VidSphere 项目文档

项目小组 VidSphere 小组

小组成员 陈敏，郭瑞桦，姚承燕，王婷，葛防，项载顺，邓龙溪

联系方式

重庆师范大学软件工程系

摘要

随着手机的普及以及生活节奏的加快，人们在上下班，等车间隙等等一些碎片化时间里，不够看完一部电影，看不了一本小说，人们越来也期待可以在这些时间里拥有可以快速获得情绪价值的娱乐活动。我们平台期望打造成一个实时性，强互动的数字社交空间，让人们在闲暇之余也可以通过他人的分享欣赏到几千里之外的美景，与千里之外的人们沟通交流。

关键词：实时性，强互动，数字社交

日期	修改	描述	作者
2025 年 11 月 1 日	0.1.0	初始版本	
2025 年 11 月 16 日	0.1.1	添加了 bussiness cases	
2025 年 11 月 20 日	0.1.2	更新了用况建模中消费视频内容和发布视频这两个用况	
2025 年 11 月 20 日	0.1.3	统一参与者名称，添加“审核视频”用况	
2025 年 11 月 23 日	0.1.4	修正了用况	
2025 年 11 月 29 日	0.2.1	添加了健壮性分析	
2025 年 12 月 9 日	0.2.2	完成了交互建模和之前的所有部分	
2025 年 12 月 21 日	0.3.0	搭建架构设计框架	

目录

第1章 立项

1.1. 项目起源与提案

1. 起源

市场驱动：在现代社会中，人们的生活节奏越来越快，碎片化时间增多，人们渴望利用碎片化时间获得情绪价值，短视频时长较短，适合在任何碎片化时间观看。并且文字与图片都有一定的局限性，视频更能真实的感受情绪。

技术驱动：智能手机的广泛应用为短视频的创作和观看提供了便捷的硬件设施，人们可以随时拿出手机拍摄或者观看视频。同时由于网络技术的发展，提高了视频加载速度，实现高效的加载视频不再是难题。各种简单、容易使用的视频剪辑软件的面世，降低了视频制作的难度，即便是普通用户也能用一部手机制作出高质量的视频内容。

商业价值驱动：短视频平台会吸引大量的商家入驻，商家可以通过视频平台宣传自己的产品，用户也可以在平台下单该商品。一些主播在直播时会有粉丝送出礼物，礼物是由人民币进行购买后送出，每一个礼物按比例平台和主播分成。

由此打造一个由通过短视频进行交流互动来获得情绪价值的平台成为必然趋势。

2. 提案

人们希望有快速，高质量的短视频平台填充生活中一些碎片化时间，并且创作视频难度降低，不论是有一定年龄的已经参加工作的人们，还是正在上学的学生亦或者是平时在家无聊的老年人，都可以拿起手机随手拍摄生活中的片段发到平台中，平台上有来自五湖四海的人们互相交流，大家可以对同一个视频发表不同的看法，可以通过弹幕，评论与其他人交流自己的想法。我们鼓励各个年龄段的民众拿起手机成为博主分享自己的生活，获得一定量的点赞，评论，转发数可获得一定的奖励。同时平台欢迎各大商家入驻平台，宣传自己的商品，用户可以在平台下单喜欢的商品，在商品到手前，平台可以实时跟踪快递的状态。

资源评估：时间：12 个月，人力：开发团队七人，资金：开发团队人员工资，使用云端服务器的资金。

1.2. Business Case

1. 内容摘要

项目名称：VidSphere - 全民短视频创作与社交平台

核心问题：现代人碎片化时间增多，渴望获得情绪价值，但现有平台未能充分满足各年龄段用户简单创作、深度互动和获得情绪价值的需求。

解决方案：开发一个集多样交互、全龄社交、透明创作激励和电商集成于一体的短视频平台。

投资需求：7 人团队，12 个月开发周期，总预算:开发团队人员工资，使用云端服务器的资金等 xx。

核心价值：通过特定内容推荐机制和构建全龄化社区，快速占领被忽视的中老年及轻度用户市场。

预期回报：预计 24 个月内实现盈亏平衡。

2. 问题陈述

⚡当前市场痛点：存在较大提升空间，足以成为与现有平台竞争的优势

1、互动体验单一：评论和点赞无法满足用户深度的网络社交需求

2、年龄跨度鸿沟：缺乏专门为多年龄段设计的交互界面和内容推荐机制

3、创作者收益模糊：创作者难以从数据中分析获得具体的收益占比

4、内容与商业平衡：内容消费与商品购买体验割裂

⚡业务影响：导致大量潜在用户仅作为内容消费者而非创作者，限制了平台的内容多样性和用户粘性。

3. 解决方案

⚡核心价值主张：

1. 对普通用户："随手拍，随时赚，找到懂你的朋友"

2. 对商家："从内容种草到下单收货的完整营销回路"

3. 对创作者："创作质量越高，回报越高"

4. 对投资者："投资 VidSphere，即是投资于一个**高增长、高竞争力、高创新、且能定义'全龄化社交'这一新品类**的领跑者。"

⚡关键创新点：

1. 多种交互体验：可变化字体模式、语音评论、多种弹幕设置，定制点赞等图像动画

2. 情绪价值社区：基于兴趣和情绪的智能匹配，连接有共鸣的用户

3. 透明数据分析机制：点赞、评论、转发等都会明理规定和分析多少数据量产生多少收益

4. 一体化电商：从视频种草到物流跟踪的无缝体验

4. 商业模式画布

⚡客户细分：

1. 风险投资机构

2. 战略投资者（如大型互联网公司、手机厂商）

3. 天使投资人

⚡价值主张：

1. 巨大市场机会：我们瞄准的是被主流平台忽视的"全龄化"蓝海市场，特别是中老年创作者群体，潜力巨大。

2. 高壁垒模式："多样交互 + 全龄社区 + 透明激励"构成难以复制的商业闭环。

3. 可验证的指标：我们将以清晰的用户增长、日活率和单位经济效益等数据证明增长潜力。

4. 独特生态：为手机厂商（如小米、华为）提供特定的预装内容应用，增强其硬件和内容对中老年用户的吸引力。

5. 流量新大陆：为大型社交平台（如腾讯）开辟其难以触及的"银发经济"和"区域经济"流

量地。

6. 数据价值：获取独特的跨代际用户行为数据，具有极高的战略价值。

7. 卓越团队：展示我们 7 人团队在技术、产品和运营上的互补性与强大执行力。

8. 颠覆性创意：强调我们“服务数字鸿沟”的创新性，这不仅是生意，更是社会价值创造。

9. 高回报潜力：以少量资金在早期支持一个可能成长为独角兽的项目。

⚡渠道通路：应用商店、老年人社区合作、校园推广、社交媒体投放广告

⚡客户关系：战略伙伴与董事会治理，协同工作组与业务技术合作，导师顾问与团队代表

⚡收入来源：

1. 电商佣金（5-15%）
2. 直播礼物分成（50%）
3. 广告收入
4. 定制专属的交互模式
5. 高级会员服务

⚡核心资源：智能推流算法、多年龄 UI 设计能力、创作者社区

⚡关键业务：平台开发、内容审核、创作者运营、商家管理

⚡重要伙伴：支付网关、物流查询 API、各大合作产品商、老年大学和社区、高校

⚡成本结构：人力成本（70%）、服务器（20%）、营销推广（10%）

5. 成本效益分析

⚡投资估算（12 个月）：

1. 成本类别：人力成本

估算金额：xx 万元

备注：7 人团队（含薪资福利）

2. 成本类别：技术基础设施

估算金额：xx 万元

备注：云服务器、数据库存储

3. 成本类别：营销推广

估算金额：xx 万元

备注：用户获取、渠道合作

4. 成本类别：运营管理

估算金额：xx 万元

备注：内容审核、管理员资金

5. 总计

估算金额：xx

⚡收益预测：

第 1 年：主要收入来自直播礼物打赏分成，预计月收入逐步增长至 xx 万元

第 2 年：电商佣金成为主要收入来源，预计月收入 xx 万元

第 3 年：广告和会员服务收入增长，预计年收入 xx 万元

⚡投资回报：

投资回收期：24 个月

3 年 ROI（投资回报率）：预计 100-200%

净现值（NPV）：正 10 万元

6. 风险评估与应对策略

风险类别	风险描述	概率	影响	缓解策略
市场风险	用户增长不及预期	中	高	聚焦细分市场（先攻校园和老年社区），建立口碑
技术风险	推送算法不达效果	中	中	采用成熟框架+人工优化，逐步迭代
实施风险	大部分人群（特别是老年群体）不愿意使用	中	高	推动广告合作和宣传，寻找合适的代言人
运营风险	内容审核压力大	高	高	招录专项审核团队，开发 AI 辅助审核工具
竞争风险	巨头复制核心功能	低	中	快速建立社区壁垒，深耕细分用户需求
财务风险	资金链断裂	中	高	设定明确的阶段性目标，准备融资预案

7. 实施计划

⚡阶段 1（1-6 个月）：核心开发

核心视频上传、播放、互动功能

基础创作者激励系统

//内测用户招募（1000 人）？

⚡阶段 2（7-9 个月）：功能完善

多样互动模式上线

电商功能集成

软件公测一轮

区域推广开始

⚡阶段 3（10-12 个月）：商业化探索

直播功能上线

广告系统测试

软件公测二轮

数据分析和优化

全规模推广

8. 替代方案分析

方案 A：自主研发（选择）

优势：建立核心技术壁垒，完全控制用户体验和业务方向

劣势：前期投入较大，开发周期较长

方案 B：基于开源方案二次开发

优势：启动快速，成本较低

劣势：定制能力有限，难以形成差异化

方案 C：与现有平台合作

优势：风险小，可借助现有流量

劣势：利润空间小，受制于合作方政策

//方案 D：不做任何事

风险：错失服务"数字鸿沟"群体的巨大市场机会

9. 结论与建议

VidSphere 项目精准地切入了一个快速增长但服务不足的市场——寻求透明创作和真实互动的多年龄段用户群体。通过"多样交互+全龄社交+透明激励"的独特定位，我们有能力在短视频红海中开辟蓝海市场。

尽管存在用户获取和内容审核的挑战，但通过聚焦细分市场和构建技术优势，项目具有清晰的盈利路径和可观的投资回报。

我们强烈建议批准该项目，并提供相应预算和资源，立即启动开发。

第2章愿景

2.1. 问题陈述

1. 问题一

要素	描述
问题	**碎片化时间：**随着手机的普及以及生活节奏的加快，人们习惯在上下班路上，写完作业的空挡打开手机放松娱乐，但这些时间过于碎片，不足以在这些时间内看完一个半小时的电影，不够看完几百章的小说，人们急切的需要能在短时间内提供轻松，有趣或有用信息的产品。
影响	满足用户多样化的娱乐需求，让用户可以在闲暇时间找到自己感兴趣的内容
结果	通过提供丰富多样的兴趣内容和良好的用户体验，吸引大量用户注册和使用平台
优点	快速播放视频

2. 问题二

要素	描述
问题	**认识难度：**在网络越来越发达的情况下，人们想要获得一个认识的机会，认识朋友的机会，认识商家的机会，认识新闻的机会。不仅是普通用户想要认识到自己世界之外的事务，各大商家，官方媒体也需要被更多人认识的机会。
影响	人们可以通过视频了解感受其他人，信息也可以通过视频传播达到更多人知晓的效果
结果	事件的影响力增强，对于普法，新闻，文章的传播力增强
优点	传播信息，快速获取信息

3. 问题三

要素	描述
问题	**网络社交的热潮：**手机成为人们的延申，人们渴望更真实，更立体的表现自我，超越文字与图片都局限，在短短的视频中展现自我魅力是当下的发展趋势。
影响	人们可以超越文件和图片的局限性，更真实的表达自我，用户也可以通过视频认识到更真实的别人。可以更深刻的挖据自己感兴趣的领域，也会发现更适合做朋友的人群。
结果	人们通过平台建交，各自分享生活，人与人之间的联系加深
优点	更真实的表达自我，对自己的兴趣挖掘更深刻

2.2. 涉众与用户

1. 涉众

涉众类型	涉众角色	涉众代表	描述
用户	业务代表	A	需求的来源，参与整个项目
用户	业务代表	B	最终用户，主要需求来源，期望可以获得一个实时互动的视频交流平台。
开发团队	需求工程师	C	分析业务代表提供的需求
开发团队	前端开发人员	D	设计软件的界面，保证用户的体验感
开发团队	后端开发人员	E	后端代码实现
开发团队	项目经理	F	负责项目计划，进度跟踪和风险管理，确保项目计划顺利完成
开发团队	测试人员	G	负责产品的质量保障，全面执行测试计划，发现并报告缺陷。
运维团队	系统运维工程师	H	负责平台的运维工作，制定运维规范和业务准入标准，推进运维自动化，提高运维效率和质量。
运维团队	网络运维工程师	I	协助网络管理员进行网络部署，维护和管理，需要确保网络安全，及时处理网络故障和安全问题。
运维团队	安全运维工程师	J	保障系统和网络安全，进行安全漏洞扫描，入侵检测，应急响应等工作。
广告商	出资者	K	通过对平台出资，保证平台开发顺利，获取平台提供的流量为自身打广告，让自己的知名度更高。此处指在开发阶段出资的广告商。
权威机构	网信办		要求软件必须采取必要的措施保护用户个人信息安全，防止信息泄漏和滥用，同时确保平台上的信息传播符合国家法律法规。
权威机构	工信部		制定通信网络安全标准，对于视频软件，会监督其网络接入，视频传输等，保证网络通信的安全可靠。

2. 用户

用户类型	描述
普通观众	广泛存在于各个年龄段和兴趣领域，使用平台主要是用来娱乐并从中获得情绪价值，并不要求一定要从从视频中获得什么信息。对视频内容的多样性和质量有要求，希望通过平台可以找到自己感兴趣的视频。也有部分是跟随大众对平台好奇。

