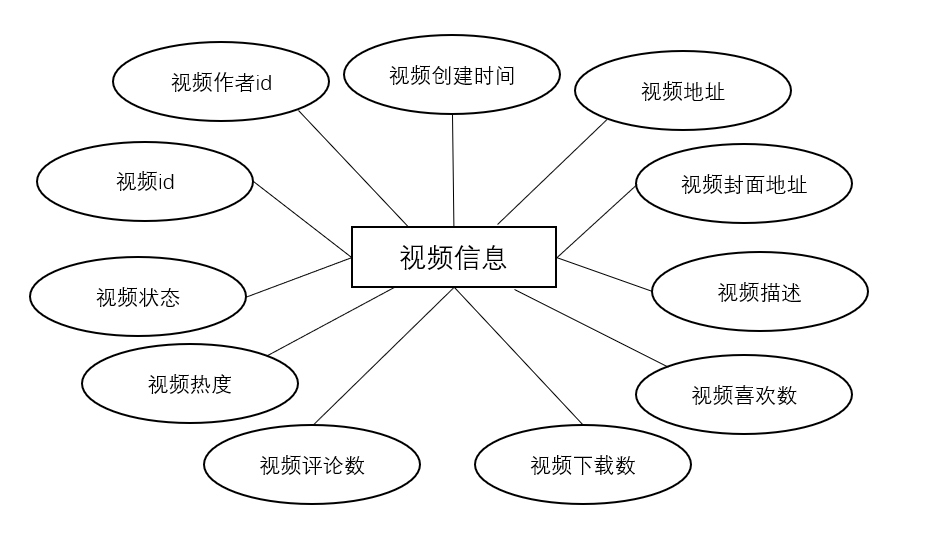


图4.15 视频信息实体属性图

（6）视频点赞信息这个属性主要包括点赞信息id，视频id，点赞用户id以及点赞时间。图4.16为视频点赞信息实体属性图。

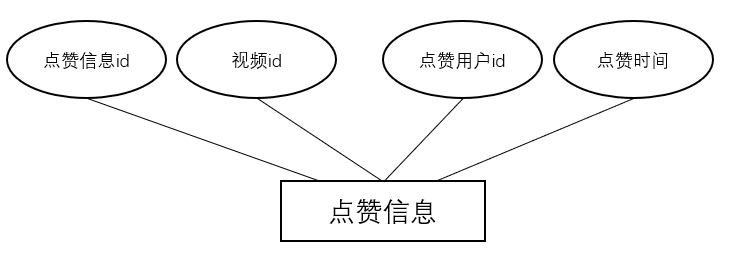


图4.17 视频点赞信息实体关系图

短视频集合平台实体联系图如图4.16所示。该图主要展示了实体之间的一对多或多对多的对应关系。

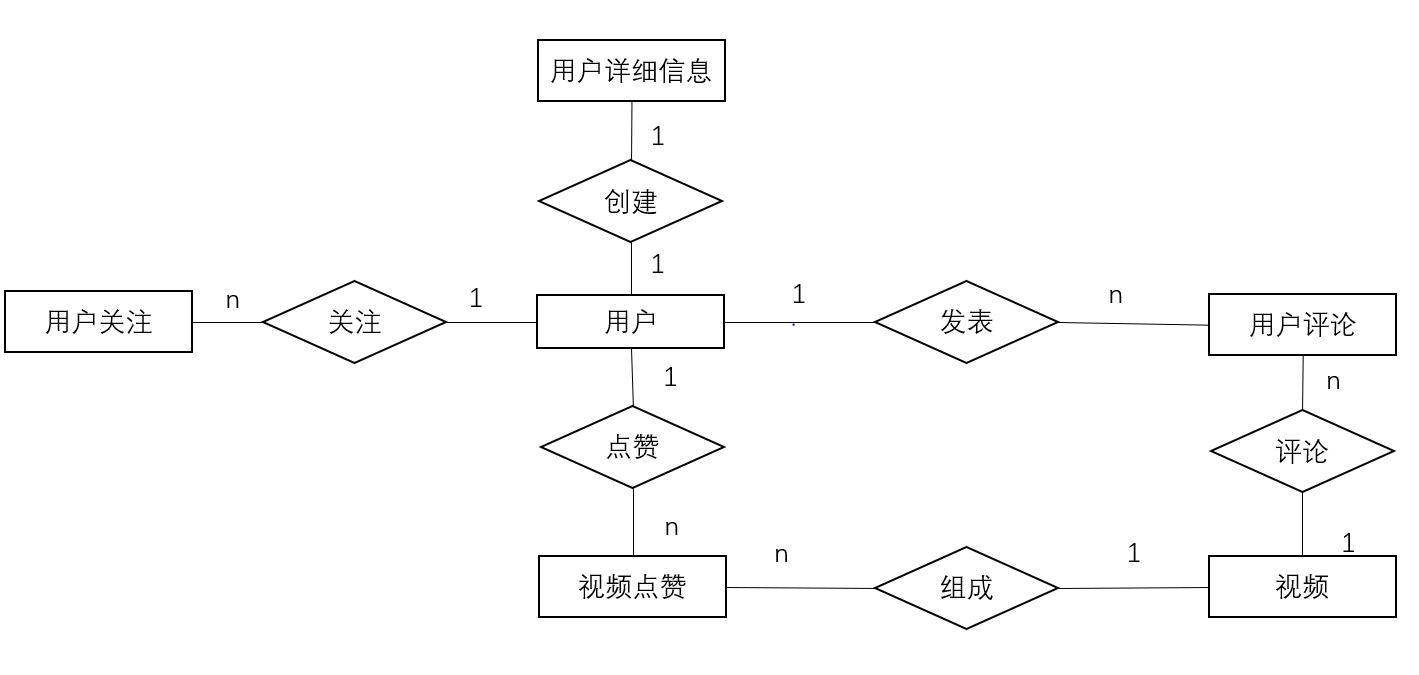


图4.16 算视频集合平台实体联系图

4.5 数据库逻辑结构设计

根据数据库结构设计所得到的E-R图转换为关系模型后，对数据库进行了逻辑结构设计。

用户（用户id，用户手机号，用户名，用户密码，注册时间，最近更新时间，最近登录时间，最近登录设备，用户状态）

此为用户实体对应的关系模式。

用户详细信息（用户信息id，用户id，用户头像，用户名，用户性别，用户年龄，用户地址，用户简介，用户手机号，用户粉丝数，用户关注数）

此为用户详细信息对应的关系模式。

用户关注（关注信息id，用户id，粉丝id，关注时间）

此为用户关注对应的关系模式。

用户评论信息（用户评论id，视频作者用户id，视频id，评论用户id，评论内容，评论时间）

此为用户评论信息对应的关系模式。

视频信息（视频id，视频作者id，视频创建时间，视频地址，视频封面地址，视频描述，视频喜欢数，视频下载数，视频评论数，视频热度，视频状态）

此为视频信息对应的关系模式。

视频点赞信息（点赞信息id，视频id，点赞用户id，点赞时间）

此为视频点赞信息对应的关系模式。

4.6 本章小结

本章主要从系统的功能结构设计、系统流程设计、系统页面设计、数据概念设计以及数据库逻辑结构设计这些方面介绍了短视频集合平台的概要设计详细过程。这一阶段主要解决了系统该如何实现，为下一阶段的详细设计与实现工作提供了实质性的指导。

第五章 系统详细设计与实现

5.1 数据库具体设计

根据上一章中的数据库概念和逻辑设计分析，可以设计出如下的数据库表，对于各表，将以字段名称、字段类型、字段说明和约束条件来展现表的结构。用户表如表5.1所示，用户详细信息表如表5.2所示，用户关注信息表如表5.3所示，用户评论信息表如表5.4所示，视频信息表如表5.5所示，视频点赞信息表如图5.6所示。

表5.1 用户表sv\_user

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 | 约束条件 |
| id | int(10) | 用户id | 主键、自动递增、非空 |
| mobile\_phone | varchar(255) | 用户手机号 | 非空 |
| user\_name | varchar(255) | 用户名 | 非空 |
| password | varchar(255) | 密码 | 非空 |
| create\_time | datetime | 注册日期 | 非空 |
| update\_time | datetime | 更新时间 |  |
| last\_login\_time | datetime | 最近登录时间 |  |
| user\_agent | varchar(255) | 使用设备 |  |
| status | int(1) | 用户状态 | 非空 |

表5.2 用户详细信息表user\_info

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 | 约束条件 |
| id | int(10) | 用户信息id | 主键、自动递增、非空 |
| user\_id | int(10) | 用户id | 非空 |
| user\_avatar | varchar(255) | 用户头像 |  |
| user\_name | varchar(255) | 用户名 | 非空 |
| sex | varchar(10) | 用户性别 |  |
| age | int(10) | 用户年龄 |  |
| area | varchar(255) | 用户地址 |  |
| introduction | varchar(1024) | 用户简介 |  |
| mobile\_phone | varchar(255) | 用户手机号码 |  |
| fans | int(10) | 用户粉丝数 |  |
| attentions | int(10) | 用户关注数 |  |

表5.3 用户关注表attention\_info

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 | 约束条件 |
| id | int(10) | 关注信息id | 主键、自动递增、非空 |
| user\_id | int(10) | 用户id | 非空 |
| fans\_id | int(10) | 粉丝id | 非空 |
| create\_time | datetime | 关注时间 | 非空 |

表5.4 用户评论信息表reply

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 | 约束条件 |
| id | int(10) | 用户评论id | 主键、自动递增、非空 |
| video\_author\_id | int(10) | 视频作者id | 非空 |
| reply\_video\_id | int(10) | 视频id | 非空 |
| reply\_maker\_id | int(10) | 评论用户id | 非空 |
| reply\_maker\_content | varchar(1024) | 评论内容 | 非空 |
| reply\_time | datetime | 评论时间 | 非空 |

表5.5 视频信息表 video

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 | 约束条件 |
| id | int(10) | 视频id | 主键、自动递增、非空 |
| author\_id | int(10) | 视频作者id | 非空 |
| create\_time | datetime | 创建时间 |  |
| url | varchar(1024) | 视频地址 | 非空 |
| cover | varchar(1024) | 封面地址 |  |
| description | varchar(255) | 视频描述 |  |
| likes | int(10) | 视频喜欢数 |  |
| downloads | int(10) | 视频下载数 |  |
| comments | int(10) | 视频评论数 |  |
| hot | int(10) | 视频热度 |  |
| status | int(1) | 视频状态 | 非空 |

表5.6 视频点赞信息表 video\_liker

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 | 约束条件 |
| id | int(10) | 点赞信息id | 主键、自动递增、非空 |
| video\_id | int(10) | 视频id | 非空 |
| liker | int(10) | 点赞用户id | 非空 |
| create\_time | datetime | 点赞时间 | 非空 |