粉丝群体	对某一个博主或者某一位明星有强烈的喜爱和关注度，积极的关注社区成员的动态和视频，为社区成员提供支持和反馈。对博主和明星相关的话题有明显的关注度。
学习者	主要用于在平台搜索相关知识并深入学习，这类用户通常只关注某一方面的普及知识类型的视频，他们的关注在于视频给他们传达的信息，与博主无关，更有可能与话题相关。
专业视频制作团队	拥有专业的设备和人员，能制作出高质量，高水准的视频内容，如动画公司等。通过平台为公司或团队提升知名度，或通过短视频的形式宣传团队的作品。
业余爱好者	不是专门做视频用来盈利的群体，只是爱好摄影或利用闲暇时间分享自己的生活或者观点，他们更注重在平台上与观众和其他社区成员的交流和互动
娱乐博主	平时创作的视频以娱乐为主，主要的获资方式为视频的流量（浏览量，点赞量等）以及直播被打赏的礼物。
宣传博主	主要以宣传某种知识或文化为主，比如宣传中国传统文化之类的博主，这些博主粉丝群体较为稳固，更容易被大众所支持，收益更多是来自官方邀请的宣传活动或者是视频流量收益，直播的情况可能较少。
明星	明星的活动多在电视剧，综艺或者线下活动中，在视频平台一是为了与参演的电视剧合作达到宣传的目的，二是为了让粉丝黏合度更高（通过在视频平台发布视频，不仅可以获得粉丝的关注度，也可以让普通观众发现）
安利博主	安利自己喜欢的人或事物，热爱是最好的发动机，为了让自己喜欢的人或事物让更多人认识到，他们通常会为爱发电，更多的剪辑并发布视频，这类博主通常不是为了盈利来得，主要目的可能是流量。
官方媒体	发布实时新闻，以科普为目的。主要是国家的一些权威的官方媒体，他们的主要目的是为了让更多人关注新闻。
某些企业或公司	为了宣传公司的企业文化或者产品。
运营团队	审核视频，为用户提供服务，处理用户投诉。

2.3. 用户需求

普通观众：可以快速清晰的播放视频，实时发表弹幕和评论，也可以看到其他用户发表的弹幕和评论。

社区成员：可以简洁快速发表视频，可以在后台实时观看到视频播放的数据，类似完播率，点赞率等等，也可以看到视频的点赞量，看到自己发布的所有视频包括隐藏的视频。

运营团队：方便快捷的管理用户投诉和视频。可以快速审核视频，将违反法律和道德的视频下架。

2.4. 产品概述

1. 产品定位陈述

我们致力于打造一个集视频创作，分享，观看与互动为一体的综合性视频社区平台。面向各个年龄段，各个行业，只要有碎片化时间无法得到有效利用这个问题的人，或者想要通过视频交流互动的人。不仅为用户提供了展示创意与才华的舞台，还鼓励用户之间积极交流与合作，形成一个充满活力和创造力的数字文化社区。主要是解决碎片化时间无法得到有效信息以及情绪价值（包括与他人交流获得的情绪价值）的问题。

2. 完整的产品概述

这是一个用户可以自由选择话题和社群的平台，可以对自己喜欢的视频或者内容进行点赞收藏和转发，也可以将不喜欢的话题或用户屏蔽，同时可以在喜欢的视频下进行评论，每一个看见视频的人都可以看见发表的评论，其他用户点赞或者回复都会有提示，每个人都可以自由评论，自由交流。社区成员可以将自己创作的视频上传，不在局限于长视频和图文，随时随地只要想就可以发。上传的视频还要经过运营团队的审核，每一个视频都将由 7 位管理者进行审核，统计所有管理者做出的决定，对于一些负能量和对社会有害的视频内容进行下架或者限流。

2.5. 产品特性

人们可以观看不同种类的视频。

人们可以发布自己想分享的视频到平台。

人们可以对喜欢的视频进行点赞评论转发。

人们可以在评论区进行交流。

系统可以收集视频相关数据进行整合呈现。

2.6. 其他产品需求

1. 可应用标准

《中华人民共和国网络安全法》，《中华人民共和国数据安全法》，《中华人民共和国信息保护法》，《中华人民共和国著作权法》，《网络短视频平台管理规范》，《网络短视频内容审核标准细则》。

2. 系统需求

兼容 Web 端，移动端

3. 其他

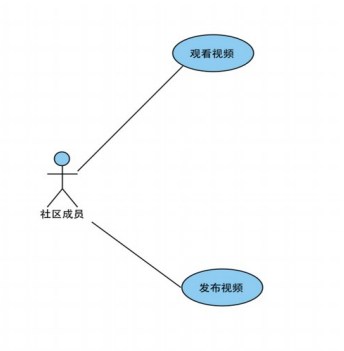
用户手册，安装指南，打包等。

第3章 用况建模

3.1. 术语表

名称	定义
社区成员	平台的内容消费者和内容社区成员，包括浏览视频的阅览者和发布视频的社区成员
管理员	负责内容审核、用户管理和平台秩序维护的后台操作人员
个性化视频推荐流	系统根据用户的历史行为、偏好等信息智能生成的视频内容序列
待审核	视频上传后等待管理员审查的状态，此状态下视频不对外公开显示
已发布	视频通过审核后正式对外公开的状态
审核失败	视频未通过管理员审核的状态，此状态下视频不会被公开显示
已下架	社区成员或管理员主动将视频从公开状态转为非公开状态
可见范围	社区成员设定的自己已发步的视频可见性规则，如公开、仅粉丝可见、私密等
目标受众群体	发布视频的社区成员期望视频推送到的特定用户群体
粉丝关系	用户关注社区成员后建立的订阅关系
黑名单	社区成员设定的禁止与其互动的用户列表
元数据	描述视频信息的数据，包括标题、描述、标签、上传时间等
播放量	视频被播放的总次数
点赞率	视频播放次数与获得点赞数的比例
完播率	用户完整观看视频的比率
敏感词	平台规定的不允许在评论等内容中出现的词汇
操作日志	记录系统所有重要操作的流水记录

3.2. 系统用况



3.3. 系统用例

1. 用况 1: 观看视频

用况名: 观看视频

简明描述: 社区成员浏览推荐流并观看短视频，通过简单操作反馈喜好,并且可以发表自己的观点

系统或子系统：短视频平台

事件流:

基本流:

1. 社区成员打开平台，系统展示个性化视频推荐流

   （验证账号）

2. 系统确认社区成员账号

   （浏览视频）

3. 社区成员上下滑动浏览，选择感兴趣的视频进入观看

   （播放视频）

4. 系统播放选中的视频，并记录播放数据

   （展示视频里的弹幕）

5. 系统在视频播放的不同节点处，在屏幕上放从左到右展示在某一时间节点各个网友发表过的弹幕，弹幕后方显示多少人选择喜欢这条弹幕，多少人选择不喜欢这个弹幕

   （发表弹幕）

6. 社区成员可以选择在视频播放到某个节点时发表弹幕

7. 视频播放完毕，自动播放下一遍

8. 社区成员返回到首页，继续浏览视频

子流:

S1：验证账号

1. 系统查看上一次登陆账号
2. 若没有登陆过的账号，则弹出注册或登陆界面，用户选择注册则系统收录账号密码，如果登陆，系统验证用户所填信息是否正确
3. 自动登陆上一次的账号
4. 展现账号的相关数据

S2：展示视频里的弹幕

1. 系统在视频播放的时间节点检索社区成员在这个节点发表过的弹幕
2. 将弹幕按照发表时间顺序从视频上方从左到右滑过
3. 继续检索下一个节点的弹幕

S3：发表弹幕

1. 社区成员喜欢该视频中的内容
2. 在视频播放的某一刻，发送文字
3. 系统记录发表的文字，发表文字时的时间节点以及账号
4. 系统将文字置于视频上方，从右向左飘过

备选流:

A1: 跳过不感兴趣内容

1. 在**{浏览视频}**当社区成员对内容不感兴趣，快速划过视频
2. 系统减少此类内容的推荐权重

A2: 收藏视频

1. 在**{播放视频}**社区成员对视频内容感兴趣，点击“收藏”
2. 系统将视频列入社区成员收藏列表中
3. 系统记录点赞的账号以及视频中相关话题，并通知发布视频的账号
4. 搜索相关话题的视频，给用户推荐

A3：点赞视频

1. 在**{播放视频}**中社区成员喜欢该视频，点击“点赞”
2. 系统将该视频列入社区成员喜欢列表中，并通知发布视频的账号
3. 系统记录点赞的账号以及视频中相关话题
4. 搜索相关话题的视频，给用户推荐

A4：屏蔽弹幕

1. 在**{播放视频}**中社区成员不喜欢在视频上方出现的弹幕
2. 点击“关闭弹幕”

3. 系统将不在视频上方显示弹幕

A5：点赞弹幕

1. 在**{播放视频}**中，点击视频上方滑过的一条弹幕
2. 系统显示喜欢或不喜欢选项
3. 系统记录选项
4. 系统在弹幕后方显示多少人选择喜欢，多少人选择不喜欢

A6：发表评论

1. 在**{播放视频}**时点开评论区
2. 系统展示所有社区成员曾在观看这个视频时发表的评论以及账号信息，还有该评论的点赞和回复评论的情况
3. 社区成员可以选择在评论区输入自己的评论，系统确认发表的评论没有违规内容，系统会将社区成员发表的评论以及该账号的信息展示在评论区（表示是该账号发表的评论），系统通知发布视频的账号表示该视频新增一条评论
4. 社区成员可以点赞自己喜欢的评论，系统记录点赞数据，更新评论的点赞数，系统通知发布评论的账号表示该账号在该视频下发布的评论被点赞了
5. 社区成员可以选择在回复自己喜欢的评论，系统确认该回复没有违规内容，该回复会展示在评论下方，系统通知发表评论的账号表示该账号在该视频下发表的评论有人回复了，可查看评论
6. 社区成员关闭评论区，继续播放视频

A7：关注账号

1. 在**{播放视频}**时可以关注发布该视频的社区成员
2. 系统提示已关注该社区成员，并且通知被关注者粉丝量加一
3. 系统记录相关数据，被关注者粉丝量加一并且将该账号列入被关注者的粉丝列表里，自己关注量加一并且将被关注者的账号列入关注列表

A8：搜索相关话题的视频

1. 在**{播放视频}**时可以点击视频文案中的话题
2. 系统记录该话题，增加该话题的推荐权重
3. 系统自动检索与该话题相关的视频并罗列出来
4. 返回**{浏览视频}**

A9：搜索视频或账户

1. 在**{浏览视频}**时，社区成员点击搜索栏
2. 输入想要搜索的账号或者话题
3. 系统检索相关话题和账户并罗列出来
4. 返回**{浏览视频}**

A10:举报视频或社区成员

1. 在**{播放视频}**时，社区成员觉得该视频或该用户违反社区规定，点击举报
2. 在系统弹出的举报信息填写界面中填写举报原因以及相关证据
3. 系统收录被举报的作品，改作品的作者，举报人等信息
4. 系统将举报信息派给管理员
5. 管理员查看举报信息并根据举报信息做出处理
6. 系统收录管理员做出的决定，并根据决定对被举报的视频以及作者做出相应处理并告知举报者和被举报者举报结果以及处罚原因和处罚内容等
7. 若作者或举报人对结果不满意，可重新提出诉讼，系统会将诉讼信息重新发送给其他管理员进行处理
8. 返回视频播放界面

A11：举报评论

1. 在**{播放视频}**时，社区成员点开评论区观看评论觉得某一条评论违反社区规定
2. 社区成员右击该评论
3. 点击举报
4. 在弹出的举报信息填写界面如实填写举报原因以及证据
5. 系统告知举报者已受理该举报信息
6. 系统收录举报信息以及评论和评论的账户
7. 系统将处罚信息发送给管理员
8. 管理员根据举报信息做出处理
9. 系统收录管理员做出的处理，对被举报的账户做出相应处罚并告知举报者和被举报者举报结果以及处罚原因和处罚内容等
10. 若作者或举报人对结果不满意，可重新提出诉讼，系统会将诉讼信息重新发送给其他管理员进行处理
11. 返回视频播放界面

A12：违规处理

1. 在**{发表弹幕}**和**{播放视频}**时打开评论区发表评论时，如果发表的内容被系统检测到有敏感词或违规内容
2. 系统会给出警告并记一次警告
3. 如果警告超过三次，系统收录该账号信息，三天内不收录并展示该账号所发的弹幕或评论
4. 如果警告超十次，系统收录该账号半年内不允许该账号发布任何弹幕或评论
5. 如果警告超十五次，系统将终身不许该账号发布任何弹幕或评论
6. 如果用户对处罚结果不认同，可进行申诉处理

A13：申诉

1. 如果社区成员对系统做出的处罚结果不认同，可提出诉讼
2. 在弹出的诉讼信息填写界面中如实填写信息以及证据
3. 系统收录该诉讼信息以及上一次做出处罚的管理员
4. 将诉讼信息发送给除上一次做出处罚结果的三位管理员并由他们做出决定是否认同上一次的处罚，如果两位以上的管理员认同，成员的申诉不会成功并继续做出处罚，用户不认同可以继续提出申诉，如果三次申诉都不成功，将不能提出申诉并强制执行处罚;如果两位以上的管理员不认同上一次的处罚，那么将重新做出处理并交由系统进行处理;
5. 无论结果如何，系统都会告知申诉的成员申诉结果

特殊需求:

前置条件:

1. 系统网络连接正常
2. 数据库中有可观看视频

后置条件:

系统返回首页，阅览者可继续浏览视频或选择其他功能

2. 用况 2: 发布视频

用况名: 发布视频

简明描述: 社区成员上传视频内容并通过审核流程公开发布

系统或子系统: 短视频平台

事件流:

基本流:

（查看个人主页）

1. 社区成员点进自己的主页，系统显示账号中的相关数据
2. 社区成员选择"上传视频"功能，系统显示视频上传界面
3. 社区成员选择本地视频文件上传，系统验证文件格式和大小
4. 上传完成后，系统提示社区成员输入视频信息

（编辑视频信息）

5. 社区成员编辑视频标题、描述和话题标签
6. 社区成员确认提交发布请求

（等待审核）

7. 系统将视频状态置为"待审核"，进入审核队列

（审核视频）

8. 管理员对进入审核队列的视频进行审核

（审核结束）

9. 管理员审核通过后，系统正式发布视频并推送给该社区成员的粉丝以及其他社区成员，提示该社区成员的粉丝更新了

（查看视频数据）

10. 系统列出该账号曾发布过的所有视频，社区成员可以查看已发布的视频的个数，点赞量，收藏量等数据

子流：

S1：查看个人账号

1. 点击个人主页
2. 系统将粉丝量，关注量，发布过的视频等数据展现出来
3. 点击粉丝列表可以看到有哪些人关注自己
4. 点击关注列表可以看到自己关注了那些账号
5. 点击点赞列表可以看到自己点赞过的视频
6. 点击收藏列表可以看到自己收藏过的视频