本系统数据库设计使用了Navicat Premium可视化数据库设计管理更具，可以简化对数据库的操作过程。数据库的关系模型如图5.1所示。

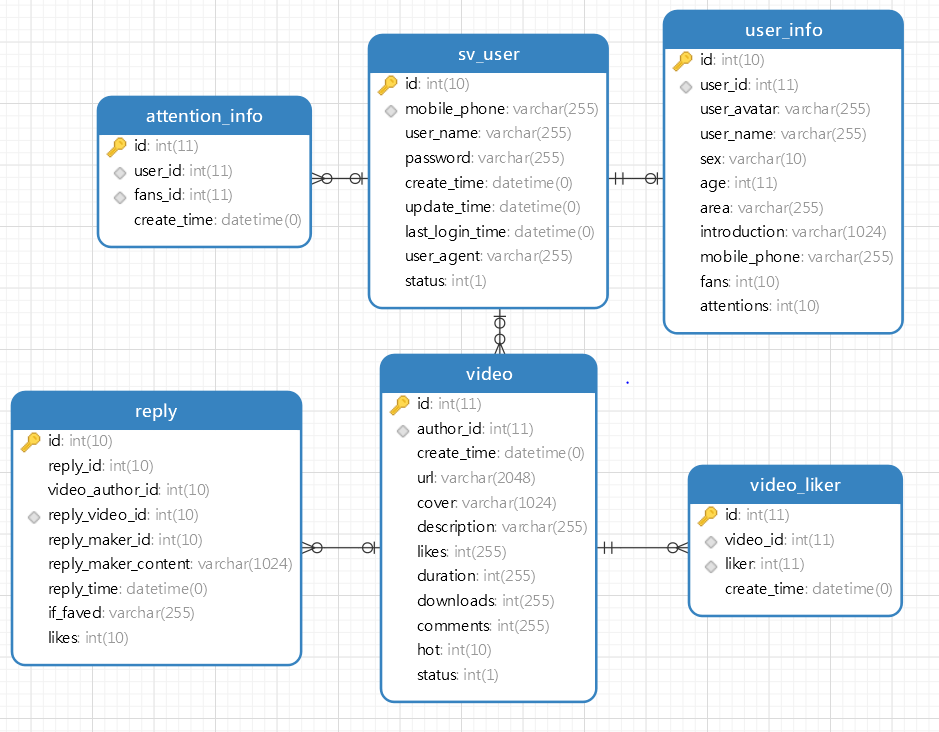


图5.1 数据库模型图

如上图所示，数据库中用户表、用户信息表以及用户关注表之间是通过用户id进行关联的。视频表、回复表以及视频点赞表之间是通过视频id进行关联的。用户表和视频表之间是同用户id进行关联的。

5.2 用户管理模块

5.2.1 模块简介

用户管理模块主要包括客户端用户的注册登录、用户信息的修改以及管理端管理员对用户信息的搜索用户、导出用户信息、重置密码、限制登录、删除用户功能。

客户端用户首次进入短视频集合平台客户端时时处于未登录的状态。用户可以点击个人中心的页面的头像进行注册登录。用户在输入账号密码后，经过后台服务端的业务处理成功后会生成一个身份认证的token并返回给客户端，从而完成用户登录。此token将用于后续业务逻辑中与服务端交互时识别用户的身份。这个token在服务端保存在redis中，保存时间为7天，即7天后需要该token将会失效，在此之后，客户端再与服务端发送请求后，服务端将无法识别用户身份，此时用户将无法使用客户端会提示用户进行重新登录操作。对于时效内token完整的用户，客户端每次对服务端发送请求时，都会在header附带token，服务端会对token进行校验，如果校验成功，则对数据库操作，返回请求的数据，如果校验失败，则返回错误码403。在此时点击用户头像则可进入用户信息的修改界面。

管理端的管理员登录机制如客户端。此外管理员能重置用户密码，重置后密码变为123456。在管理端限制用户登录后，用户将无法再登录客户端。删除用户后将删除用户的一切信息，包括用户的视频、评论、关注信息等所有信息。

5.2.2 身份认证功能

短视频集合平台的客户端登录界面如图5.2所示。

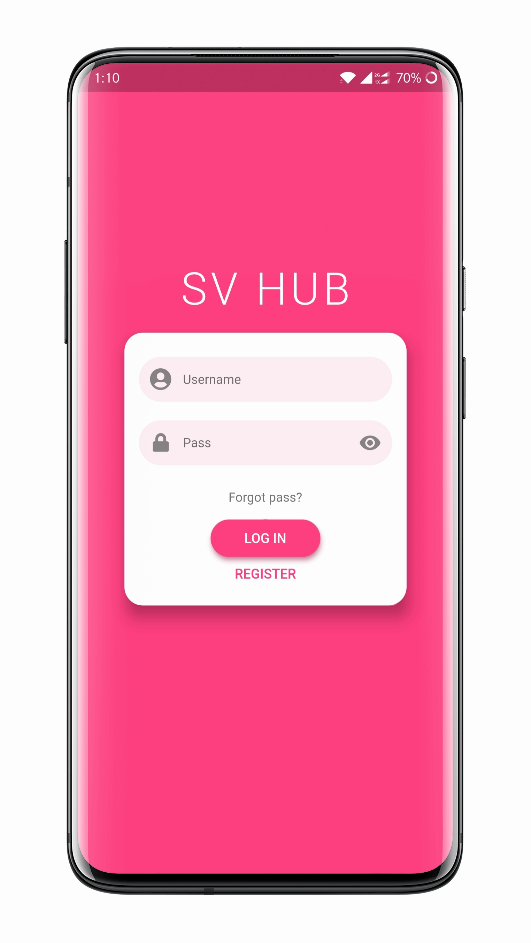


图5.2 短视频集合平台客户端登录界面

短视频集合平台服务端登录界面如图5.3所示。

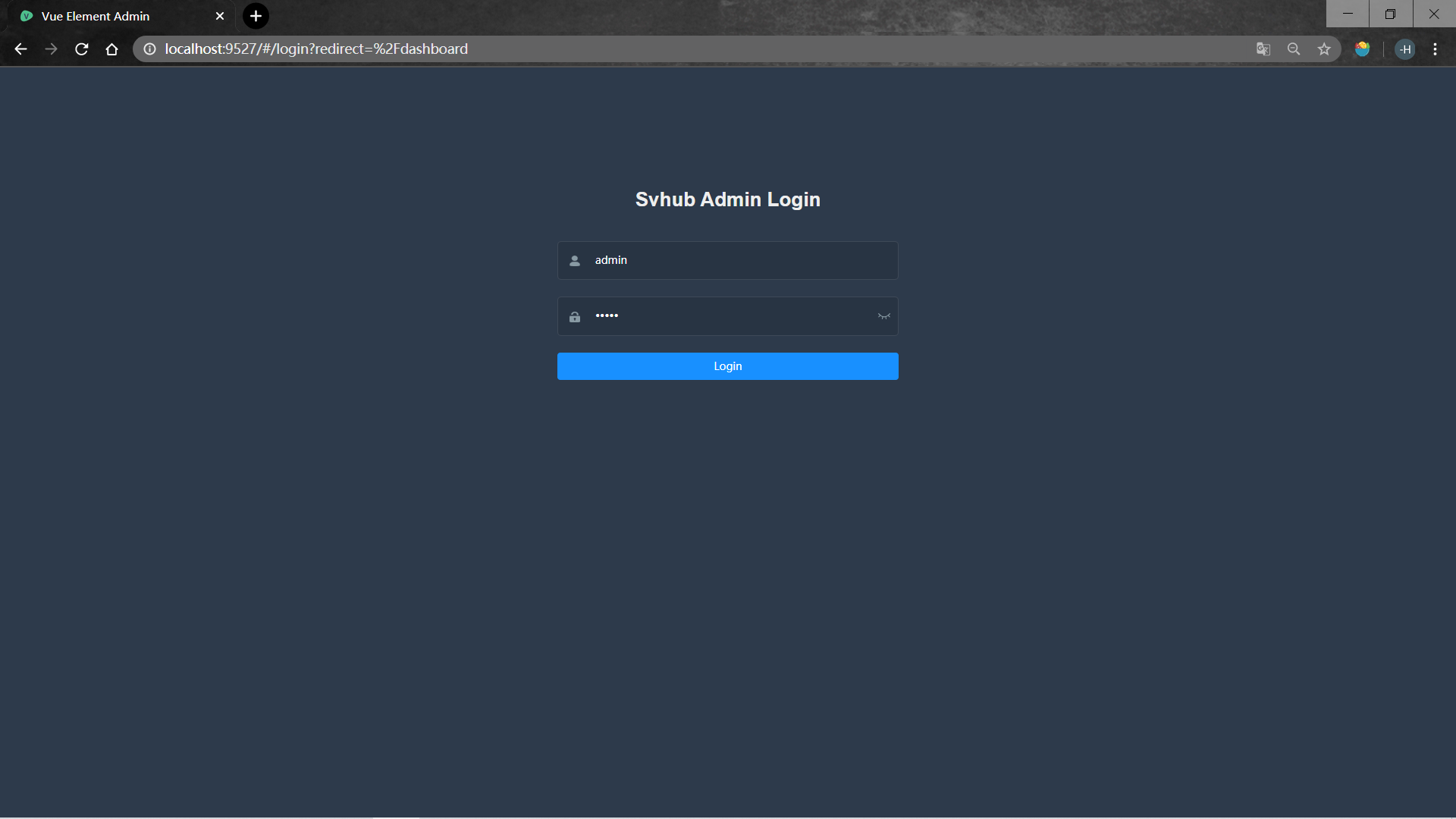


图5.3 短视频集合平台管理端端登录界面

对于客户端的登录，如图5.4所示。

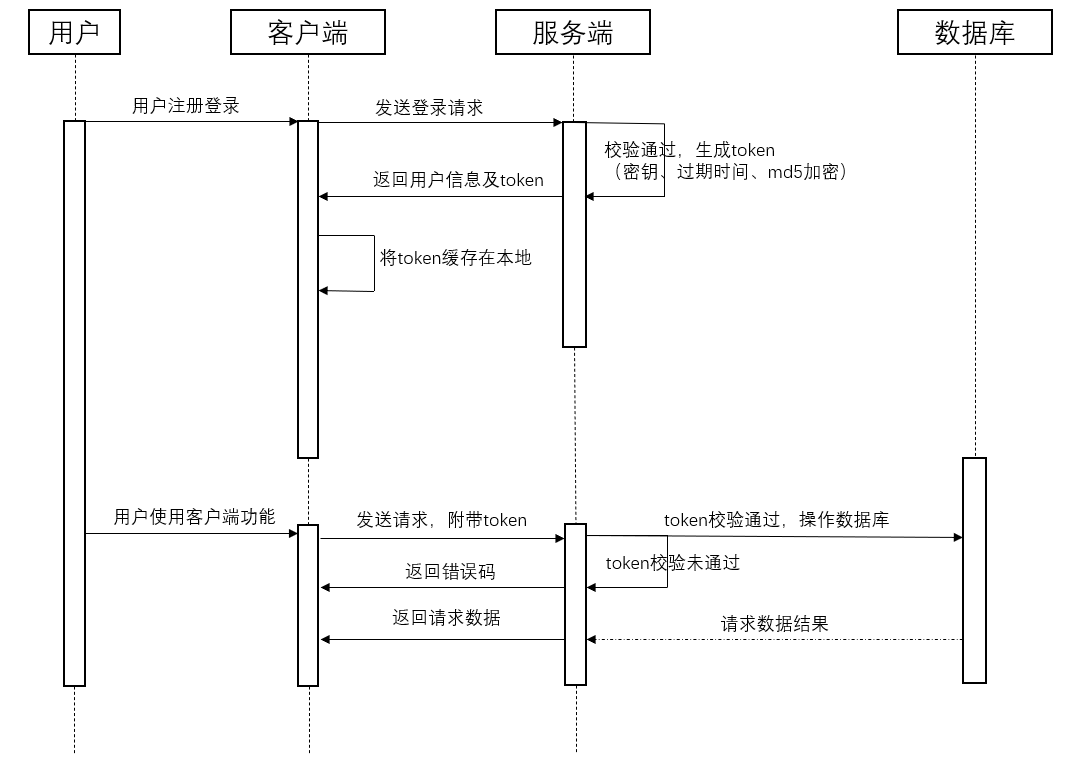


图5.4 管理端用户登录顺序图

客户端登录请求通过封装token，核心代码如下：

Future<String> \_loginUser(LoginData loginData) async {

Map<String, dynamic> data = {

"mobilePhone": loginData.name,

//md5加密用户密码

"password": TsUtils.generateMd5(loginData.password)

};

Result result = await DioRequest.dioPost(URL.USER\_LOGIN, data);

return Future.delayed(loginTime).then((\_) async {

if (result.code == 1) {

loginSuccess (result);

TsUtils.showShort("登录成功");

Navigator.pop(context);

return null;

} else {

return 'Username or password is incorrect or login is not allowed';

}

});

}

其中loginSuccess()方法：

void loginSuccess(Result result) async{

UserInfo userInfo = await UserInfoUtil.map2UserInfo(result.data);

Map<String, dynamic> data = {

"access\_token": userInfo.accessToken

};

//保存token

UserUntil.saveToke(data);

User user = User(userInfo.userId, userInfo.userName,

userInfo.userAvatar, userInfo.mobilePhone);

OsApplication.eventBus.fire(LoginEvent(

userInfo.userId,

user.userName,

user.userAvatar,

userInfo.age,

userInfo.sex,

userInfo.area,

userInfo.introduction,));

//保存用户信息

UserUntil.saveUserInfo(user);

UserInfoUtil.saveUserInfo(userInfo);

}

服务端接收到客户端的登录请求后，会先判断账号密码的是否正确，如果正确，则会生成一个token，并且存储在redis中，之后token会连同用户数据一起返回给客户端，其核心代码如下：

private String loginSuccess(UserInfo userInfo) {

HttpServletRequest request = ((ServletRequestAttributes)RequestContextHolder.getRequestAttributes()).getRequest();

String userAgent = request.getHeader("User-Agent");

User user1 = new User();

user1.setLastLoginTime(new Date());

user1.setId(userInfo.getId());

user1.setUserAgent(userAgent);

user1.setUpdateTime(new Date());

userMapper.updateUser(user1);

String loginToken = "";

loginToken += DigestUtils.md5DigestAsHex(UUID.randomUUID().toString().getBytes());

Object o = redisService.get(userInfo.getUserId().toString());

//检查重复登录

UserInfo logined = new UserInfo();

if(o != null) {

log.info("重复登录，前者登出");

logined = JSONObject.parseObject(JSONObject.parse(JSONObject.toJSONString(o)).toString(), UserInfo.class);

String token = logined.getAccessToken();

if(!StringUtils.isEmpty(token)) {

redisService.remove(token);

redisService.remove(userInfo.getUserId().toString());

}

}

//登录缓存

userInfo.setAccessToken(loginToken);

redisService.set(loginToken, JSONObject.toJSONString(userInfo), 7L, TimeUnit.DAYS);

redisService.set(userInfo.getUserId().toString(), JSONObject.toJSONString(userInfo), 7L, TimeUnit.DAYS);

return loginToken;

}

管理端管理员的登录身份认证机制与客户端的类似。

5.2.3 客户端用户信息修改

当用户注册完账号第一次登录后，再次点击头像会进入个人信息编辑页，如图5.5所示，图5.6为信息页面展示效果。在此页面，进入后会在文本框显示原本的信息，用户可以修改自己在主页的展示信息，以此可以吸引更多的粉丝。修改完信息后点击确认修改按钮，客户端会携带修改好后的信息像服务端发送request请求。管理端接收到请求参数后会对数据库中用户的信息进行修改，其核心代码如下：

<!-- 用户信息更新 -->

<update id="updateUserInfo">

update user\_info

<set>

<if test="userInfo.userAvatar != null and userInfo.userAvatar != ''"> user\_avatar = #{userInfo.userAvatar}, </if>

user\_name = #{userInfo.userName},sex = #{userInfo.sex},age = #{userInfo.age},

area = #{userInfo.area},introduction = #{userInfo.introduction},

</set>

<where>

user\_id = #{userInfo.userId};

</where>

</update>



图5.5 客户端用户信息修改界面



图5.6 客户端用户信息展示页面

5.2.4 管理端的用户管理

在管理端，管理员拥有对用户的操作权限，包括搜索用户、导出用户信息、重置密码、限制登录、删除用户功能。管理端用户管理界面如图5.7所示。

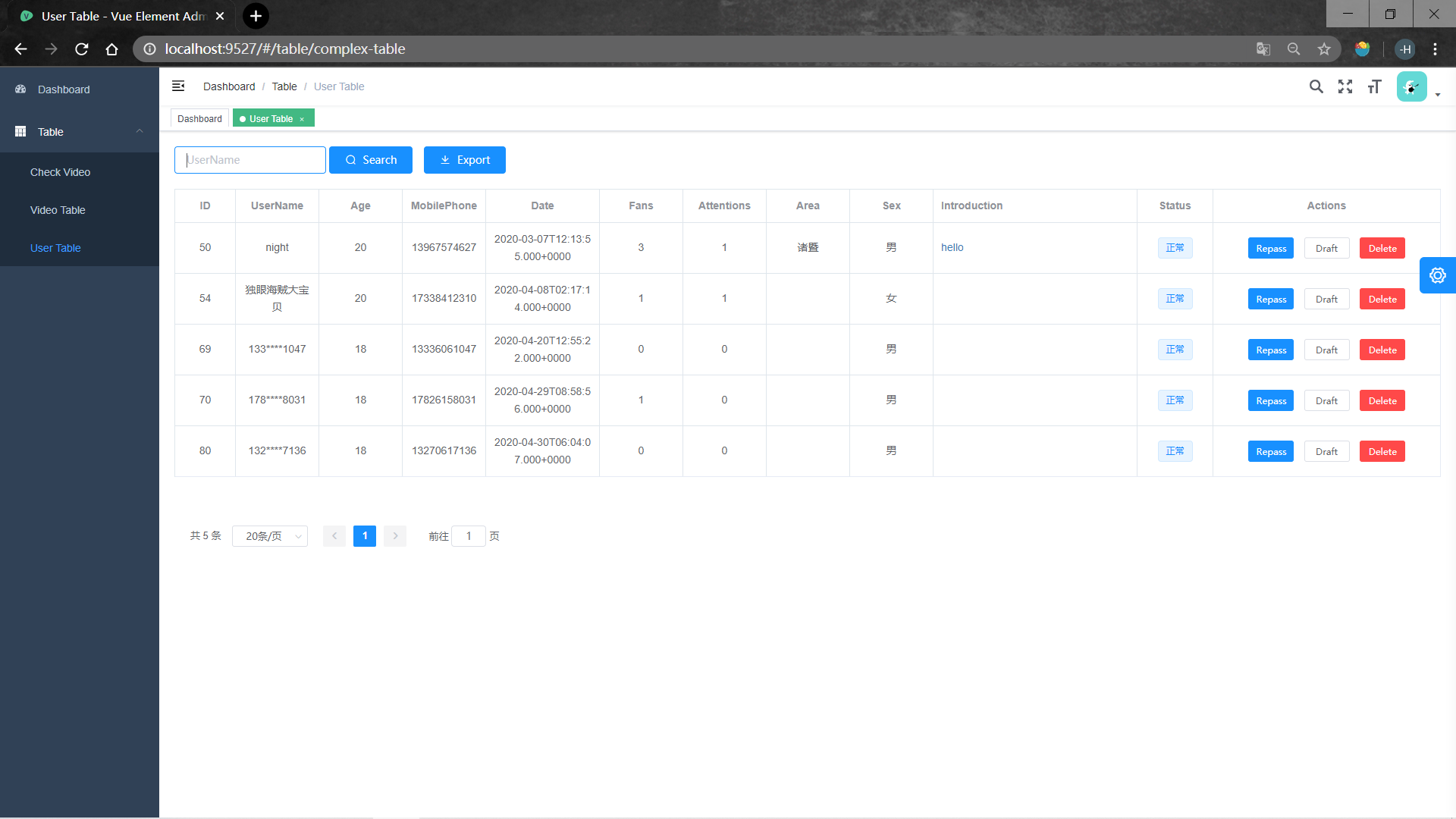


图5.7 管理端用户管理界

（1） 用户搜索功能主要是先获取管理员输入的关键字，然后携带此关键字向后台发送request请求。服务端认证管理员权限后会获取到搜索关键字这个参数后，会去向数据库进行模糊查询。之后会将查询到的结果分页后返回给管理端页面进行显示，图5.8为搜索后结果，其核心代码如下：

<!-- 取得用户列表 -->

<select id="getUserList" resultType="com.zzh.shortvideohub.pojo.UserInfo">

select ui.user\_name userName, ui.user\_id userId, ui.age age, ui.mobile\_phone mobilePhone,

ui.user\_avatar userAvatar, ui.sex sex, ui.area area, ui.introduction introduction,

ui.fans fans, ui.attentions attentions, su.status status, su.create\_time createTime

from sv\_user su

left join user\_info ui on su.id = ui.user\_id

where su.status &lt; 2

<if test="userInfo.userName != null and userInfo.userName != ''">

and ui.user\_name like concat('%', #{userInfo.userName}, '%')

</if>

</select>



图5.8 搜索后管理端页面图

(2)用户信息导出功能会导出目前页面所显示的用户信息，其操作在js中完成，不与服务端产生交互。点击此按钮后会获取列表中的各个属性，再根据设置好的映射条件映射到excel表中，导出后excel表如图5.9所示。其核心js代码如下：

handleDownload() {

this.downloadLoading = true

import('@/vendor/Export2Excel').then(excel => {

const tHeader = ['UserId', 'UserName', 'Age', 'MobilePhone', 'Fasn', 'Attentions', 'Area', 'Sex', 'Introduction', 'Status']

const filterVal = ['UserId', 'UserName', 'Age', 'MobilePhone', 'Fasn', 'Attentions', 'Area', 'Sex', 'Introduction', 'Status']

const data = this.formatJson(filterVal)

excel.export\_json\_to\_excel({

header: tHeader,

data,

filename: 'table-list'

})

this.downloadLoading = false

})

}

export function export\_json\_to\_excel({

multiHeader = [],

header,

data,

filename,

merges = [],

autoWidth = true,

bookType = 'xlsx'