S2：审核视频

1. 系统将带审核的视频分别发送到在线的管理员
2. 管理员观看视频并审核视频内容以及文案等相关内容
3. 管理员对视频是否审核通过做出决定，审核通过，系统将视频推流，审核不通过，系统通知发布视频的成员审核不通过以及原因

备选流：

A1: 审核不通过

1. 在**{审核结束}中**当**审核不通过**，系统通知社区成员审核结果
2. 系统说明具体原因，视频状态保持"待审核"

A2: 保存草稿

1. 在**{编辑视频信息}**当**社区成员保存草稿**
2. 系统保存当前进度供后续继续编辑

A3：取消发布

1. 在发布视频的任意环节，都可以返回首页取消发布视频
2. 系统会退出发布界面，并删除跟视频有关的所有信息

A4：查看视频数据

1. 在**{查看视频数据时}**社区成员点击已发步的一个视频，
2. 系统将该视频的点赞量，评论量等相关数据展现

A5：拉黑账号

1. 在**{查看个人账号}**时点击个人主页
2. 点进粉丝列表，右击一个账号，系统列出选项
3. 选择拉黑
4. 系统记录操作将该账号移除粉丝列表，并加入黑名单列表，并屏蔽该账号所有消息

A6：移除黑名单

1. 在**{查看个人账号}**中，点击黑名单
2. 系统列出所有拉黑的账号
3. 右击其中一个账号，可选择将账号移除黑名单
4. 系统记录操作，并解除对该账号的消息屏蔽，但并不会将该账号再次列入粉丝列表

A7：下架视频

1. 在**{查看个人账号}**时，选择已发步的一个视频
2. 右击可以选择将该视频将下架
3. 系统记录操作
4. 系统将该视频删除，并且删除该视频相关的一系列数据，比如评论，点赞等数据。删除后其他社区成员将看不到该视频以及视频的弹幕等数据

A8：社区成员设置视频的可见范围和目标受众群体

1. 在**{编辑视频信息}**中，社区成员选择设置视频的可见范围和目标受众群体
2. 选择可见范围
3. 选择目标受众
4. 继续执行第 6 步

特殊需求：

前置条件：

1. 社区成员已登录且通过身份验证
2. 视频文件格式符合要求

后置条件：

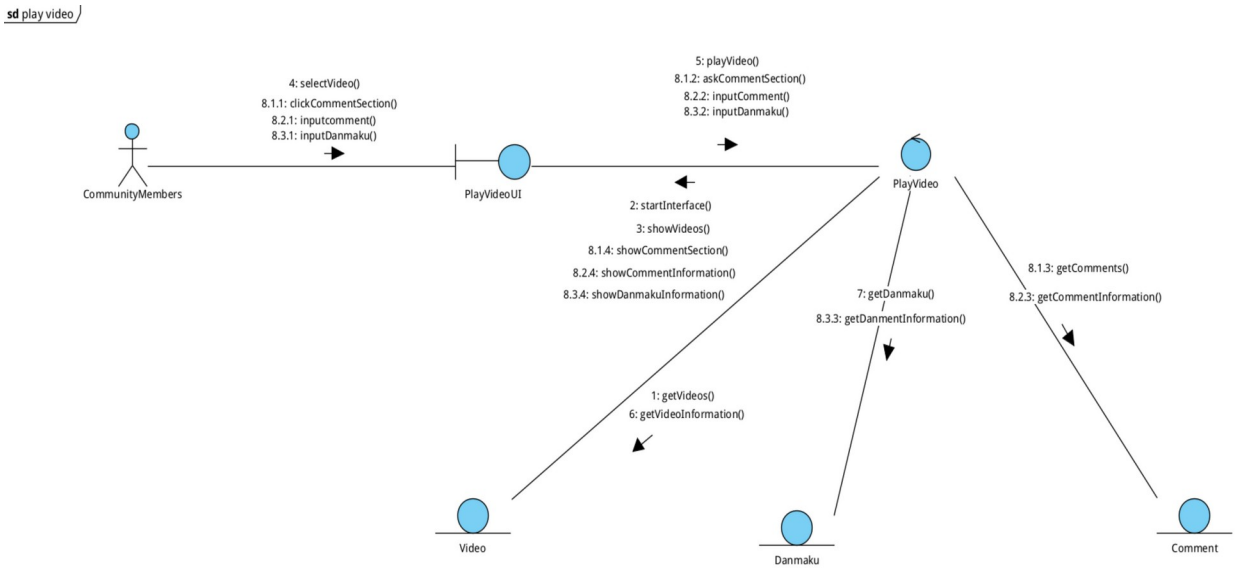
视频进入审核流程或保存为草稿

第4章需求分析

4.1. 健壮性分析

4.1.1 通信图

1.观看视频

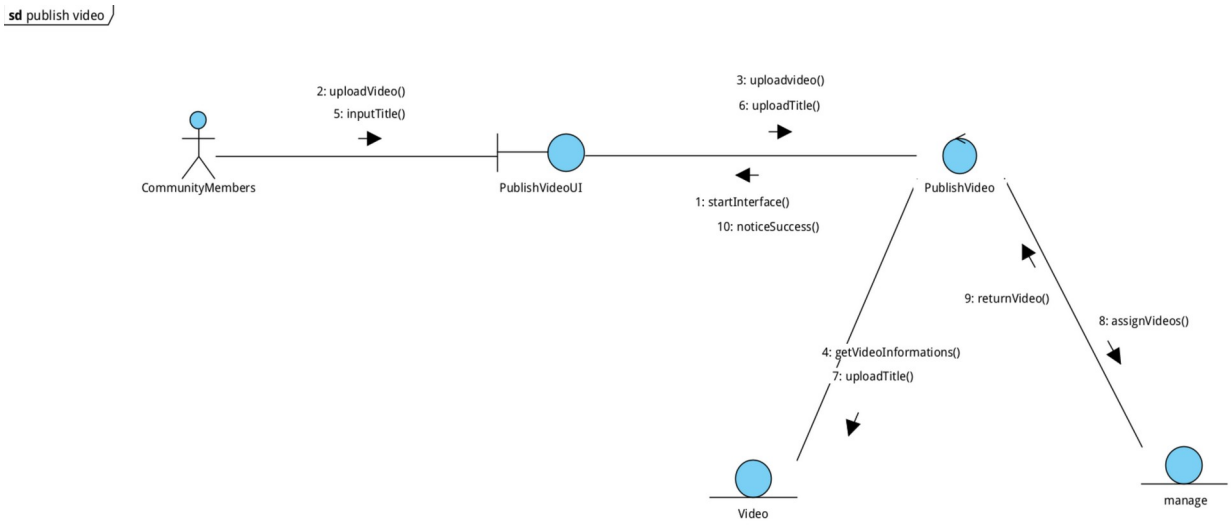


通信图：playVideo

交互描述

1. 控制对象获得视频列表（这个列表是包含了所有系统中存储的审核通过的正常状态的视频），然后请求边界对象开启界面，展示视频列表
2. 社区对象点击一个视频，边界对象请求控制对象播放视频
3. 控制对象获得 Video 的相关信息（是指点击的视频的播放信息，比如播放时间，作者，格式等视频自己的信息），以及 Video 中的一些 Danmaku 信息（指的是不同社区对象在视频播放时发布一些弹幕），之后播放视频时 Danmaku 会在视频上方按发布时间顺序从右向左飘过
4. 社区对象点击评论区，边界对象展示通过请求控制对象获取该视频的所有 Conment 的评论区（指的是控制对象在获得该视频中发布的所有评论后在边界对象中展示评论区中的一些评论）
5. 社区对象上传 Comment，边界对象将控制对象获得的评论信息展示到评论区（社区对象输入评论后传给控制对象存储相关信息后交由边界对象展示在评论区)
6. 社区对象输入 Danmaku，边界对象将控制对象获得的弹幕信息展示在视频上方(社区对象输入弹幕后由控制对象存储相关信息后交由边界对象展示在视频上方）

2.发布视频

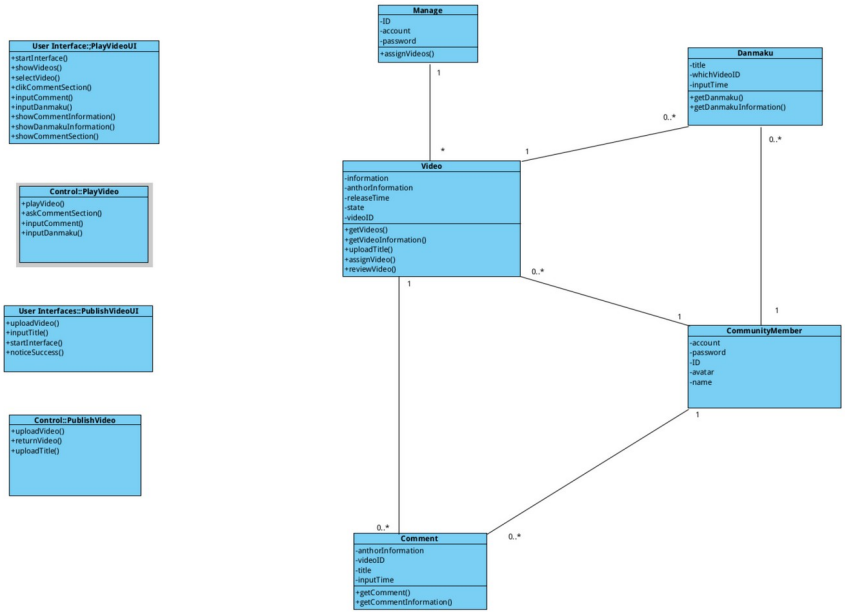


通信图：publishVideo

交互描述

1. 控制对象开启界面
2. 社区对象上传视频，控制对象获取视频信息（此处获取的仅仅是视频自身的信息比如视频流和音频流信息）
3. 社区对象上传视频标题，控制对象获取相关信息（这里是获取标题的信息，该标题是在发布视频时上传的标题，所以是该视频的标题，会随着视频一同发布并保存），视频变为待审核状态
4. 控制对象给管理员分配待审核状态的视频
5. 管理员审核视频并返回审核通过的视频，由控制对象通知成员发布成功同时发布并将该视频保存在系统内（系统将保存视频自身的流信息以及后续上传的标题信息以及作者信息等一同保存）

4.1.2 类图

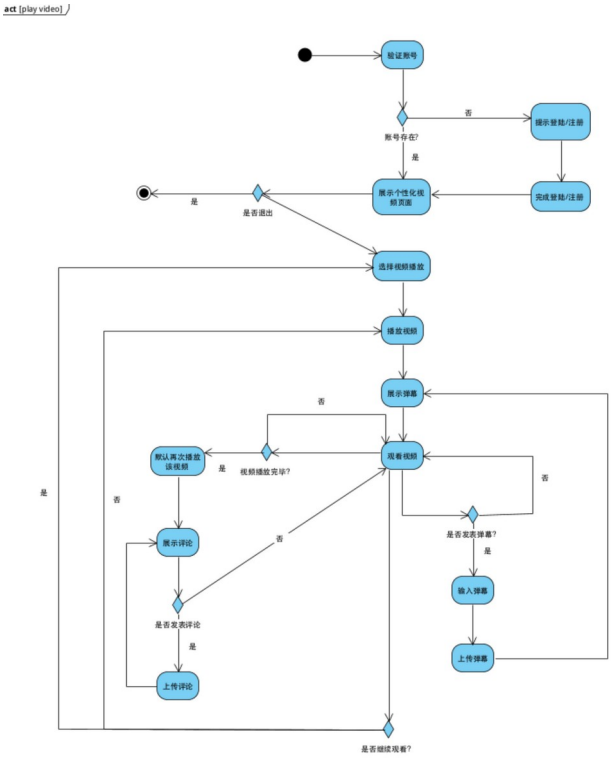


类图：VidShpere

4.2. 交互建模

4.2.1 活动图

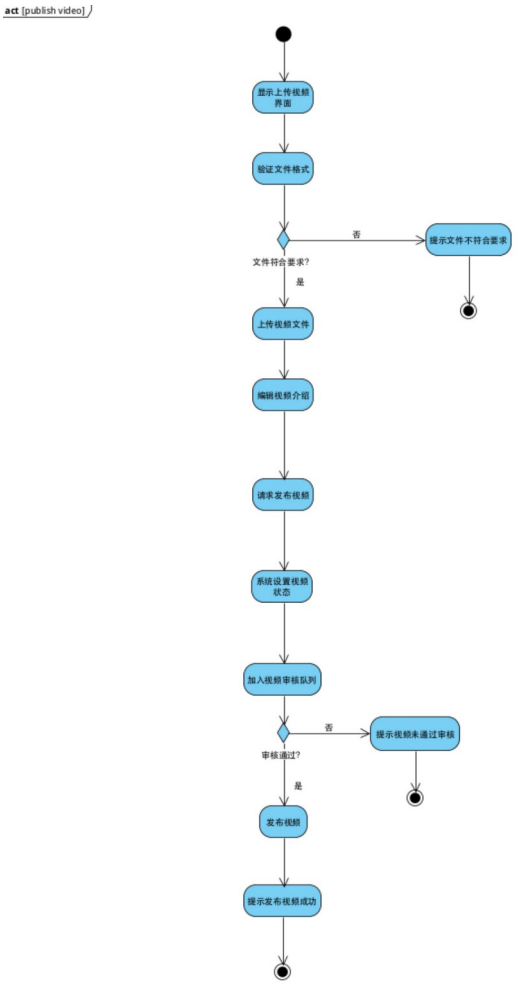
1.观看视频



活动图：playVideo

1. 社区成员打开系统，系统验证成员账号，没有保存的账号则提示注册，有账号则直接登陆
2. 系统展示个性化视频界面（控制对象获取的视频列表）
3. 选择播放一个视频，系统播放视频，在视频上方从右往左展示控制对象获取的弹幕
4. 成员观看视频时选择是否发表弹幕，选择是则输入弹幕，系统保存相关信息并上传弹幕且在视频上方从右向左展示;选择否则继续观看视频
5. 成员选择是否打开评论区，选择是，系统检索所有评论，并在评论区展示评论，此时成员可以选择是否发表评论，选择是则输入并上传评论，系统会保存相关信息并在评论区展示，选择否则继续展示评论;选择不打开评论区则继续观看视频
6. 视频播放完毕，成员可以选择是否推出视频播放，选择是则返回个性化视频界面，选择否则继续播放该视频
7. 在个性化视频界面，成员可以选择是否退出系统，选择是则结束，选择否则继续选择视频