} = {}) {

/\* original data \*/

filename = filename || 'excel-list'

data = [...data]

data.unshift(header);

for (let i = multiHeader.length - 1; i > -1; i--) {

data.unshift(multiHeader[i])

}

var ws\_name = "SheetJS";

var wb = new Workbook(),

ws = sheet\_from\_array\_of\_arrays(data);

if (merges.length > 0) {

if (!ws['!merges']) ws['!merges'] = [];

merges.forEach(item => {

ws['!merges'].push(XLSX.utils.decode\_range(item))

})

}

if (autoWidth) {

/\*设置worksheet每列的最大宽度\*/

const colWidth = data.map(row => row.map(val => {

/\*先判断是否为null/undefined\*/

if (val == null) {

return {

'wch': 10

};

}

/\*再判断是否为中文\*/

else if (val.toString().charCodeAt(0) > 255) {

return {

'wch': val.toString().length \* 2

};

} else {

return {

'wch': val.toString().length

};

}

}))

/\*以第一行为初始值\*/

let result = colWidth[0];

for (let i = 1; i < colWidth.length; i++) {

for (let j = 0; j < colWidth[i].length; j++) {

if (result[j]['wch'] < colWidth[i][j]['wch']) {

result[j]['wch'] = colWidth[i][j]['wch'];

}

}

}

ws['!cols'] = result;

}

/\* add worksheet to workbook \*/

wb.SheetNames.push(ws\_name);

wb.Sheets[ws\_name] = ws;

var wbout = XLSX.write(wb, {

bookType: bookType,

bookSST: false,

type: 'binary'

});

saveAs(new Blob([s2ab(wbout)], {

type: "application/octet-stream"

}), `${filename}.${bookType}`);

}

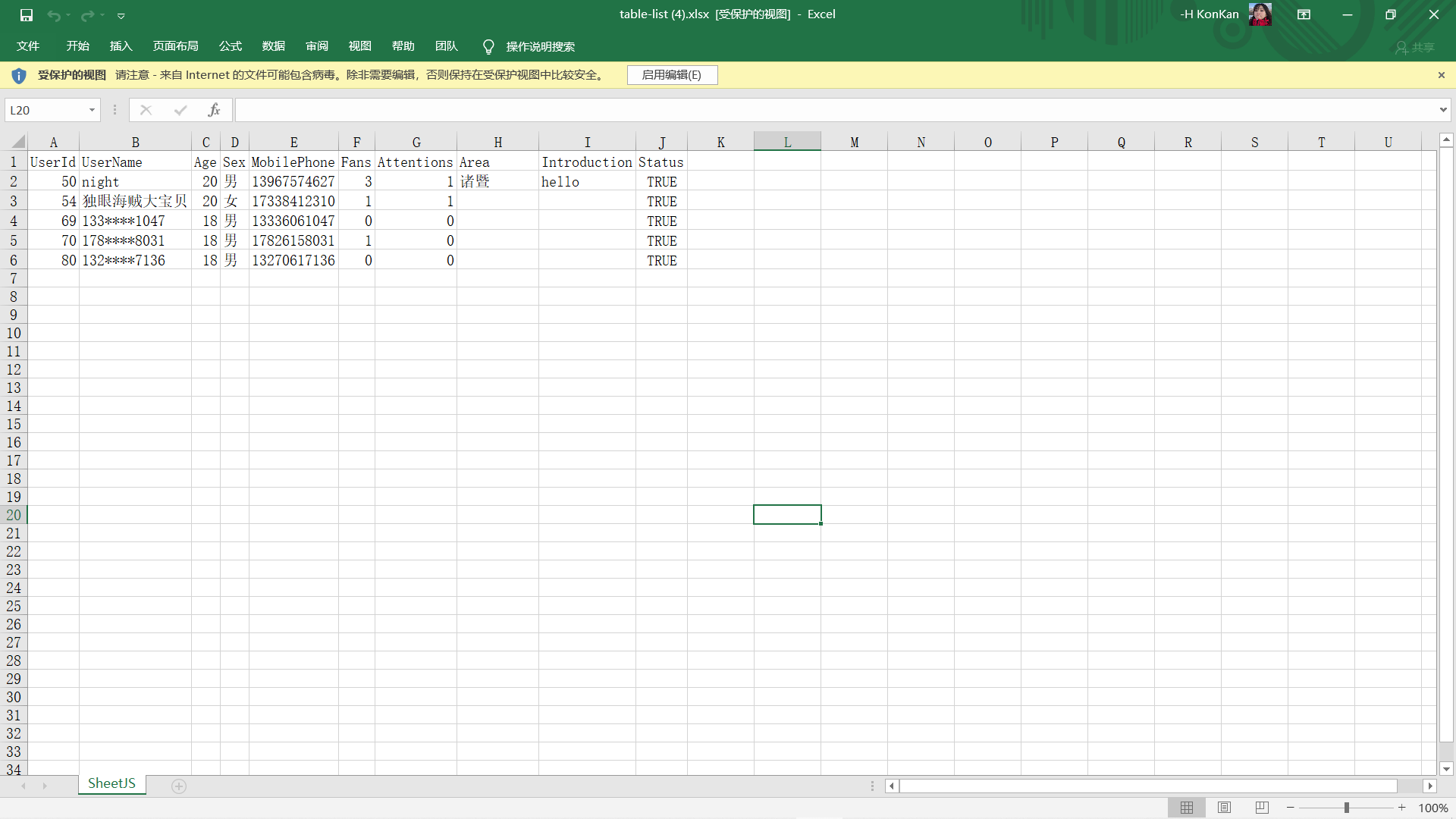


图5.9 用户信息导出图

（3）用户密码重置后会将密码重置成123456，初始密码配置在yml中。点击此按钮后管理端会像服务端发送request请求，服务端认证到管理员权限后会先获取到在配置文件中的初始密码，然后会对初始密码进行MD5加密，之后再对数据库进行操作。其加密核心代码如下：

static public String md5Encryption(String password) throws NoSuchAlgorithmException {

MessageDigest md5 = MessageDigest.getInstance("md5");

md5.update(password.getBytes());

String result = new BigInteger(1, md5.digest()).toString(16);

return result;

}

（4）管理员限制用户登录后回对服务端发送request请求，服务端认证管理员身份后允许请求改变用户的状态，状态由原本的1正常状态变为0限制登录状态。改变用户状态后用户登录情况如图5.10所示，管理端显示状态如图5.11所示。

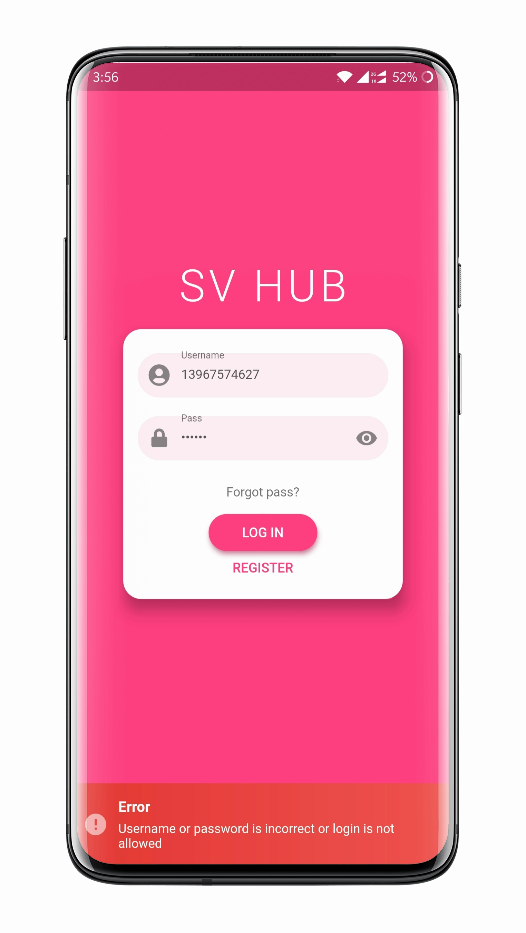


图5.10 限制用户后登录图

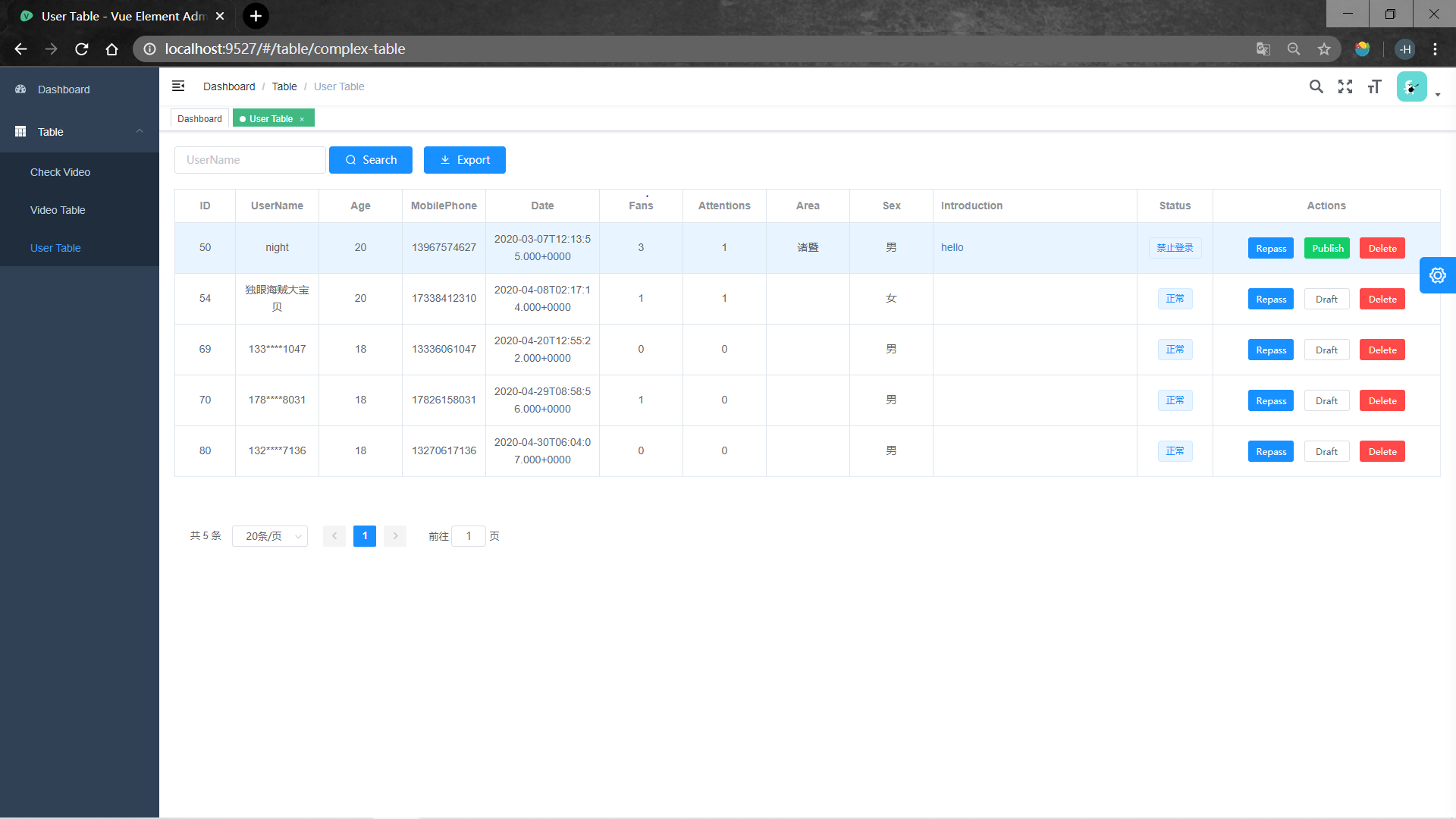


图5.11 限制用户后显示状态

（5）管理员删除用户后会向服务端发送request请求，服务端认证管理员身份后会允许管理员的操作，删除关于用户的一切信息包括删除用户信息，删除用户视频，删除用户评论，删除用户关注及粉丝。

5.3 视频管理模块

5.3.1 视频管理模块主要包含客户端的视频浏览、视频播放、点赞视频、评论视频、下载视频、上传视频、删除视频以及管理端的审核视频、搜索用户视频、封禁用户视频。

（1）在客户端的主页面是视频广场，如图5.12所示。此页初始化时回想服务端发送请求，请求获取最新的10个视频，获取成功后存储在每个视频块中，每个视频块都存储了一个视频的信息，如视频地址、视频作者用户名、视频封面地址、视频作者头像地址、视频发布时间、视频描述、视频点赞量、视频评论数以及视频下载量，在视频块中可以看到视频封面、视频作者头像以及视频点赞数。往上滑则会刷新该页面。往下滑会向服务端发送请求，请求获取下个10个的最新视频。

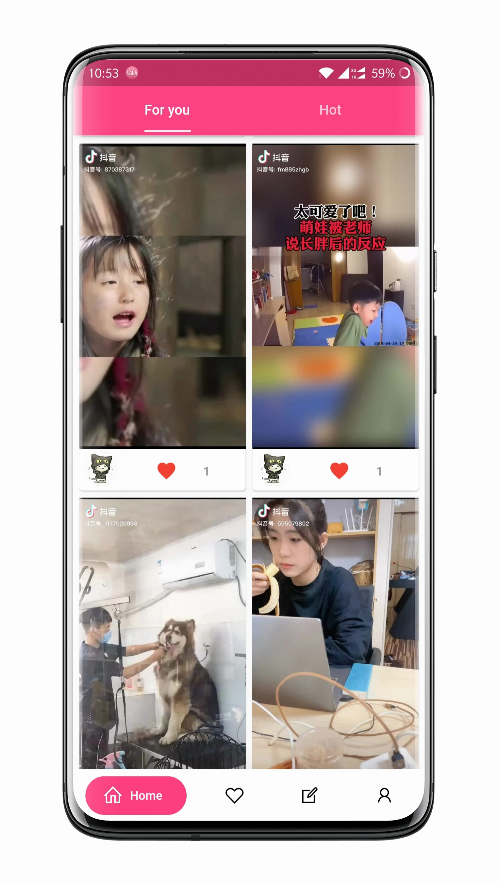


图5.12 客户端视频广场页

（2）点击视频块后进入视频播放页，如图5.13所示，此时客户端会像服务端发送request请求，请求得到更详细的数据，如当前用户是否关注了视频作者，当前用户是否点赞了该视频，客户端会根据以上条件调整页面布局。若当前用户已关注了视频作者，头像下的关注按钮不会显示关注按钮。若当前用户之前已经点赞了该视频，点赞按钮的初始化是已点赞状态。

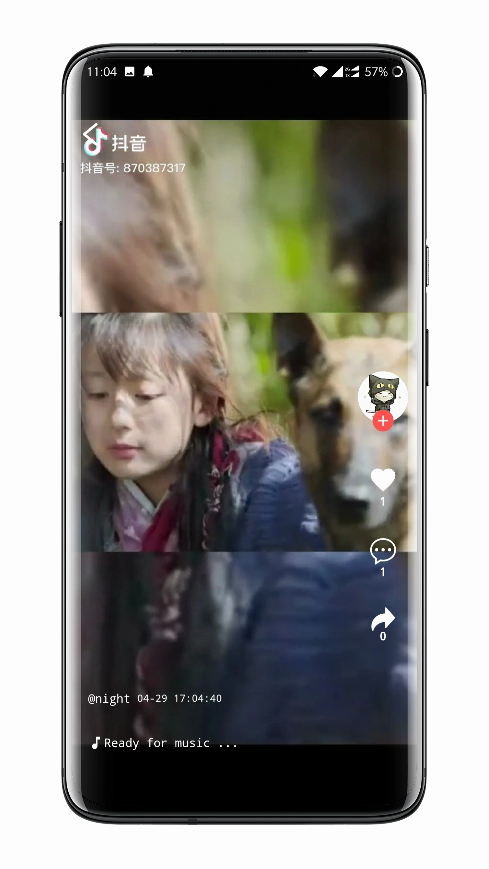


图5.13 客户端视频播放页

（3）在视频播放页面点击点赞按钮后，会出现一个点赞动画，同时会向服务端发送request请求，服务端取得request里的数据，在数据库中增加用户的点赞信息。点赞成功会爱心按钮编程红色，同时点赞数加1。

（4）在视频播放页面点击评论图标进入评论页。点击评论图标后客户端向服务端发送了请求，请求取得视频的评论，客户端取得评论列表后在评论页面展示评论列表，如图5.14，所示。同时点击评论框用户输入评论并点击发布按钮后，客户端会向服务端发送请求，服务端认证用户信息后将用户的评论添加进数据库中。



图5.14 客户端视频评论页

（5）在视频播放页的下载按钮，点击后客户端会根据视频的地址，将视频下载在用户手机的相册中，下载完成后下载数加1。其核心代码如下：

//下载视频

\_downLoadVideo(Video video) async {

WidgetsFlutterBinding.ensureInitialized();

FlutterDownloader.initialize();

// 获取存储路径

//var \_localPath = (await \_findLocalPath()) + ConstantData.VIDEO\_LOCALHOST;

var \_localPath = ConstantData.VIDEO\_LOCALHOST;

final savedDir = Directory(\_localPath);

// 判断下载路径是否存在

bool hasExisted = await savedDir.exists();

// 不存在就新建路径

if (!hasExisted) {

savedDir.create();

}

String fileName = video.url.toString().substring(0, video.url.toString().indexOf(".")) + '.mp4';

print(ConstantData.VIDEO\_FILE\_URI + fileName);

// ProgressDialog pr = TsUtils.showProgressDiolog(context, "视频下载中....");

// await pr.show();

await FlutterDownloader.enqueue(

url: ConstantData.VIDEO\_FILE\_URI + fileName,

fileName: fileName,

savedDir: \_localPath,

showNotification: true,

// show download progress in status bar (for Android)

openFileFromNotification: true, // click on notification to open downloaded file (for Android)

);

// pr.hide();

TsUtils.showShort('下载成功, 文件保存在' + \_localPath);

setState(() {

video.downloads ++;

});

Map<String, dynamic> data = Video.model2map(video);

Result result = await DioRequest.dioPut(URL.VIDEO\_DOWNLOAD, data);

}

（5）在视频上传中，用户可以选择性的进行视频的封面和描述进行上传，视频大小不超过15MB，如图5.15所示。点击发布按钮后，客户端会向服务端发送request请求，服务端会取得request中的参数，会先将视频保存在云服务器上，然后用ffmpeg工具将视频切片，转换成hls流存储在云服务器上，在数据库中以hls流的地址进行存储，其核心代码如下：

public String MP42M3U8(String videoUrl) {

String m3u8NamePrefix = videoUrl.substring(videoUrl.lastIndexOf("/") + 1);

String m3u8Name = m3u8NamePrefix.substring(0, m3u8NamePrefix.indexOf(".")) + ".m3u8";

String ml = "ffmpeg -i " + videoUrl +

" -vcodec copy -acodec copy -hls\_list\_size 0 -vbsf h264\_mp4toannexb " + videoFileUrl + m3u8Name;

try {

Process p = Runtime.getRuntime().exec(ml);

} catch (IOException e1) {

log.info("切片失败: " + ml);

e1.printStackTrace();

}

log.info("切片成功: " + ml);

return m3u8Name;

}



图5.15 客户端视频上传页

之后会判断用户是否上传了视频封面，如果有，则以用户上传的封面为视频封面。若没有封面，在会用ffmpeg工具取得视频的首帧图片作为视频的封面，其核心代码如下：

public String getVideoCover(String videoUrl) {

String coverName = UUID.randomUUID().toString().replace("-", "") + ".jpg";

String ml = "ffmpeg -i " + videoUrl + " " + coverFileUrl + coverName;

try {

Process p = Runtime.getRuntime().exec(ml);

} catch (IOException e1) {

log.info("切取失败: " + ml);

e1.printStackTrace();

}

log.info("切取封面成功: " + ml);

return coverName;

}

（5）点击个人信息页的我的视频后进入个人视频页，在进入此页面时客户端会向服务端发送request，服务端认证到个人信息，在数据库查询到个人视频后发送给客户端，客户端在页面上展示个人视频，如图5.16所示，在此页，长按某个向删除的视频，会出现删除提示，如图5.17，选择确定后会向服务端发送请求，服务端会删除关于此视频的一切信息。