2.发布视频

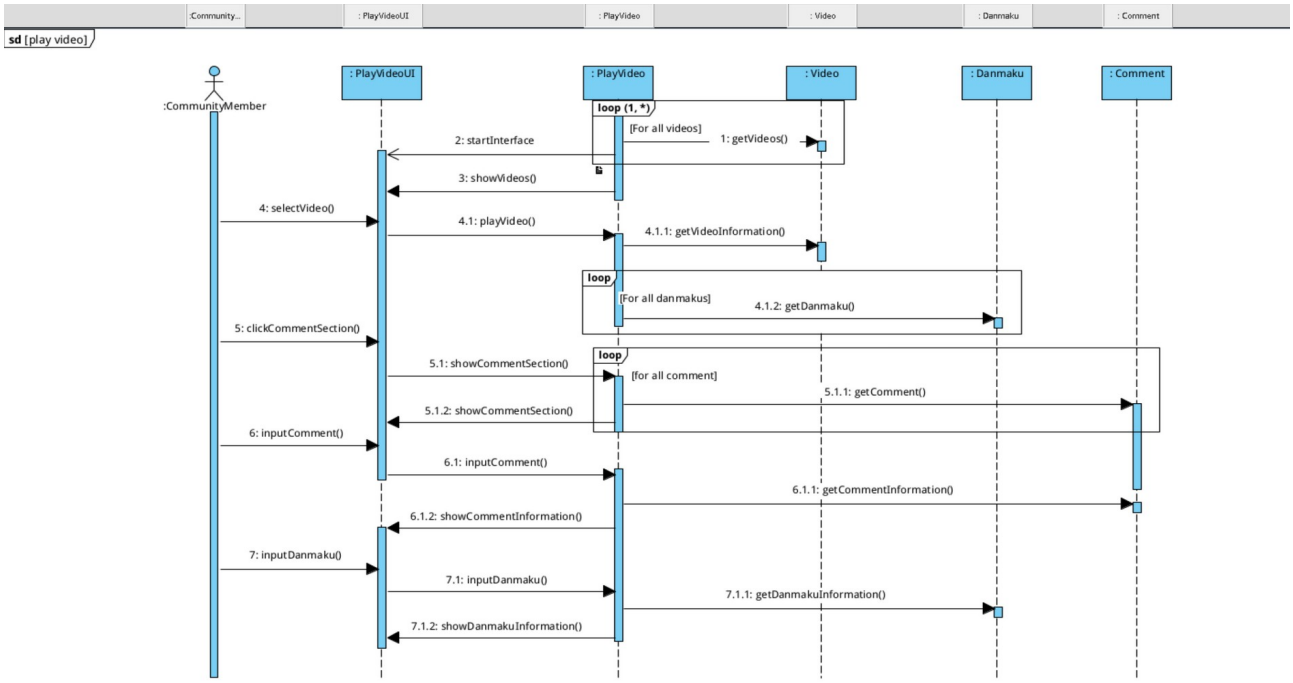


活动图：publishVideo

1. 社区成员进入系统，系统显示上传视频的界面。
2. 社区成员选择上传本地视频，系统会验证该视频的文件格式(mkv,mp4,avi 等）是否符合系统要求的文件格式。
3. 如果格式不符合，提示社区成员文件不符合要求，结束该活动；如果格式符合，进入下一步。
4. 系统上传视频文件完成后，系统提示社区成员输入视频信息，社区成员需要编辑视频介绍，包括视频标题、描述和话题标签。
5. 编辑完成后，社区成员请求发布视频，系统会设置该视频为待审核状态，并加入视频审核队列。
6. 如果审核未通过，系统提示视频未通过审核，该活动结束；
7. 如果视频通过审核，系统发布视频，并提示社区成员发布视频成功，该活动结束。

4.2.2 顺序图/序列图

1.观看视频



序列图：play video

1. 下面是 Viedo.getVideos()消息的说明：

Operation specification: getVideos

Operation intent: 得到系统存储的视频资源

调用显示不需要任何参数，返回类型为 void

Operation signature: Video::getVideos()

Logic description(pre- and post-conditions):

context Video

pre: startInterface()

post: showVideos()

2. 下面是 PlayVideoUI.selectVideo()消息的说明：

Operation specification: selectVideo()

Operation intent: 上下滑动视频选择观看

调用显示不需要任何参数，返回类型为 void

Operation signature: PlayVideoUI::selectVideo()

Logic description(pre- and post-conditions):

context PlayVideoUI

pre: showVideos()

post: 无

3. 下面是 PlayVideo.playVideo()消息的说明：

Operation specification: playVideo()

Operation intent: 播放视频，默认显示弹幕

调用显示不需要任何参数，返回类型为 void

Operation signature: PlayVideo::playVideo()

Logic description(pre- and post-conditions):

context PlayVideo

pre: playVideo()

post: Video.getVideoInformation，Danmaku.getDanmaku()

4. 下面是 PlayVideoUI.clickCommentSection()消息的说明：

Operation specification: clickCommentSection()

Operation intent: 点击评论区，显示评论

调用显示不需要任何参数，返回类型为 void

Operation signature: PlayVideoUI::clickCommentSection()

Logic description(pre- and post-conditions):

context PlayVideoUI

pre: playVideo()

post:
PlayVideo.showCommentSection()，Comment.getComment()，PlayVideoUI.showCommentSection()

5. 下面是 PlayVideoUI.inputComment()消息的说明：

Operation specification: inputComment()

Operation intent: 输入评论并显示

调用显示不需要任何参数，返回类型为 void

Operation signature: PlayVideoUI::inputComment()

Logic description(pre- and post-conditions):

context PlayVideoUI

pre: playVideo()，clickCommentSection()

post:
PlayVideo.inputComment()，Comment.getCommentInformation()，PlayVideoUI.showCommentInformation()

6. 下面是 PlayVideoUI.inputDanmaku()消息的说明：

Operation specification: inputDanmaku()

Operation intent: 输入弹幕并显示

调用显示不需要任何参数，返回类型为 void

Operation signature: PlayVideoUI::inputDanmaku()

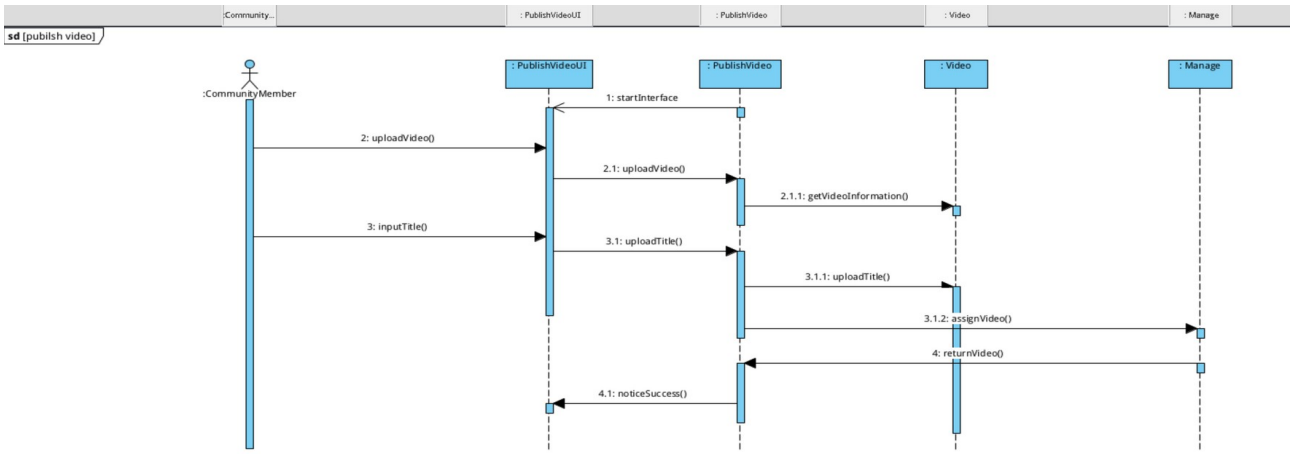
Logic description(pre- and post-conditions):

context PlayVideoUI

pre: playVideo()

post:
PlayVideo.inputDanmaku()，Comment.getDanmakuInformation()，PlayVideoUI.showDanmakuInformation()

2.发布视频



序列图：publish video

1. 下面是 PublishViedoUI.uploadVideo()消息的说明：

Operation specification: uploadVideo()

Operation intent: 上传视频，系统判断视频格式是否支持

调用显示不需要任何参数，返回类型为 void

Operation signature: PlayVideoUI::uploadVideo()

Logic description(pre- and post-conditions):

context PlayVideoUI

pre: startInterface()

post: PublishVideo.uploadVideo()，Video.getVideoInformation()

2. 下面是 PublishViedoUI.inputTitle()消息的说明：

Operation specification: inputTitle()

Operation intent: 输入该视频的介绍，系统将该视频归入审核队列

调用显示不需要任何参数，返回类型为 void

Operation signature: PlayVideoUI::inputTitle()

Logic description(pre- and post-conditions):

context PlayVideoUI

pre: uploadVideo()

post: PublishVideo.uploadTitle()，Video.uploadTitle()，Manage.assginVideo()

3. 下面是 PublishVideo.returnVideo()消息的说明：

Operation specification: returnVideo()

Operation intent: 系统返回已审核的视频，并通知审核视频是否成功

调用显示不需要任何参数，返回类型为 void

Operation signature: PublishVideo.returnVideo()

Logic description(pre- and post-conditions):

context PublishVideo

pre: uploadVideo()，assginVideo()

post: PublishVideoUI.noticeSuccess()

4.3. 操作描述

1. 播放视频

操作名: playVideo

意图: 播放选中的视频，初始化播放环境并加载弹幕

签名: playVideo(videoId: string): void

前置条件:

1. videoId 对应的视频对象状态为“已发布”
2. 用户网络连接正常

后置条件:

1. 视频开始播放

2. 该视频的`playCount`属性值增加 1

3. 弹幕系统准备就绪并显示关联弹幕

逻辑描述:

开始:

```
if 网络连接检测失败 then
    显示提示("网络连接异常");
    return false;
end if;
视频对象 = 数据库查询视频(id = videoId);

if 视频对象 = null OR 视频对象.状态 != "已发布" then
    显示提示("视频不可用");
    return false;
end if;
显示加载中();

弹幕列表 = 数据库查询弹幕(视频 ID = videoId);

if 播放器.加载(视频对象.文件路径) = false then
    显示提示("视频加载失败");
    return false;
end if;

if 弹幕系统.启动(弹幕列表) = false then
    记录日志("弹幕系统初始化失败，视频 ID：" + videoId);
end if;

if 播放器.播放() = true then
    视频对象.播放次数 = 视频对象.播放次数 + 1;
    数据库异步更新(视频对象);
```

隐藏加载状态();

return true;

else

显示提示("播放失败");

return false;

end if;

结束.

2. **发表弹幕**

操作名: submitDanmaku

意图: 用户在视频的特定时间点发表一条弹幕，需通过内容审核

签名: submitDanmaku(userId: string, videoId: string, content: string, timestamp: float): boolean`

前置条件:

1. 用户未被禁言（即`用户.警告次数 < 15`）

2. 弹幕内容不包含系统定义的敏感词

后置条件:

1. 一个新的弹幕对象被创建并持久化

2. 弹幕实时显示在视频的对应时间点

3. 返回`true`表示成功

逻辑描述:

开始:

用户对象 = 数据库.查询用户(条件: id = userId);

if 用户对象.警告次数 >= 15 then

return false;

end if;

if 敏感词过滤器.检测(content) = true then

用户对象.警告次数 = 用户对象.警告次数 + 1;

数据库.更新(用户对象);

return false;

end if;

新弹幕 = 创建 弹幕对象;

新弹幕.内容 = content;

新弹幕.作者 = 用户对象;

新弹幕.所属视频 = videoId;

新弹幕.时间戳 = timestamp;

数据库.保存(新弹幕);

弹幕系统.实时推送(新弹幕);

return true;

结束.

3. **上传并提交视频**

操作名: submitVideo

意图: 用户上传视频文件并提交到审核队列

签名: submitVideo(userId: string, file: File, title: string, description: string): string

前置条件:

1. 用户已登录
2. 文件格式为支持的视频格式且≤500MB
3. 标题和描述不包含敏感词

后置条件:

1. 创建 Video 对象，状态="待审核"
2. 视频文件保存到服务器
3. 视频进入审核队列

逻辑描述:

开始:

if not 文件格式验证(file) or 文件大小>500MB then

返回 "文件格式或大小不符合要求";

end if;

if 敏感词检测(title) or 敏感词检测(description) then

返回 "标题或描述包含违规内容";

end if;

视频 = 创建 Video 对象;

视频.文件路径 = 文件服务器.上传(file);

视频.标题 = title;

视频.描述 = description;

视频.上传者 = userId;

视频.状态 = "待审核";

数据库.保存(视频);

审核队列.添加(视频.id);

返回 "提交成功，等待审核";

结束.

4. 下架视频

操作名：takeDownVideo

意图: 用户或管理员下架已发布的视频

签名: takeDownVideo(operatorId: string, videoId: string, reason: string): void

前置条件:

1. 操作者是视频作者或管理员
2. 视频状态="已发布"

后置条件:

1. 视频状态="已下架"
2. 从推荐流中移除
3. 记录下架原因

逻辑描述:

开始:

视频 = 数据库.查询视频(id = videoId);

if 操作者 != 视频.作者 and 操作者不是管理员 then

返回 "无权限";

end if;

视频.状态 = "已下架";

视频.下架原因 = reason;

推荐系统.移除视频(videoId);

数据库.更新(视频);

结束.

5. **审核视频（管理员）**

操作名: reviewVideo

意图: 管理员审核视频内容并决定是否发布

签名: reviewVideo(adminId: string, videoId: string, approved: boolean, reason: string): void

前置条件:

1. 用户是管理员
2. 视频状态="待审核"

后置条件:

1. 视频状态根据审核结果更新
2. 创建审核记录
3. 通知上传者

逻辑描述:

开始:

视频 = 数据库.查询视频(id = videoId);

if approved then

视频.状态 = "已发布";

视频.发布时间 = 当前时间;

推荐系统.添加视频(视频);

else

视频.状态 = "审核失败";

视频.失败原因 = reason;

end if;

数据库.更新(视频);

通知系统.发送(视频.上传者, "视频审核结果: " + (if approved then "通过" else "不通过"));

结束.

6. **点赞视频**

操作名: likeVideo

意图: 用户表达对当前视频的喜爱

签名: likeVideo(userId: string, videoId: string): void

前置条件:

用户此前未点赞此视频

后置条件:

视频点赞数增加，建立点赞关系，作者收到通知

逻辑描述:

检查是否已点赞，若否，则更新点赞计数与记录，并发送通知

7. **发表评论**

操作名: submitComment

意图: 用户在视频评论区发表评论

签名: submitComment(userId: string, videoId: string, content: string, parentCommentId: string|null): string

前置条件:

1. 用户未被禁言（即用户.警告次数 < 15）
2. 视频状态为"已发布"
3. 评论内容不包含敏感词

后置条件:

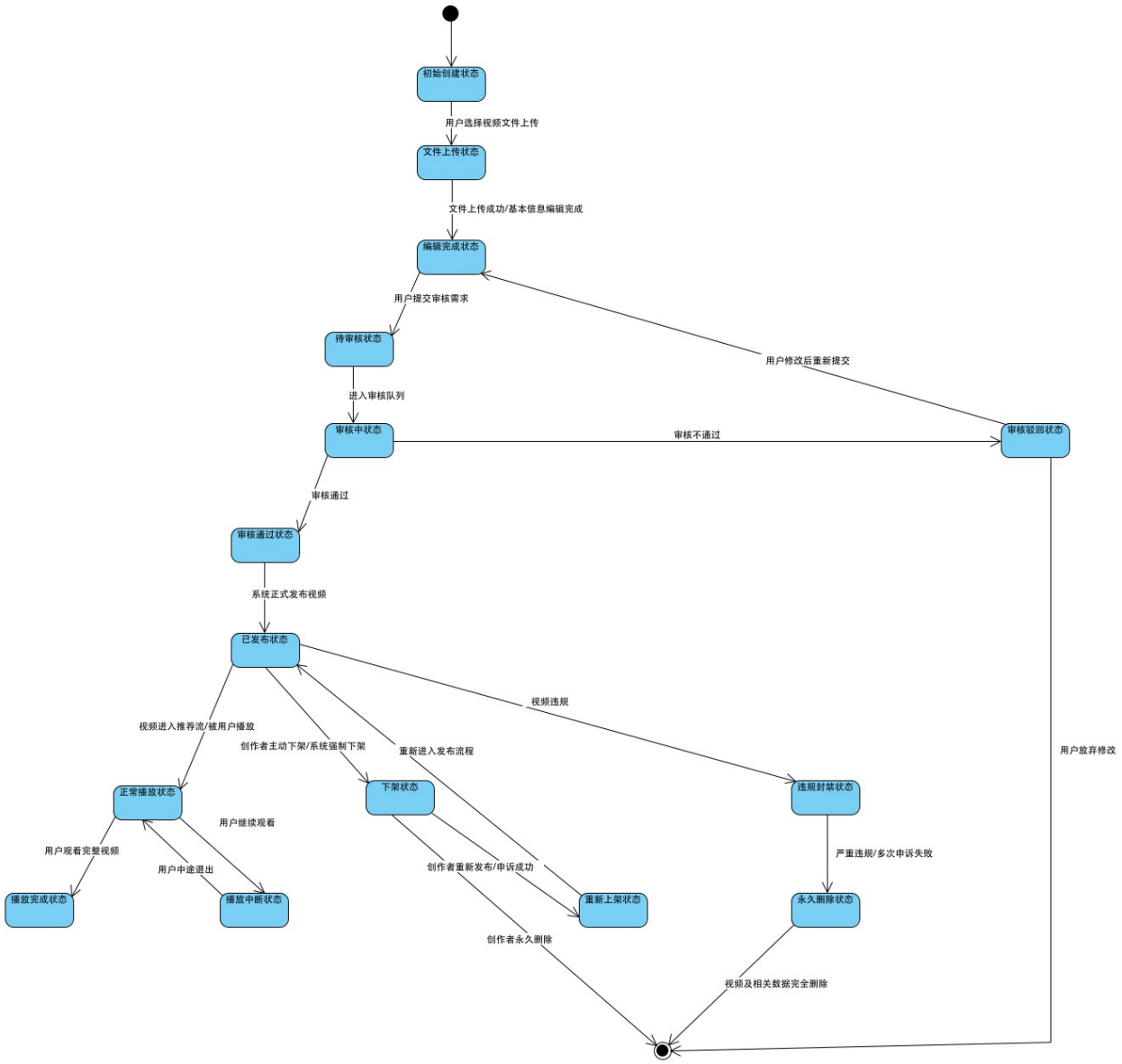
1. 新的评论对象被创建并持久化
2. 评论显示在视频评论区
3. 视频作者收到新评论通知
4. 若为回复评论，被回复者收到通知

逻辑描述:

1. 验证用户状态和评论内容
2. 创建评论对象并保存
3. 发送相关通知
4. 返回评论 ID

4.4. 状态机图

1. 视频对象状态



状态机图：视频状态

状态	触发进入的事件	触发退出的事件
初始创建状态	系统初始化	用户开始编辑视频信息
文件上传状态	用户选择视频文件上传	文件上传成功/基本信息编辑完成
编辑完成状态	文件上传成功/基本信息编辑完成	1. 用户提交审核请求 2. 用户修改后重新提交

状态	触发进入的事件	触发退出的事件
待审核状态	用户提交审核请求	1. 进入管理员审核队列 2. 用户修改后重新提交
审核中状态	进入管理员审核队列	管理员审核通过
审核通过状态	管理员审核通过	系统正式发布视频
审核驳回状态	审核不通过	用户修改后重新提交
已发布状态	系统正式发布视频	1. 进入推荐流 2. 进入播放 3. 被下架 4.视频违规
正常播放状态	视频进入推荐流/用户播放	1. 用户观看完整视频 2. 用户中途退出
播放完成状态	用户观看完整视频	-
播放中断状态	用户中途退出	用户继续观看
下架状态	创作者主动下架(A7)/系统强制下架	1. 创作者重新发布 2. 创作者永久删除
重新上架状态	创作者重新发布	重新进入发布流程
违规封禁状态	严重违规/多次警告(A12)	严重违规/多次申诉失败
永久删除状态	1. 创作者永久删除 2. 严重违规/多次申诉失败	视频及相关数据完全删除

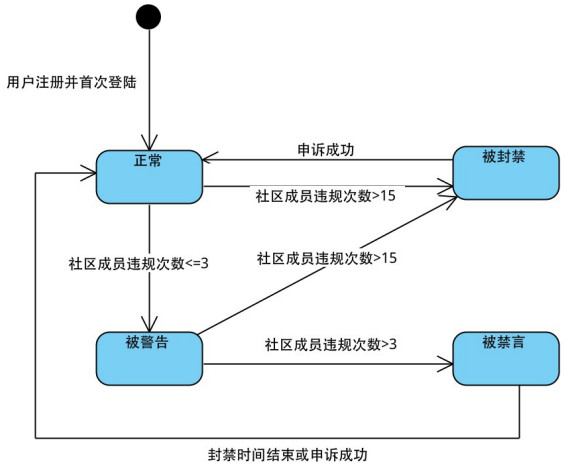
详细状态说明

1. **初始创建状态** - 视频发布的起始状态
2. **文件上传状态** - 视频文件正在上传到服务器
3. **编辑完成状态** - 视频信息和文件都已准备就绪
4. **待审核状态** - 已提交审核，等待管理员处理
5. **审核中状态** - 管理员正在审核视频内容
6. **审核通过状态** – 审核已通过，等待系统发布
7. **审核驳回状态** – 审核不通过，需返回修改
8. **已发布状态** - 视频已正式发布
9. **正常播放状态** - 用户正在播放视频
10. **播放完成状态** - 用户完整观看视频
11. **播放中断状态** - 用户中途停止观看
12. **下架状态** - 视频被暂时下架
13. **重新上架状态** - 视频准备重新上架
14. **违规封禁状态** - 视频因严重违规被封禁
15. **永久删除状态** – 视频被永久删除

状态转换说明

- **用户选择视频文件上传**：从"初始创建状态"转换到"文件上传状态"
- **文件上传成功/基本信息编辑完成**：从"文件上传状态"转换到"编辑完成状态"
- **用户提交审核请求**：从"编辑完成状态"转换到"待审核状态"
- **进入管理员审核队列**：从"待审核状态"转换到"审核中状态"
- **用户修改后重新提交**：从"审核驳回状态"转换回"编辑完成状态"
- **管理员审核通过**：从"审核中状态"转换到"审核通过状态"
- **系统正式发布视频**：从"审核通过状态"转换到"已发布状态"
- **视频进入推荐流/用户播放**：从"已发布状态"转换到"正常播放状态"
- **用户观看完整视频**：从"正常播放状态"转换到"播放完成状态"
- **用户中途退出**：从"正常播放状态"转换到"播放中断状态"
- **用户继续观看**：从"播放中断状态"转换回"正常播放状态"
- **创作者主动下架(A7)/系统强制下架**：从"已发布状态"转换到"下架状态"
- **创作者重新发布/申诉成功**：从"下架状态"转换到"重新上架状态"
- **创作者永久删除**：从"下架状态"转换到终止状态
- **严重违规/多次警告(A12)**：从"正常展示状态"转换到"违规封禁状态"
- **严重违规/多次申诉失败**：从"违规封禁状态"转换到"永久删除状态"
- **用户放弃修改**：从"审核驳回状态"转换到终止状态
- **重新进入发布流程**：从"重新上架状态"转换到"已发布状态"
- **视频及相关数据完全删除**：从"永久删除状态"转换到终止状态

2. 用户状态



状态机图：用户状态

状态	触发进入的事件	触发退出的事件
正常	1. 用户注册并首次登录 2. 申请成功	1. 社区成员违规次数<=3 2. 社区成员违规次数>3 3. 社区成员违规次数>15
被警告	社区成员违规次数<3	申请成功
被禁言	社区成员违规次数>3	申请成功
被封禁	1. 社区成员违规次数>15 2. 被警告状态下违规次数>15	系统结束处理

详细状态说明

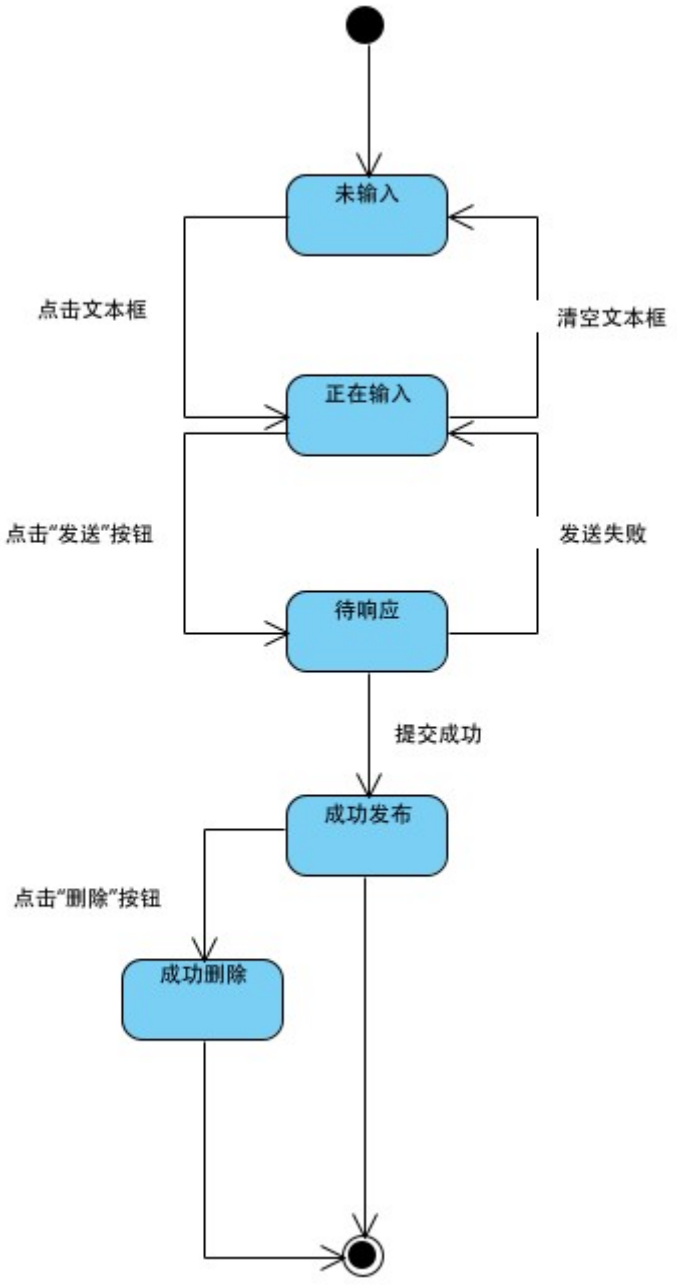
1. **正常** - 用户初始状态，可以正常使用平台功能
2. **被警告** - 用户有轻微违规行为，收到警告但功能正常
3. **被禁言** - 用户有较严重违规行为，被限制发言功能
4. **被封禁** – 用户有严重违规行为，账号被永久封禁