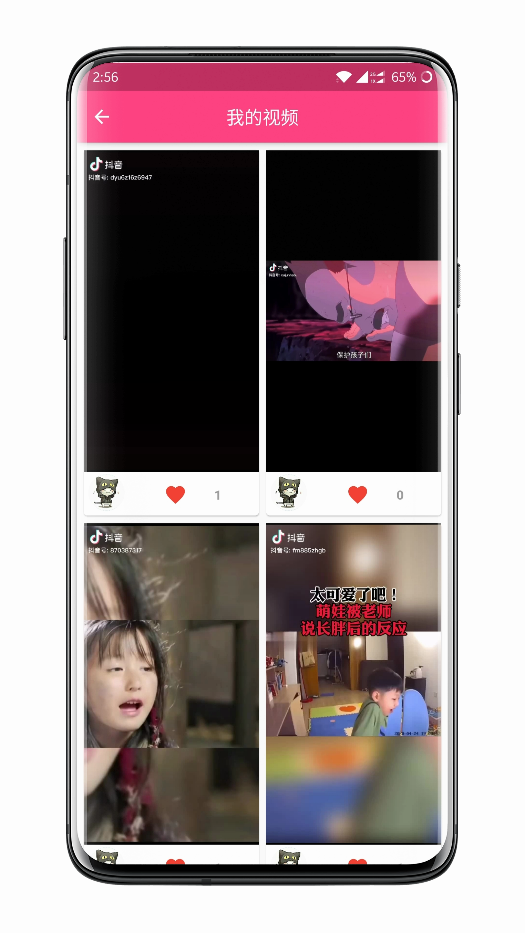


图5.16 客户端个人视频页

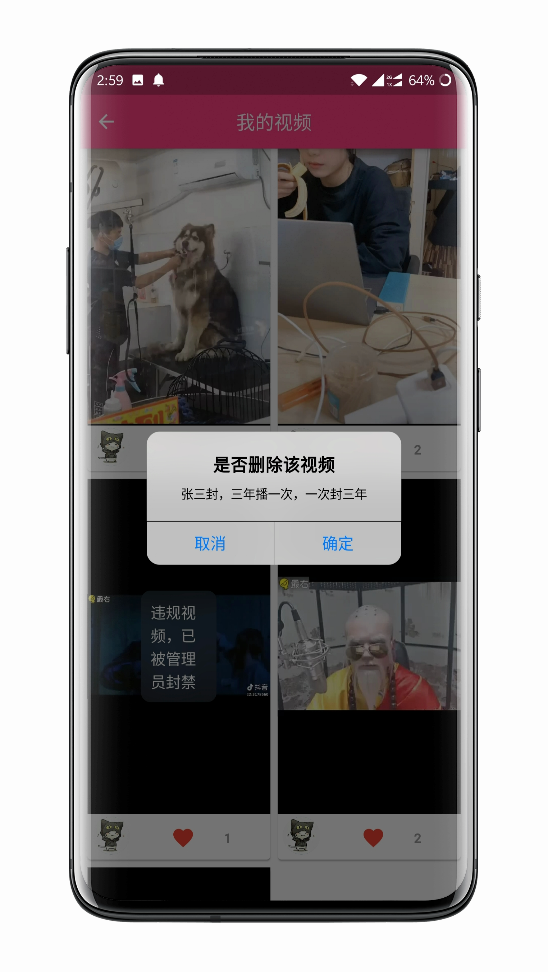


图5.17 客户端个人视频页删除操作

（6）客户端用户上传视频后并不会马上就发布在视频广场，发布的视频会先让管理员审核，如图5.18，管理员可以审核该视频合法或者违规，只有视频合法，视频才会发布在视频广场上。若管理员选择视频违规，该视频将只会出现在个人视频页，并提示视频违规信息，如图5.19。在审核后，管理端会向服务端发送请求，管理端认证管理员身份后，改变视频状态，管理端视频审核页的布局也会发生改变，会显示审核后视频的状态，如图5.20。在管理员审核期间，上传的视频会出现在个人视频页，并提示“审核中”信息，如图5.21。