状态转换说明

1. **初始状态**：
    - **正常** ← 用户注册并首次登录
    - 用户首次登录后进入正常状态
2. **正常 → 被警告**：
    - 触发事件：社区成员违规次数<3
    - 事件说明：用户违规次数达到 1-2 次，进入警告状态
3. **被警告 → 被禁言**：
    - 触发事件：社区成员违规次数>3
    - 事件说明：用户违规次数达到 3 次以上，进入禁言状态
4. **被禁言 → 正常**：
    - 触发事件：申请成功
    - 事件说明：用户申诉成功，恢复为正常状态
5. **被警告 → 被封禁**：
    - 触发事件：社区成员违规次数>15
    - 事件说明：用户违规次数达到 15 次以上，直接封禁
6. **被封禁**：
    - 最终状态，无法恢复

- 触发事件：①正常状态下违规次数>15 ②被禁言状态下违规次数>15

3. 评论/弹幕状态



状态机图：评论/弹幕

状态	触发进入的事件	触发退出的事件	目标状态
未输入	1. 系统初始化 2. 清空文本框 3. 成功删除完成	点击文本框	正在输入
正在输入	1. 点击文本框 2. 发送失败后重新输入	1. 点击"发送"按钮 2. 清空文本框	1. 待响应 2. 未输入

状态	触发进入的事件	触发退出的事件	目标状态
待响应	点击"发送"按钮	1. 系统处理成功 2. 系统处理失败	1. 提交成功 2. 发送失败
成功发布	提交成功状态系统确认发布	用户删除操作	成功删除
成功删除	成功发布状态用户删除操作	删除操作完成	未输入

详细状态说明

1. **未输入** - 初始状态，评论文本框为空
2. **正在输入** - 用户在文本框中输入内容
3. **待响应** - 已提交评论，等待系统处理
4. **成功发布** - 评论已正式发布并显示
5. **成功删除** – 已发布的评论被成功删除

状态转换说明

1. **点击文本框**：从"未输入"状态转换到"正在输入"状态
2. **清空文本框**：从"正在输入"状态转换回"未输入"状态
3. **点击"发送"按钮**：从"正在输入"状态转换到"待响应"状态
4. **系统处理失败**：从"待响应"状态转换到"发送失败"状态
5. **系统确认发布**：从"提交成功"状态转换到"成功发布"状态
6. **用户删除操作**：从"成功发布"状态转换到"成功删除"状态
7. **删除操作完成**：从"成功删除"状态转换回终止状态

第5章 架构设计

5.1. 架构概览

1. 架构风格

封闭式分层架构：每一层只能使用下一层提供的服务。

2. 简略描述

系统分为服务器端和客户端，客户端可以通过网络接口将视频上传到服务器端，也可以从服务器端缓存视频进行播放。

客户端主要分为表示层（社区成员使用系统UI界面），业务逻辑层（c++代码实现播放视频，缓存视频，以及用户在UI界面操作的后端实现），数据管理层（保存社区成员的个人信息，以及缓存的视频等信息）;服务器端主要分为接口层（接收客户端的请求），业务逻辑层（接口以及对客户端传过来的视频做的操作）和数据库层（永久性储存视频信息，弹幕，以及评论等等信息）。

5.2. 详细设计

1. 服务器端

. 接入层

1.1.1.接入层概述定位：

作为客户端与业务逻辑层的中间层，负责请求路由、协议转换、安全控制、数据聚合，屏蔽业务逻辑层的复杂性，提供统一、高效、安全的API服务。

目标：

统一入口：为各类客户端（Web、移动端、第三方）提供一致的API接口

协议适配：支持HTTP/REST、WebSocket、gRPC等多种协议

安全防护：统一的身份认证、权限控制和安全校验

流量管控：限流、熔断、降级等流量控制机制

监控可观测：完整的接口调用链路追踪和监控

1.1.2.核心组件

API网关组件

核心职责：

- 统一入口：接收所有客户端请求。
- 路由分发：根据路径、方法、头部将请求路由至对应BFF服务。
- 安全控制：完成JWT校验、IP白名单、限流熔断。
- 协议转换：将HTTP转换为gRPC或WebSocket。

关键子组件：

1. 路由引擎
    - 技术选型：Nginx 1.24 + Kong 3.6（开源网关）。
    - 功能：
        - 基于路径（如/api/v1/videos/*）和方法（GET/POST）路由。
        - 支持动态路由规则（如通过Kong插件配置）。
2. 限流熔断器
    - 技术选型：Kong限流插件 + Sentinel-C++（熔断）。
    - 功能：
        - 单IP每秒最大100次请求（令牌桶算法）。。
3. 认证授权模块
    - 技术选型：JWT + Redis（黑名单）。
    - 功能：
        - 解析JWT令牌，校验用户身份和权限（如普通用户仅能访问视频播放接口）。
        - 维护Token黑名单（如注销的Token）。

部署方案：

- 集群部署（3节点），通过Keepalived实现高可用。
- 每个节点配置4核CPU、8GB内存，支持10万+ QPS。

BFF服务组件

核心职责：

- 客户端适配：针对不同客户端（Web/App/管理后台）提供定制化API。
- 接口聚合：聚合多个微服务接口，减少客户端请求次数。
- 协议适配：将gRPC响应转换为客户端需要的JSON格式。

关键子组件：

1. 聚合服务
    - 技术选型：Node.js 18 + Express（轻量级框架）。
    - 服务划分：
        - Web-BFF：适配PC端，返回桌面端分辨率视频流。
        - App-BFF：适配移动端，支持老年模式（大字体、高对比度）。
        - Admin-BFF：适配管理后台，提供用户管理、数据统计接口。
    - 功能：
        - 聚合视频元数据（VideoService）、作者信息（UserService）、商品信息（ProductService）。
        - 示例接口：

          ```
          http
          ```

```
GET /api/v1/videos/detail?videoId=vid123
```

- → 聚合 `VideoService.GetVideoInfo + InteractionService.GetDanmakuList`

2. 缓存控制模块
   - 技术选型：Redis 7.0（集群模式）。
   - 功能：
     - 对静态数据（如视频封面）设置 Cache-Control 头部。
     - 实现多级缓存（本地缓存→Redis→DB）。

部署方案：

- 容器化部署（Docker + Kubernetes），根据负载动态扩容（HPA）。
- 每个 Pod 配置 2 核 CPU、4GB 内存，支持 5000+并发连接。

接口安全组件

核心职责：

- 数据加密：保护敏感数据（如用户密码、支付信息）。
- 防攻击：拦截 SQL 注入、XSS、DDoS 等攻击。

关键子组件：

1. 加密模块
   - 技术选型：OpenSSL 1.1.1（TLS 1.3）。
   - 功能：
     - 所有接口强制使用 HTTPS（HSTS 头部）。
     - 对敏感字段（如 password）进行 AES-256 加密存储。
2. WAF（Web 应用防火墙）
   - 技术选型：ModSecurity。
   - 功能：
     - 拦截 SQL 注入（如' OR '1'='1）。
     - 过滤 XSS 攻击（如转义<script>标签）。

部署方案：

- 与 API 网关共部署（Nginx + ModSecurity 插件）。
- 规则库每日更新（通过 OWASP Core Rule Set）。

接口层与业务逻辑层的交互

典型流程（以视频播放为例）：

1. 客户端请求：

```
http
```

- `GET /api/v1/videos/play?videoId=vid123`
- API 网关处理：
- 校验 JWT 令牌，确认用户身份。
- 路由至 App-BFF 服务（移动端请求）。
- BFF 服务处理：
- 调用 VideoService.GetVideoInfo(videoId)（gRPC）。
- 调用 InteractionService.GetDanmakuList(videoId)（gRPC）。
- 合并数据并返回 JSON。
- 业务逻辑层响应：
- VideoService 从 MySQL 主从架构读数据（从库负载均衡）。
- InteractionService 从 Redis 缓存读取弹幕（若未命中则回源到 MongoDB）。

. 业务逻辑层

1.2.1 业务逻辑层概述

业务逻辑层负责处理系统的核心业务流程，它是连接接口层与数据层的关键部分。在本系统中，业务逻辑层主要实现视频处理、用户管理、内容审核和互动管理等核心功能。

1.2.2.业务逻辑层组件划分

基于系统需求分析，服务器业务逻辑层包含以下主要组件：

1. 视频处理组件

- 职责：处理视频上传、存储、转码和检索

- 核心功能：

- 接收并验证上传的视频文件

- 调用视频转码服务生成不同分辨率

- 管理视频元数据（标题、描述、标签等）

- 处理视频播放请求

2. 用户管理组件

- 职责：管理用户账户、认证和权限

- 核心功能：

  - 用户注册和登录验证

  - 用户信息维护和更新

- 权限验证和会话管理
- 用户状态监控（正常/警告/禁言）

3. 内容审核组件
- 职责：审核用户上传的内容
- 核心功能：
- 自动检测违规内容（敏感词、不良画面等）
- 管理人工审核队列
- 记录审核结果并更新内容状态
- 处理用户申诉请求

4. 互动管理组件
- 职责：处理用户间的互动行为
- 核心功能：
- 评论和弹幕的发布与检索
- 点赞、收藏、关注等交互操作
- 实时互动消息推送
- 用户互动数据统计

1.2. 3 业务逻辑层接口设计

业务逻辑层为表示层提供以下主要服务接口：

1. 视频服务接口
- 上传视频(文件, 元数据) -> 上传结果
- 获取视频(视频 ID，元数据) -> 视频详情
- 获取视频列表(过滤条件, 分页) -> 视频列表
- 更新视频状态(视频 ID, 新状态) -> 操作结果

2. 用户服务接口
- 注册用户(注册信息) -> 注册结果
- 登录用户(凭证) -> 登录结果
- 获取用户信息(用户 ID) -> 用户信息
- 更新用户状态(用户 ID, 新状态) -> 操作结果

3. 审核服务接口

- 提交审核请求(内容 ID, 内容类型) -> 提交结果

- 获取审核任务() -> 待审内容列表

- 提交审核结果(内容 ID, 审核结果, 审核人) -> 操作结果

4. 互动服务接口

- 发表评论(用户 ID, 视频 ID, 内容) -> 评论结果

- 点赞视频(用户 ID, 视频 ID) -> 操作结果

- 发表弹幕(用户 ID, 视频 ID, 内容, 时间点) -> 操作结果

- 获取视频互动数据(视频 ID) -> 互动统计

. 数据管理层

核心定位

作为服务器端数据持久化核心，承接 C++ 的数据读写请求，负责结构化数据（用户、视频）与非结构化数据（弹幕、评论扩展属性）的统一存储，保障数据一致性、高可用与高效访问。

设计原则

- **单一数据源**：全业务数据统一存储于 MySQL，通过 JSON 字段适配非结构化数据，避免多存储组件维护成本。
- **C++ 友好**：适配 MySQL Connector/C++ 驱动，支持标准 C++20 语法，数据类型与 C++ 对象无缝映射。

核心表设计（MySQL+C++ 映射）

1. 用户表

字段名	数据类型	约束	默认值	备注（C++ 映射）
user_id	varchar(32)	主键	-	std::string
username	varchar(20)	非空，唯一	-	std::string
password	varchar(64)	非空	-	std::string
phone	varchar(11)	唯一	NULL	std::string
email	varchar(50)	唯一	NULL	std::string
role	enum('user','creator','admin','merchant')	非空	'user'	enum class UserRole
status	enum('normal','frozen','forbidden')	非空	'normal'	enum class UserStatus
violation_count	int	非空	0	int
fans_count	bigint	非空	0	long
create_time	varchar(30)	非空	CURRENT_TIMESTAMP	std::string
update_time	varchar(30)	非空	CURRENT_TIMESTAMP ON UPDATE	std::string

字段名	数据类型	约束	默认值	备注（C++ 映射）
			CURRENT_TIMESTAMP	

2. 视频表

字段名	数据类型	约束	默认值	备注（C++ 映射）
video_id	varchar(32)	主键	-	std::string
user_id	varchar(32)	非空，外键（user.user_id）	-	std::string
title	varchar(50)	非空	-	std::string
description	text	-	NULL	std::string
tags	json	-	NULL	std::vector<std::string>
file_url	json	非空	-	nlohmann::json（多分辨率地址）
duration	int	非空	-	int（秒）
format	enum('mp4','avi','mkv')	非空	-	enum class Video
status	enum('pending','reviewing','published','rejected','deleted')	非空	'pending'	enum class VideoStatus
visible_scope	enum('public','fans','private')	非空	'public'	enum class Video
play_count	bigint	非空	0	long
like_count	bigint	非空	0	long
comment_count	bigint	非空	0	long
danmaku_count	bigint	非空	0	long
create_time	varchar(30)	非空	CURRENT_TIMESTAMP	std::string
update_time	varchar(30)	非空	CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP	std::string
product	json	非空	-	std::vector<std::string>

3. 互动表

字段名	数据类型	约束	默认值	备注（C++ 映射）
interaction_id	varchar(32)	主键	-	std::string
user_id	varchar(32)	非空，外键（user.user_id）	-	std::string
video_id	varchar(32)	非空，外键（video.video_id）	-	std::string

字段名	数据类型	约束	默认值	备注（C++ 映射）
type	enum('danmaku','comment','like','collect')	非空	-	enum class InteractionType
content	varchar(500)	-	NULL	std::string（弹幕/评论）
timestamp	int	-	NULL	std::vector<int>（秒）
status	enum('normal','deleted')	非空	'normal'	enum class InteractionStatus
create_time	varchar(30)	非空	CURRENT_TIMESTAMP	std::string

4. 审核记录表

字段名	数据类型	约束	默认值	备注（C++ 映射）
review_id	varchar(32)	主键	-	std::string
admin_id	varchar(32)	非空，外键（user.user_id）	-	std::string
video_id	varchar(32)	非空，外键（video.video_id）	-	std::string
result	enum('passed','rejected')	非空	-	enum class Manage
reason	text	非空	-	std::string (驳回原因)

5. 关注表

字段名	数据类型	约束	默认值	备注（C++ 映射）
follower_id	varchar(32)	主键	-	std::string
followee_id	varchar(32)	非空，外键（user.user_id）	-	std::string
create_time	varchar(30)	非空	CURRENT_TIMESTAMP	std::string

6. 黑名单表

字段名	数据类型	约束	默认值	备注（C++ 映射）
user_id	varchar(32)	主键（user.user_id）	-	std::string
blocked_user_id	varchar(32)	非空，外键（user.user_id）	-	std::string

7. 举报记录表

字段名	数据类型	约束	默认值	备注（C++ 映射）
reporter_id	varchar(32)	主键(user.user_id)	-	std::string (举报人)

字段名	数据类型	约束	默认值	备注（C++ 映射）
report_id	varchar(32)	非空，外键（user.user_id）	-	std::string（记录举报 ID）
target_type	enum('video', 'comment', 'user','danmaku')	非空	-	enum class Report Type
target_id	varchar(32)	非空	-	std::string
reason	text	非空	-	std::string
evidence	json	非空	-	std::vector <std::string> (截图/连接等)
status	enum('pending', 'processed', 'dismissed')	非空	-	enum class ReportStatus
process_result	text	非空	-	std::string
process_admin_id	varchar(32)	非空，外键（user.user_id）	-	std::string
create_time	varchar(30)	非空	CURRENT_TIMESTAMP	std::string
process_time	varchar(30)	非空	-	std::string

2. 客户端

. 表示层

这个部分是这个软件系统中与用户直接接触的部分，负责所有面向用户的视觉呈现、交互逻辑与多平台适配，它的核心是提供完整的用户界面和交互体验，通过网络与服务器端通信，自身不处理业务逻辑或数据持久化。

主要功能模块与页面设计：

1. 用户登录与认证模块

这个模块包含了用户的登录，注册与退出登录状态以及记住登录状态，界面的小字部分还有忘记密码，找回密码以及注册协议，

注册：新用户通过此注册来创建账户，系统收集必要信息并验证其唯一性，确保账户安全合规后再对其信息进行保存

登录：该模块负责用户身份验证，提供安全的登录入口。用户通过输入注册时使用的凭据（邮箱/手机号+密码）来访问系统功能。系统支持记住登录状态功能，避免频繁登录。

忘记密码：用户忘记密码时通过此模块重新设置密码，需经过身份验证。

2. 视频浏览主界面模块

这个模块的功能主要包含了展示视频列表无限滚动加载，视频分类筛选，搜索视频等，这些功能能让用户在视频浏览的时候更加方便，界面里面还包含搜索框，用户头像等信息。

视频列表展示区：以网格或列表形式展示视频内容即可

3. 视频播放界面模块

这个模块的功能主要包含有播放视频，显示视频信息，点赞/收藏/分享，弹幕功能，评论功能，还包含着顶部操作栏的一些返回以及

视频播放：提供完整的视频播放控制为主要功能

互动操作：用户与视频内容互动的各种操作按钮，包含放大，点赞收藏转发等

4. 个人中心界面模块

这个模块主要包含显示用户信息，管理发布的内容，查看粉丝/关注，设置，以及设置，编辑资料等基础功能的模块

5. 客户端表示层的职责总结

界面渲染：将数据转换为用户可理解的视觉元素

用户交互：捕获并响应用户的操作

数据获取：通过网络服务器获取数据

状态管理：管理本地应用状态（如用户登录状态）

路由导航：管理页面之间的跳转

本地存储：缓存必要数据以提升体验

错误处理：处理网络错误和异常情况

加载状态：提供友好的加载提示

这个客户端表示层是一个完整的前端应用，它通过 HTTP 请求与服务器通信，专注于提供良好的用户界面和交互体验，而将业务逻辑和数据持久化完全交给服务器端处理

. 业务逻辑层

2.2.1 客户端逻辑层概述

客户端逻辑层负责处理用户界面交互、本地状态管理、请求封装及响应解析，是用户与服务器之间的桥梁。在本系统中，客户端逻辑层主要实现视频播放控制、用户操作处理、本地缓存管理及实时互动响应等核心功能。

2.2.2 客户端逻辑层组件划分

基于用户交互需求，客户端逻辑层包含以下主要组件：

1.视频播放组件

- 职责：管理视频加载、播放、控制与画质切换

- 核心功能：

  - 请求并加载指定视频流（支持多分辨率）

  - 控制播放/暂停、进度跳转、音量调节

  - 根据网络状况自动切换清晰度

– 本地缓存关键播放元数据（如观看历史、播放进度）

2.用户交互组件

– 职责：处理用户输入与状态反馈

- 核心功能：
  - 管理登录状态与会话令牌（Token）
  - 封装注册、登录、登出等操作请求
  - 本地校验用户输入（如密码强度、昵称合法性）
  - 同步用户信息变更（头像、简介等）

3. 内容展示组件

- 职责：组织与呈现视频、评论、弹幕等内容
- 核心功能：
  - 获取并渲染视频列表（推荐、搜索、个人主页等）
  - 拉取并展示评论/弹幕（支持分页与实时更新）
  - 处理视频元数据显示（标题、标签、作者信息）
  - 支持本地搜索与筛选（基于缓存数据）

4.互动行为组件

- 职责：封装用户互动操作并响应反馈
- 核心功能：
  - 发起点赞、收藏、关注等请求
  - 提交评论或弹幕（含敏感词本地预检）
  - 接收实时通知（如新评论提醒、审核结果）
  - 本地记录互动状态（避免重复操作）

2.2.3 客户端接口设计

客户端逻辑层为用户界面层提供以下主要服务接口：

1.视频播放接口

- 播放视频(视频 ID) → 播放器初始化结果
- 加载视频详情(视频 ID) → 视频元数据
- 获取推荐视频列表(用户 ID, 分页参数) → 视频列表
- 切换清晰度(分辨率标识) → 切换结果（成功/失败）

2.用户交互接口

- 执行登录(账号, 密码) → 登录状态与用户信息
- 执行注册(注册信息) → 注册结果（含错误提示）
- 获取当前用户信息() → 当前用户数据（含权限状态）
- 退出登录() → 本地状态清除结果

3.内容展示接口

- 搜索视频(关键词, 分页) → 视频搜索结果列表

- 获取评论列表(视频 ID, 分页) → 评论数据集合

- 获取弹幕流(视频 ID, 时间范围) → 弹幕数据列表

- 获取个人主页内容(用户 ID) → 用户视频与互动摘要

4.互动行为接口

- 发送评论(视频 ID, 内容) → 本地评论对象（含状态：发送中/成功/失败）

- 点赞操作(视频 ID) → 点赞状态更新结果（已赞/取消）

- 发送弹幕(视频 ID, 内容, 时间点) → 弹幕提交结果

- 获取互动统计(视频 ID) → 点赞数、评论数、收藏数等聚合数据

. 数据库层

目标与定位

客户端数据库层负责本地数据的持久化存储与管理，支持离线观看、用户配置存储、缓存元数据管理、播放记录跟踪等功能，确保用户在弱网或无网络环境下仍能流畅使用核心功能。

设计原则

- **封闭式分层**：数据库层仅对业务逻辑层提供服务接口，不直接与表示层交互。
- **轻量级本地化**：仅存储客户端必要数据，避免冗余，数据模型与服务器端对应但结构简化。
- **数据同步机制**：支持与服务器端数据库的增量同步，保障数据一致性。
- **安全性**：敏感信息加密存储，访问控制严格。

2.3.1 数据库层组件划分（基于封闭式分层）

（1）数据模型层（Data Model Layer）
定义本地 MySQL 表结构，对应客户端核心实体：

- local_user：用户配置、偏好设置、登录状态
- cached_video：缓存视频元数据（视频 ID、标题、本地路径、缓存状态、大小、时长等）
- playback_history：播放记录（视频 ID、播放进度、最后播放时间、播放次数）
- favorite_video：收藏视频列表
- local_settings：客户端配置（界面模式、字体大小、是否开启弹幕等）
- sync_log：同步日志（最后一次同步时间、同步状态、失败原因）

（2）数据访问层（Data Access Layer）
提供统一的 Repository/DAO 接口供业务逻辑层调用，封装所有数据库操作：

- UserRepository
- VideoCacheRepository
- PlaybackHistoryRepository
- FavoriteRepository
- SettingsRepository

- SyncLogRepository

（3）数据库连接与事务管理层（Connection & Transaction Manager）

- 管理 MySQL 连接池，控制连接生命周期
- 提供事务支持（ACID），确保数据一致性
- 支持连接重试、超时与故障转移

（4）数据同步服务层（Data Sync Service）

- 负责与服务器端数据同步（通过 REST API 或 WebSocket）
- 增量同步策略：基于 last_updated_at 时间戳
- 冲突处理：客户端优先或时间戳优先策略
- 支持断点续传与后台静默同步

（5）缓存管理模块（Cache Manager）

- 管理视频文件的本地存储与清理策略（如 LRU、缓存大小上限）
- 提供接口供业务逻辑层调用，控制视频下载、删除、查询缓存状态

（6）数据安全模块（Data Security Module）

- 敏感数据加密存储（如用户令牌、偏好设置中的隐私选项）
- 使用 MySQL 加密函数或应用层加密（AES-256）
- 访问权限控制：每个客户端实例使用独立数据库用户，权限最小化

2.3.2 与其它层的交互（封闭式分层）

- 对业务逻辑层的服务接口：

业务逻辑层通过 Repository 接口访问数据库，所有操作均封装在 DAO 中。
示例：业务逻辑层调用 VideoCacheRepository::addCachedVideo(video) 来存储缓存记录。

- 对表示层的隔离：

表示层不能直接访问数据库，必须通过业务逻辑层代理。

- 对服务器端的同步接口：

数据同步服务通过 HTTPS 调用服务器端 REST API，进行增量数据拉取与推送。

2.3.4 数据同步机制设计

1. 增量同步策略
    - 每张表包含 sync_version 字段，记录最后修改版本
    - 客户端定期向服务器发送同步请求，携带本地最后同步版本号
    - 服务器返回该版本号之后有变更的记录
2. 冲突解决策略
    - 采用“客户端优先”策略，本地修改优先保留
    - 冲突记录记录到 conflict_log 表，支持用户手动解决
3. 断点续传与后台同步

- 同步过程记录状态，支持网络中断后恢复
- 后台服务在Wi-Fi环境下自动执行同步

2.3.5 部署与运行模型

- 数据库层作为客户端进程（ClientProcess）的一部分运行
- MySQL以嵌入式模式（SQLite兼容模式）或本地服务模式运行
- 数据同步服务为后台常驻线程，支持网络状态监听与自动同步

第6章详细设计

6.1. 类模型详细设计

1.PlayVideoUI

操作	数据类型	可见性	说明
startInterface（）	void	+	在社区成员打开系统时，打开播放界面
setupVideoPlayerUI（）	void	+	**(QWidget *parent, QWidget *&videoPlayerWidget, QToolButton *&returnButton)** **初始化 UI 组件**
showVideos（）	void	+	在首页展示所有视频的首页以及标题，作者，顺序排列方便用户浏览
selectVideo（）	void	+	用户看到感兴趣的视频，选择是=该视频播放
clikCommentSection（）	void	+	在播放视频时，弹幕在视频上方飘过，用户可以点击选择弹幕对它进行操作
inputComment（）	void	+	输入评论
inputDanmaku（）	void	+	输入弹幕
showCommentInformation（）	void	+	展示评论
showDanmakuInformation（）	void	+	展示弹幕
showCommentSection（）	void	+	展示评论区，用户可以看到其他用户在观看视频时留下的评论

2.PlayVideo

操作	数据类型	可见性	说明
setVideoSource(source)	void	+	**const QUrl &source, 设置视频源**
play()	void	+	**播放视频**
pause()	void	+	**暂停视频**
stop()	void	+	**停止视频播放**
setVolume()	void	+	**将 0-100 的音量值转换为 0.0-1.0 比例值；设置音频输出音量；发送音量状态信号**
setDownloadUrl(url)	void	+	**const QString &url，设置视频下载链接；若 UI 控制器存在且链接非空，启用下载按钮**

getDownloadUrl()	QString	+	**返回当前存储的视频下载链接**
isPlaying()	bool	+	**获取播放状态**
setUIController(uiController)	void	+	**PlayVideoUI *uiController, 设置 UI 控制器**
resumeVideo	void	+	协调视频恢复。委托 Player 对象从暂停状态继续播放。
submitDanmaku	void	+	协调弹幕提交。接收弹幕内容，委托 Danmaku 对象进行校验与持久化，并协调通知服务。
submitComment	void	+	协调评论提交。接收评论内容，委托 Comment 对象进行校验与持久化，并协调通知服务。
likeVideo	void	+	协调点赞操作。记录用户点赞行为，并委托服务更新视频点赞计数及通知作者。
reportVideo	void	+	协调举报流程。接收举报信息，委托审核服务进行处理。

3.PbulishVideoUI

属性	数据类型	可见性	说明
currentUser	User	private	当前登录的用户对象，用于标识发布视频的用户。
videoFile	File|null	private	用户选择的视频文件对象
videoTitle	string	private	用户输入的视频标题
uploadProgress	double	private	视频上传的进度信息

操作	数据类型	可见性	说明
startInterface()	void	public	初始化视频发布界面
selectVideoFile()	void	public	用户选择视频文件
inputTitle(title: string)	void	public	参数: title – 用户输入的视频标题，类型为字符串。 描述: 用户输入视频标题。
uploadVideo()	void	public	上传视频文件和标题到服务器。
noticeSuccess(videoID: string)	void	public	参数: videoID – 服务器返回的视频 ID，类型为字符串。 描述: 通知用户视频上传成功，并显示视频 ID 或链接

4.PublishVideo

操作	数据类型	可见性	说明
uploadVideo	void	+	编排视频上传与转码流程。协调文件校验、存储上传、启动转码及元数据保存

			等一系列操作。
SubmitForReview	void	+	协调内容审核提交。在视频转码完成后，将其提交至审核服务，并根据结果协调后续发布或驳回流程。
cancelUpload	void	+	协调上传取消。终止正在进行的上传或转码任务，并清理相关临时数据。
setVisibility	void	+	协调可见范围设置。根据用户选择，委托 Video 对象更新视频的可见性策略（如公开、私密等）。

5.Manage

属性	数据类型	可见性	说明
ID	string	private	管理员 ID
acount	string	private	管理员账户
password	string	private	管理员账户密码
videoReviewStatusMap	Map<string, string>	private	视频审核状态映射，记录视频 ID 与审核状态的对应关系