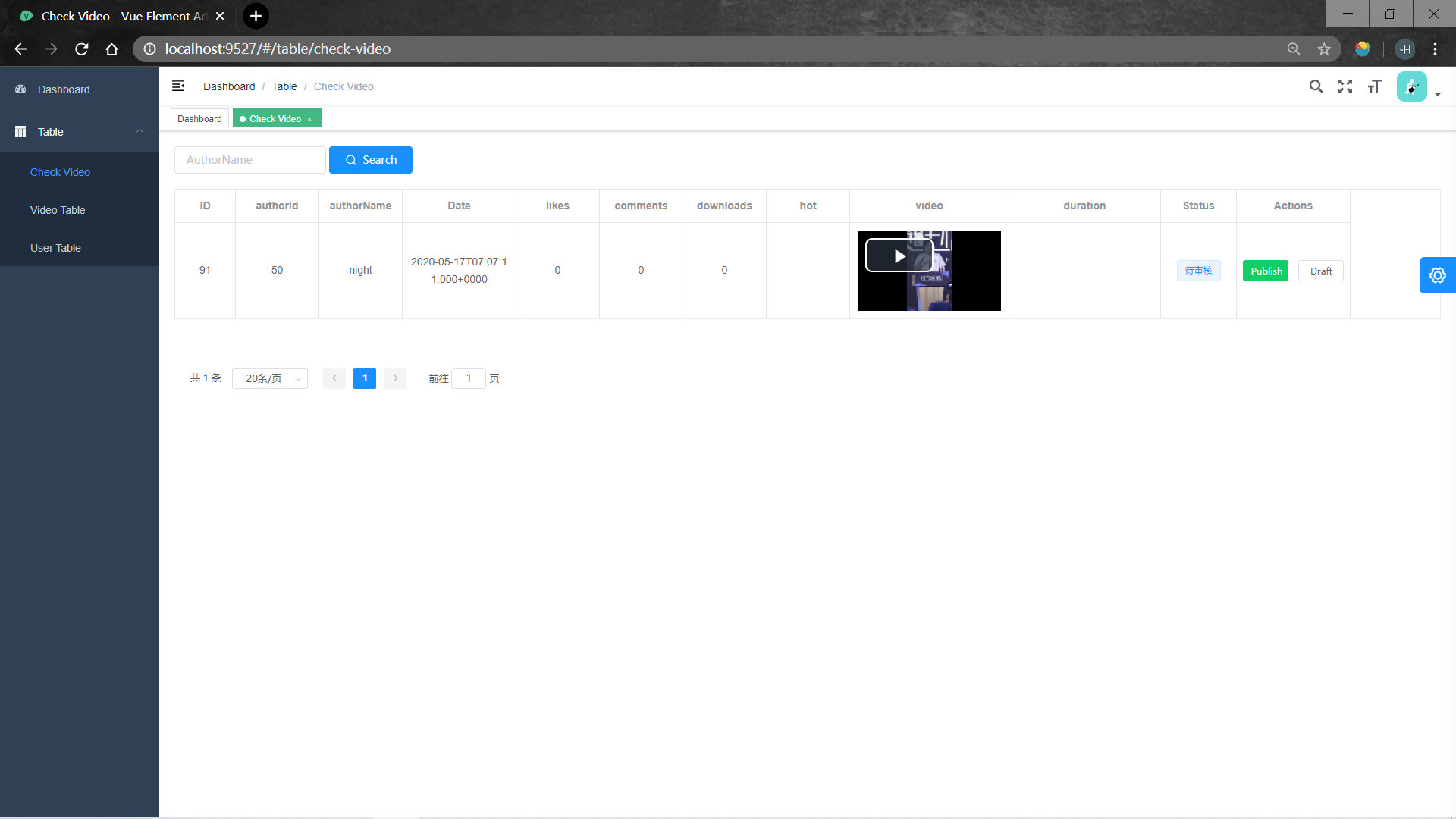


图5.18 管理员视频审核页

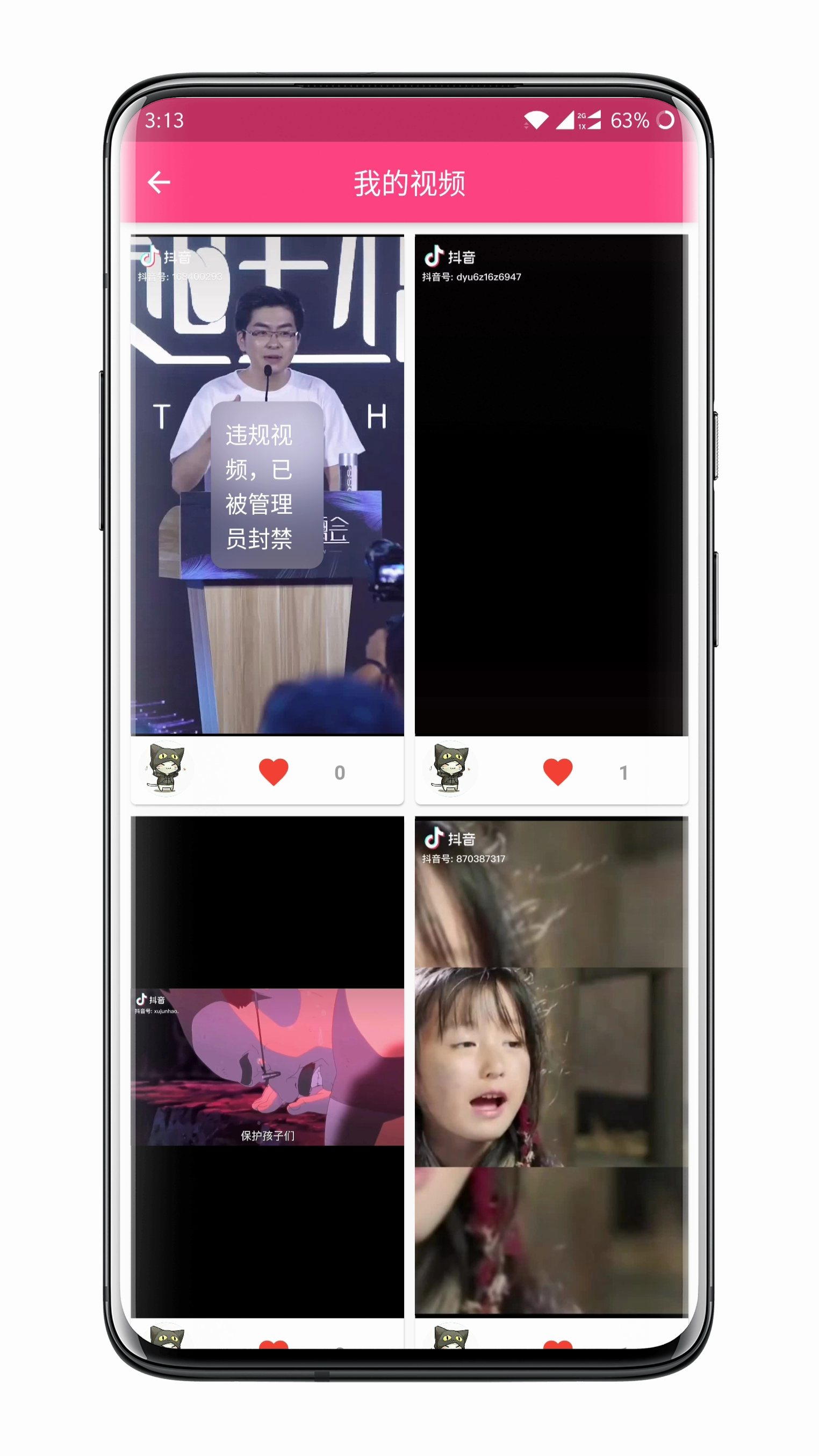


图5.19 客户端个人视频页违规视频提示

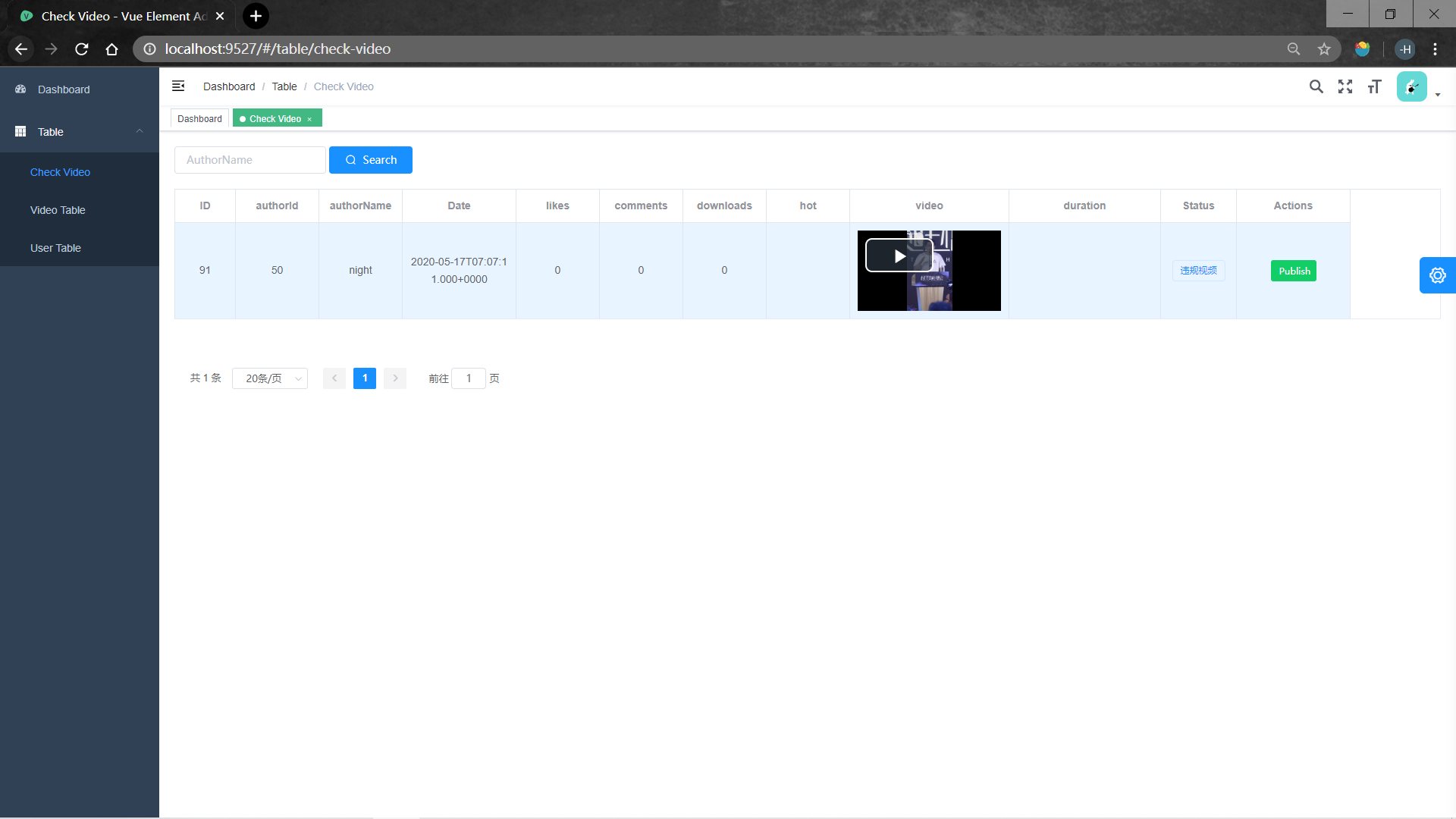


图5.20 管理端视频审核完成

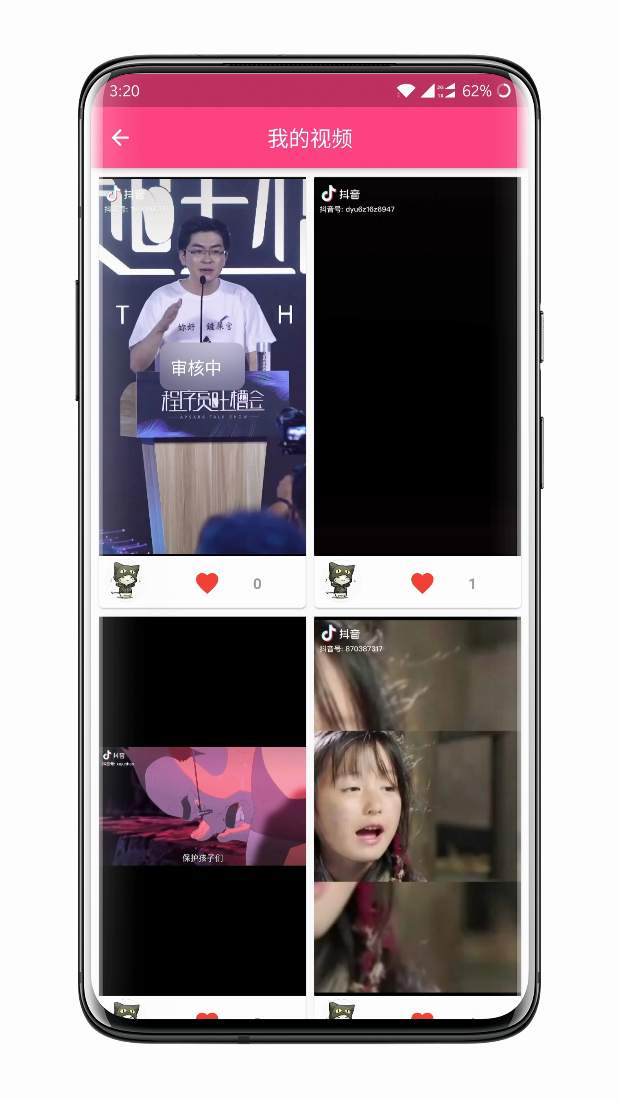


图5.21 客户端视频审核中

（7）在管理端的视频管理页，用户视频搜索功能主要是先获取用户的输入的关键词，然后携带关键词向服务端发送request请求。服务端接收到搜索关键词这个参数后，会去数据库进行模糊搜索。最后将查询的结果返回管理端显示，如图5.22。

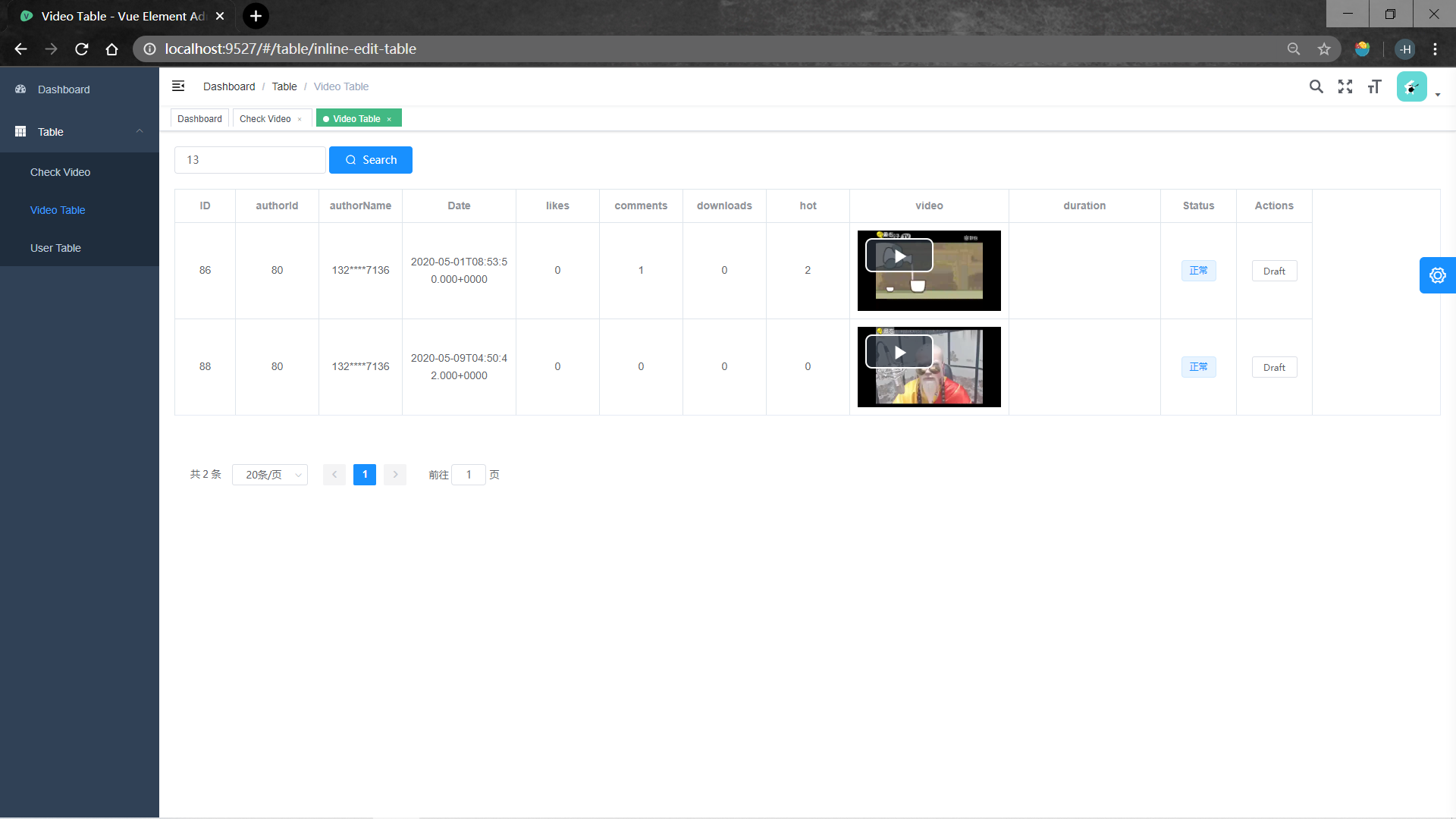


图5.22 管理端搜索后页面

（8）管理员封禁用户视频后会向服务端发送一个请求，服务端认证管理员身份后会改变视频的状态，视频状态改变后，其在管理端的状态显示也将发送变化，如图5.23，同时，在客户端，此视频将无法在视频广场出现，只有在视频作者的个人视频页上，并且有违规视频的提示，如图5.23。