操作	数据类型	可见性	说明
login(account: string, password: string)	bool	public	参数: account – 管理员账户名，类型为字符串。 password – 管理员密码，类型为字符串。 描述: 管理员登录验证。
reviewVideo(videoID: string, status: string)	bool	public	参数: videoID – 要审核的视频 ID，类型为字符串。 status – 审核状态（如“通过”、“拒绝”），类型为字符串。 描述: 审核视频，并设置审核状态。
getUserVideos(userID: string)	Video[]	public	参数: userID – 要查询的用户 ID，类型为字符串。 返回值: 返回指定用户被分配的视频对象列表，类型为 Video[]。 描述: 获取指定用户被分配的视频列表。

6.Video

属性	数据类型	可见性	说明
videoId	std::string	private	视频 ID
m_author	QString	private	存储视频作者（格式为 “UP 主：作者名”）
m_title	QString	private	存储视频标题
m_thumbnailUrl	QString	private	存储视频缩略图的网络 URL

m_downloadUrl	QString	private	存储视频的下载 URL
m_videoUrl	QString	private	存储视频的源播放 URL
m_thumbnailPixmap	QPixmap	private	存储加载完成后的缩略图图像数据
m_thumbnailLabel	QLabel*	private	用于显示缩略图的 UI 标签控件
m_titleLabel	QLabel*	private	用于显示视频标题的 UI 标签控件
m_authorLabel	QLabel*	private	用于显示作者信息的 UI 标签控件（代码中注释未启用，但头文件已定义）
m_downloadButton	QPushButton*	private	下载功能按钮（是否显示由 m_showDownloadButton 控制）
m_thumbnailLoaded	bool	private	标记缩略图是否加载完成（true = 已加载）
m_showDownloadButton	bool	private	控制是否显示下载按钮（true = 显示，false = 隐藏）
m_thumbnailGenerator	VideoThumbnailGenerator*	private	用于从视频源生成缩略图的工具类实例
description	std::string	private	视频简介
tags	std::vector<std::string>	private	话题标签
fileUrls	std::unordered_map<std::string, std::string>	private	视频文件的多分辨率地址
duration	int	private	时长(秒)
format	VideoFormat	private	视频格式枚举（mp4/avi/mkv）
status	VideoStatus	private	视频状态枚举，上传成功、通过审核、未通过、删除
visibleScope	VideoVisibleScope	private	视频可见范围枚举
rejectReason	std::string	private	审核失败原因
likeCount	long	private	点赞数
playCount	long	private	播放次数
commentCount	long	private	评论数
collectCount	long	private	收藏数
danmakuCount	long	private	弹幕数
createTime	std::string	private	创建时间，格式：YYYY-MM-DD HH:MM:SS
updateTime	std::string	private	更新时间，格式：YYYY-MM-DD HH:MM:SS
publishTime	std::string	private	发布时间，视频状态变为 published 时设置

passTime	std::string	private	视频审核通过时间
products	std::vector<std::string>	private	关联商品 ID 列表，存储为 JSON 数组

操作	数据类型	可见性	说明
Video()	void	public	const QString &title（默认空）、const QString &author（默认空）、const QString &thumbnailUrl（默认空）、const QString &downloadUrl（默认空）、bool showDownloadButton（默认 false）、const QString &videoUrl（默认空）、QWidget *parent（默认 nullptr） 初始化视频控件： 1. 根据 showDownloadButton 设置控件固定大小（显示下载按钮 200x280，否则 200x250）； 2. 创建 UI 布局和控件（缩略图、标题标签）； 3. 加载默认缩略图或占位文本； 4. 初始化成员变量
Video(id, userId, title, fileUrls, duration, format)	void	public	带参数构造函数，初始化基本属性
getVideoId()	std::string	public	获取视频 ID，返回 std::string
getAuthor()	QString	public	返回视频作者（m_author）
getTitle()	QString	public	返回视频标题（m_title）
getThumbnailUrl()	QString	public	返回缩略图 URL（m_thumbnailUrl）
getDownloadUrl	QString	public	返回下载 URL（m_downloadUrl）
getVideoUrl	QString	public	返回视频源 URL（m_videoUrl）
getDescription()	std::string	public	获取视频描述，返回 std::string
getTags()	std::vector<std::string>	public	获取标签列表，返回 std::vector<std::string>
getFileUrls()	std::unordered_map<std::string, std::string>	public	获取文件 URL 映射，返回 std::unordered_map<std::string, std::string>
getDuration()	int	public	获取视频时长，返回 int
getFormat()	VideoFormat	public	获取视频格式，返回 VideoFormat
getStatus()	VideoStatus	public	获取视频状态，返回 VideoStatus
getVisibleScope()	VideoVisibleScope	public	获取可见范围，返回 VideoVisibleScope

getRejectReason()	std::string	public	获取拒绝原因，返回 std::string
getLikeCount()	long	public	获取点赞数，返回 long
getPlayCount()	long	public	获取播放次数，返回 long
getCommentCount()	long	public	获取评论数，返回 long
getCollectCount()	long	public	获取收藏数，返回 long
getDanmakuCount()	long	public	获取弹幕数，返回 long
getCreateTime()	std::string	public	获取创建时间，返回 std::string
getUpdateTime()	std::string	public	获取更新时间，返回 std::string
getPublishTime()	std::string	public	获取发布时间，返回 std::string
getPassTime()	std::string	public	获取通过时间，返回 std::string
getProducts()	std::vector<std::string>	public	获取商品列表，返回 std::vector<std::string>
setTitle(title)	void	public	const QString &title，设置视频标题，并更新 m_titleLabel 的显示文本
setAuthor(author)	void	public	const QString &author, 设置视频作者（格式化为 “UP 主：作者名”），并更新 m_authorLabel（代码中注释未启用）
setThumbnail(thumbnailUrl)	void	public	const QString &thumbnailUrl, 设置缩略图 URL，显示 “加载中” 占位文本，生成默认灰色占位图（实际网络加载未实现）
setDownloadUrl(downloadUrl)	void	public	const QString &downloadUrl, 设置视频的下载 URL，更新 m_downloadUrl 成员
setShowDownloadButton(show)	void	public	bool show, 设置是否显示下载按钮（控制 m_showDownloadButton 标志）
setVideoUrl(videoUrl)	void	public	const QString &videoUrl, 设置视频的源播放 URL，更新 m_videoUrl 成员
loadThumbnailFromVideo()	void	public	调用 VideoThumbnailGenerator 从视频源 URL 生成缩略图： 1. 若 m_videoUrl 非空，创建缩略图生成器； 2. 连接生成器的信号（缩略图就绪 / 错误）； 3. 显示 “生成缩略图中” 提示

setDescription(desc)	void	public	desc: std::string，设置视频描述，自动更新 updateTime
setTags(tagList)	void	public	tagList: std::vector<std::string>，设置标签列表，自动更新 updateTime
setVisibleScope(scope)	void	public	scope: VideoVisibleScope，设置可见范围，自动更新 updateTime
setStatus(newStatus)	void	public	newStatus: VideoStatus，设置视频状态，自动更新 updateTime；若状态变为 published，设置 passTime 和 publishTime
setRejectReason(reason)	void	public	reason: std::string，设置拒绝原因，自动更新 updateTime
incrementPlayCount()	void	public	播放次数加 1，自动更新 updateTime
incrementLikeCount()	void	public	点赞数加 1，自动更新 updateTime
decrementLikeCount()	void	public	点赞数减 1（保证不小于 0），自动更新 updateTime
incrementCommentCount()	void	public	评论数加 1，自动更新 updateTime
decrementCommentCount()	void	public	评论数减 1（保证不小于 0），自动更新 updateTime
incrementCollectCount()	void	public	收藏数加 1，自动更新 updateTime
decrementCollectCount()	void	public	收藏数减 1（保证不小于 0），自动更新 updateTime
incrementDanmakuCount()	void	public	弹幕数加 1，自动更新 updateTime
decrementDanmakuCount()	void	public	弹幕数减 1（保证不小于 0），自动更新 updateTime
addProduct(productId)	void	public	productId: std::string，添加关联商品，自动更新 updateTime
removeProduct(productId)	void	public	productId: std::string，移除关联商品，自动更新 updateTime
addFileUrl(resolution, url)	void	public	resolution: std::string, url: std::string，添加文件 URL，自动更新 updateTime
removeFileUrl(resolution)	void	public	resolution: std::string，移除指定分辨率文件 URL，自动更新 updateTime
isVisibleToUser(viewerId, isFan)	bool	public	viewerId: std::string, isFan: bool，检查视频对指定用户是否可见，基于 visibleScope
isPublished()	bool	public	检查视频是否已发布，返回 bool
isPending()	bool	public	检查视频是否待审核，返回 bool
isReviewing()	bool	public	检查视频是否审核中，返回 bool
isRejected()	bool	public	检查视频是否被拒绝，返回 bool
isDeleted()	bool	public	检查视频是否已删除，返回 bool
getFormattedDuration()	std::string	public	获取格式化的时长字符串（HH:MM:SS），返回 std::string

toJson()	nlohmann::json	public	将视频对象转换为 JSON 格式，返回 nlohmann::json
fromJson(jsonObj)	static Video	public	jsonObj: nlohmann::json，从 JSON 对象创建视频对象（静态方法），返回 static Video

7.Danmaku

属性	数据类型	可见性	说明
title	String	private	存储用户输入的弹幕文字
whichVideoID	Long	private	标识该弹幕属于哪个视频，用于数据关联与查询
inputTime	LocalDateTime	peivate	记录弹幕发送时间

操作	数据类型	可见性	说明
getDanmaku()	String	public	无参数，返回弹幕的原始文本内容，便于直接日志输出
getDanmakuInformation()	Map<String, Object>	public	无参数，返回包含弹幕所有元数据的结构化对象，便于前端或 API 调用者获取完整信息

8.CommunityMember

属性	数据类型	可见性	说明
userid	string	private	用户唯一标识
role	enum('user','creator','merchant','admin')	private	用户角色
status	enum('normal','warned','muted','banned')	private	账户状态
violationCount	int	private	违规次数，用于自动状态转换

操作	数据类型	可见性	说明
login(account: string, pwd: string)	bool	public	登录验证
getRole()	enum	public	获取用户角色
getStatus()	enum	public	获取当前状态
canPublish()	bool	public	是否允许发布视频（状态正常且非禁言）
canComment()	bool	public	是否允许评论/弹幕（状态正常且非禁言）

updateViolation(count: int)	void	private	更新违规次数（内部触发状态变更）

9.Comment

属性	数据类型	可见性	说明
userId	string	private	发布评论的用户 ID
videoID	string	private	关联的视频 ID
title	string	private	评论标题或内容摘要
inputTime	LocalDateTime	private	评论发表时间（yyyy-MM-dd HH:mm:ss）

操作	数据类型	可见性	说明
getComment()	string	public	用于列表展示，返回 ID、作者、内容摘要、点赞数等
getCommentInformation()	string	public	用于详情页，返回完整内容、状态、关联视频等

6.2. 设计关联

PbulishVideoUI 与 VideoService 的交互:

1.PublishVideoUI 调用 VideoService 的上传接口，将视频文件和标题发送到服务器。

2.VideoService 处理上传逻辑，并返回上传结果。

PbulishVideoUI 与 CommunityMember 类的关联：

3.PublishVideoUI 的 currentUser 属性引用当前登录的用户对象，用于标识视频发布者。

Manage 与 Video 类的交互:

Manage 类调用 Video 类的相关方法，更新视频的分配信息和审核状态。

Manage 类通过 Video 类查询视频信息。

Manage 与 CommunityMember 类的关联:

Manage 类的 adminUser 属性引用当前登录的管理员用户对象。

Manage 类在分配视频时，需要引用 CommunityMember 类来验证用户是否存在。

Manage 与数据库的交互:

Manage 类需要与数据库交互，以存储和检索管理员账户信息、视频分配信息和审核状态。

6.3. 完整性约束

1. PublishVideoUI 类的完整性约束

属性约束

- currentUser:
  - 必须为有效的 User 对象，不能为 null 或未定义，在界面初始化时确保用户已登

录。

- `videoFile`:
  - 在调用 `uploadVideo` 方法之前，必须确保 `videoFile` 不为 `null`。
  - 可以限制视频文件的大小和类型，例如只允许 MP4 文件，且大小不超过 500MB。
- `videoTitle`:
  - 标题不能为空字符串，且长度应在一定范围内（如 1 到 100 个字符）。

方法约束

- `selectVideoFile`:
  - 确保选择的文件符合视频文件的约束条件（大小、类型等）。
- `uploadVideo`:
  - 在上传之前，检查网络连接状态，确保设备已连接到互联网。
  - 处理上传过程中的错误，如网络中断、服务器错误等，并提供适当的用户反馈。

2. `Manage` 类的完整性约束

属性约束

- `adminUser`:
  - 在执行任何管理操作之前，必须确保 `adminUser` 不为 `null`，即管理员已登录。
- `videoAssignmentMap` 和 `videoReviewStatusMap`:
  - 确保键（视频 ID）在对应的 `Map` 中是唯一的。
  - 可以定期清理这些 `Map`，移除不再需要的旧数据。

方法约束

- `login`:
  - 密码在存储和传输过程中应加密，确保安全性。
  - 可以实施账户锁定策略，防止暴力破解密码。
- `assignVideo`:
  - 在分配视频之前，验证视频 ID 和用户 ID 是否存在于系统中。
  - 确保一个视频不会被分配给多个用户（除非业务逻辑允许共享，此时需要额外处理）。
- `reviewVideo`:
  - 审核状态 `status` 应限制为预定义的值（如“通过”、“拒绝”、“待审核”等）。
  - 确保只有具有审核权限的管理员才能执行此操作。
- `getAssignedVideos`:
  - 返回的视频列表应基于最新的 `videoAssignmentMap` 数据。

- 可以实施缓存策略，但需要确保缓存与数据库的一致性。

3. 数据库交互约束

- 在与数据库交互时，使用事务来确保数据的一致性。例如，在分配视频或更新审核状态时，相关操作应在一个事务中完成。
- 实施适当的索引策略，以优化查询性能，特别是对于经常查询的字段（如视频 ID、用户 ID 等）。

4. 安全性约束

- 除了密码加密外，还应考虑使用 HTTPS 来保护数据传输过程中的安全性。
- 实施适当的访问控制策略，确保只有授权用户才能访问敏感数据或执行关键操作。