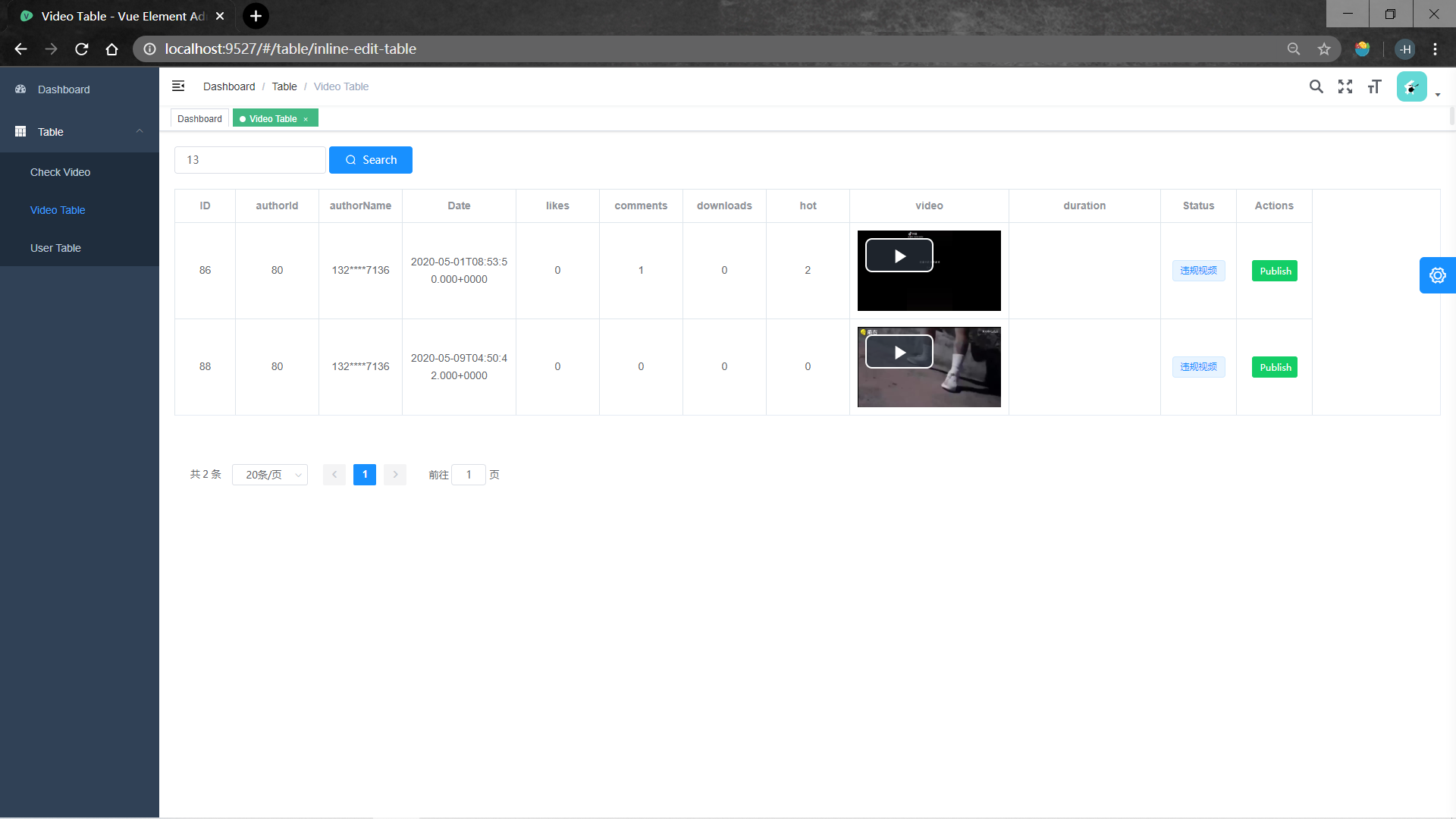


图5.23 管理端违规视频处理后页面

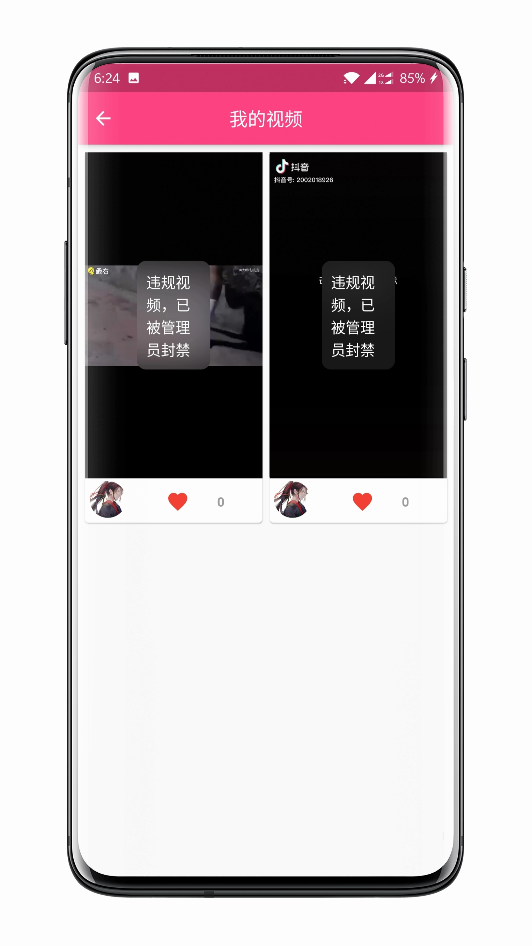


图5.24 客户端个人视频页违规视频显示

5.4 本章小结

本章主要从数据库的设计和系统的用户管理模块和视频管理模块功能设计方面介绍了短视频集合平台夫人详细设计与实现过程。这一阶段的完成意味着系统开发工作的结束。而伴随着系统编码阶段的单元测试以及阶段结束后的综合测试将在下一章节系统测试里详细介绍。

第六章 系统测试

6.1 用户管理模块测试

6.1.1 客户端注册登录功能测试

表6.1 客户端注册登录功能测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试项 | | 测试类别 | 描述/输入/操作 | 期望结果 | 真实结果 |
| 登录注册页 | 注册功能 | 注册登录功能测试 | 注册已有账号 | 注册不成功 | 符合期望结果 |
| 注册新账号 | 注册成功 | 符合期望结果 |
| 登录功能 | 输入不存在账号 | 登录不成功 | 符合期望结果 |
| 输入错误密码 | 登录不成功 | 符合期望结果 |
| sql注入测试 | 登录不成功 | 符合期望结果 |

6.1.2 客户端个人信息页功能测试

表6.2 客户端个人中心页功能测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试项 | | 测试类别 | 描述/输入/操作 | 期望结果 | 真实结果 |
| 个人信息页 | 查看点赞视频 | 未登录功能测试 | 触发事件 | 无法转跳，提示请登录 | 符合期望结果 |
| 查看个人视频 | 触发事件 | 无法转跳，提示请登录 | 符合期望结果 |
| 查看个人关注 | 触发事件 | 无法转跳，提示请登录 | 符合期望结果 |
| 查看个人粉丝 | 触发事件 | 无法转跳，提示请登录 | 符合期望结果 |
| 退出登录 | 触发事件 | 无法执行，提示请登录 | 符合预期结果 |
| 查看点赞视频 | 登录状态功能测试 | 触发事件 | 转跳到我的点赞页面之下 | 符合期望结果 |
| 查看个人视频 | 触发事件 | 转跳到我的视频页面之下 | 符合期望结果 |
| 查看个人关注 | 触发事件 | 转跳到我的关注页面之下 | 符合期望结果 |
| 查看个人粉丝 | 触发事件 | 转跳到我的粉丝页面之下 | 符合期望结果 |
| 退出登录 | 触发事件 | 清除个人信息缓存，退出登录 | 符合预期结果 |
| 重复登录 | 触发事件 | 前者登出 | 符合预期结果 |

6.1.3 客户端个人信息修改

表6.3 客户端个人信息页功能测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试项 | | 测试类别 | 描述/输入/操作 | 期望结果 | 真实结果 |
| 个人信息页 | 修改文字类个人信息 | 信息修改测试 | 触发事件 | 个人信息修改成功 | 符合预期结果 |
| 修改头像 | 拍照设置头像 | 头像更换成功 | 符合预期结果 |
| 从相册选择头像 | 头像更换成功 | 符合预期结果 |

6.1.4 管理端用户管理功能测试

表6.4 管理端用户管理功能测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试项 | | 测试类别 | 描述/输入/操作 | 期望结果 | 真实结果 |
| 管理端  用户管理模块 | 用户搜索 | 用户管理测试 | 输入不完整用户名搜索 | 搜索到用户名类似用户 | 符合预期结果 |
| 用户信息导出 | 触发事件 | 将用户信息导出到excel | 符合预期结果 |
| 重置用户密码 | 触发事件 | 用户密码变为123456 | 符合预期结果 |
| 限制用户登录 | 触发事件 | 用户无法登录 | 符合预期结果 |
| 删除用户 | 触发事件 | 用户所有信息删除 | 符合预期结果 |

6.2 视频管理模块测试

6.2.1 客户端视频浏览测试

表6.5 客户端视频浏览测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试项 | | 测试类别 | 描述/输入/操作 | 期望结果 | 真实结果 |
| 客户端视频模块 | 视频播放页 | 视频功能测试 | 触发事件 | 视频播放 | 符合预期结果 |
| 视频点赞 | 触发事件 | 点赞成功，出现在我的点赞页 | 符合预期结果 |
| 视频评论 | 评论视频 | 在评论页评论并显示 | 符合预期结果 |
| 视频下载 | 触发事件 | 下载视频在相册中 | 符合预期结果 |
| 视频上传 | 上传小于15MB视频 | 上传成功，在我的视频页显示 | 符合预期结果 |
| 上传大于15MB视频 | 上传失败 | 符合预期结果 |
| 上传视频不上传封面 | 视频封面为首帧 | 符合预期结果 |
| 上传视频上传封面 | 视频封面为上传的封面 | 符合预期结果 |

6.2.2 管理端视频管理测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试项 | | 测试类别 | 描述/输入/操作 | 期望结果 | 真实结果 |
| 管理端视频管理模块 | 审核视频 | 登录状态功能测试 | 审核视频为正规 | 客户端广场显示视频 | 符合预期结果 |
| 审核视频为违规 | 视频只出现在跟人视频页，并显示违规提示 | 符合预期结果 |
| 模糊搜索用户视频 | 输入搜索用户不完整用户名 | 搜索出用户所有视频 | 符合预期结果 |
| 播放视频 | 触发事件 | 视频正常播放 | 符合预期结果 |
| 改变视频状态 | 更改视频为违规状态 | 视频只出现在跟人视频页，并显示违规提示 | 符合预期结果 |
| 更改视频为正常状态 | 视频出现在视频广场 | 符合预期结果 |

第七章 结论与展望

7.1 本文主要结果

本毕业设计针对短视频集合平台的设计与实现开展许多调查和学习工作。通过图书馆借阅相关文献材料以及在网络上查询相关学术资料等研究方式学习与掌握了短视频集合平台的设计思想和开放流程。在物质生活过剩的今天，对精神生活的追求成了很大一部分人对生活的定义，而短视频集合平台作为一个可以展示自我、分享生活的平台，完美的满足了这一需求。在那本系统的设计与开发过程中，运用了软件工程思想，将系统进行模块化，循环渐进完成了业务需求，保证了软件的开发质量。使用了主流的框架技术对系统进行了分层开发，便于系统的维护。

在本系统的实现过程中，利用自己所掌握的专业知识，选择使用JavaScript、flutter以及Spring Boot。针对这些技术，通过技术可行性分析和开发效率角度分析，可以认识到他们进行系统开发的优缺点，进而设计出弥补缺点的方案。 根据现实情况，采用最佳的实现方案，最终确定技术路线。首先搭建开发环境，其次搭建业务处理器，之后开发服务端接口，然后设计客户端、管理端页面，最后实现前后端之间请求响应。

7.2 下一阶段的作用

本毕业设计实现了一个完整的短视频集合平台。

在本次系统的开发过程中，客户端采用了新技术flutter，此技术在本次毕业设计之前从未接触过，因此，这是一个边学习便开发的过程，而且flutter在国内网络上也没有过多的技术参考，导致客户端的页面设计不够完美以及存在一些代码的冗余。完成了对客户端的开发，此时已经对flutter技术有了一定的了解，下一阶段的工作可以考虑优化客户端界面以及消除代码冗余。

经过这几个月的毕业设计历程，我深刻认识到了自身在技术方面的不足，只有通过不断学习才能跟上互联网时代的步伐。在以后的工作生涯中，我将扎实理论知识基础，不断学习主流的技术，从而提高自身的能力，让发展道路更为广阔

致谢