

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，人们越来越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 business 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

| | |
|-------------------------|----|
| 摘要..... | 2 |
| 第1章 立项..... | 5 |
| 1.1. 项目起源与提案..... | 5 |
| 1. 起源..... | 5 |
| 2. 提案..... | 5 |
| 1.2. Business Case..... | 5 |
| 1. 内容摘要..... | 5 |
| 2. 问题陈述..... | 6 |
| 3. 解决方案..... | 6 |
| 4. 商业模式画布..... | 6 |
| 5. 成本效益分析..... | 7 |
| 6. 风险评估与应对策略..... | 8 |
| 7. 实施计划..... | 8 |
| 8. 替代方案分析..... | 9 |
| 9. 结论与建议..... | 9 |
| 第2章 愿景..... | 10 |
| 2.1. 问题陈述..... | 10 |
| 1. 问题一..... | 10 |
| 2. 问题二..... | 10 |
| 3. 问题三..... | 10 |
| 2.2. 涉众与用户..... | 11 |
| 1. 涉众..... | 11 |
| 2. 用户..... | 11 |
| 2.3. 用户需求..... | 12 |
| 2.4. 产品概述..... | 13 |
| 1. 产品定位陈述..... | 13 |
| 2. 完整的产品概述..... | 13 |
| 2.5. 产品特性..... | 13 |
| 2.6. 其他产品需求..... | 13 |
| 1. 可应用标准..... | 13 |
| 2. 系统需求..... | 13 |
| 3. 其他..... | 13 |
| 第3章 用况建模..... | 14 |
| 3.1. 术语表..... | 14 |
| 3.2. 系统用况..... | 15 |
| 3.3. 系统用例..... | 15 |
| 1. 用况 1: 消费视频内容..... | 15 |
| 2. 用况 2: 视频发布..... | 18 |
| 3. 用况 3: 内容审核..... | 20 |
| 4. 用况 4: 用户管理..... | 21 |

| | |
|-----------------|----|
| 第 4 章 需求分析..... | 23 |
| 4.1. 健壮性分析..... | 23 |
| 4.2. 交互建模..... | 23 |
| 第 5 章 架构设计..... | 24 |
| 第 6 章 详细设计..... | 25 |
| 后记..... | 26 |
| 参考文献..... | 27 |

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，不论是具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，平台上有人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’这一新品种类的领跑者。”

⚡ 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： 手机成为人们的延伸，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。可以更深刻的挖掘自己感兴趣的领域，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，每一个视频都将是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

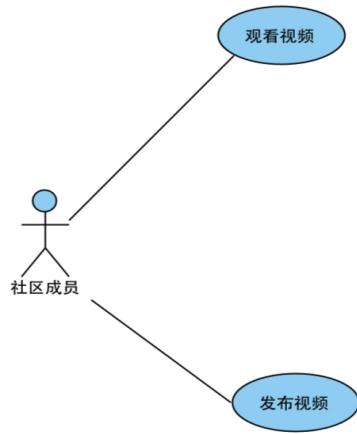
用户手册，安装指南，打包等。

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布的视频可见性规则，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

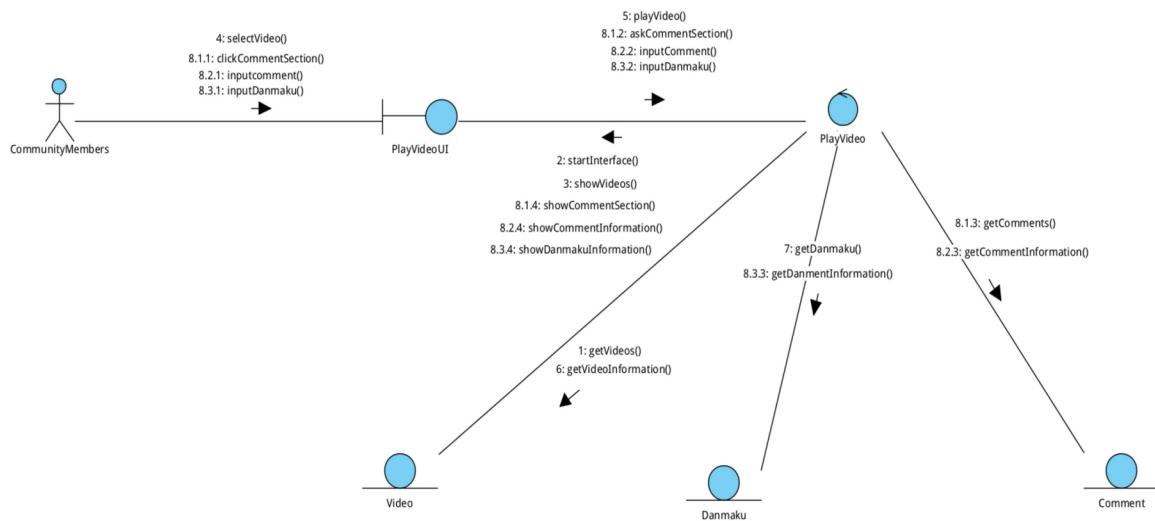
第4章需求分析

4.1. 健壮性分析

4.1.1 通信图

1. 观看视频

sd play video



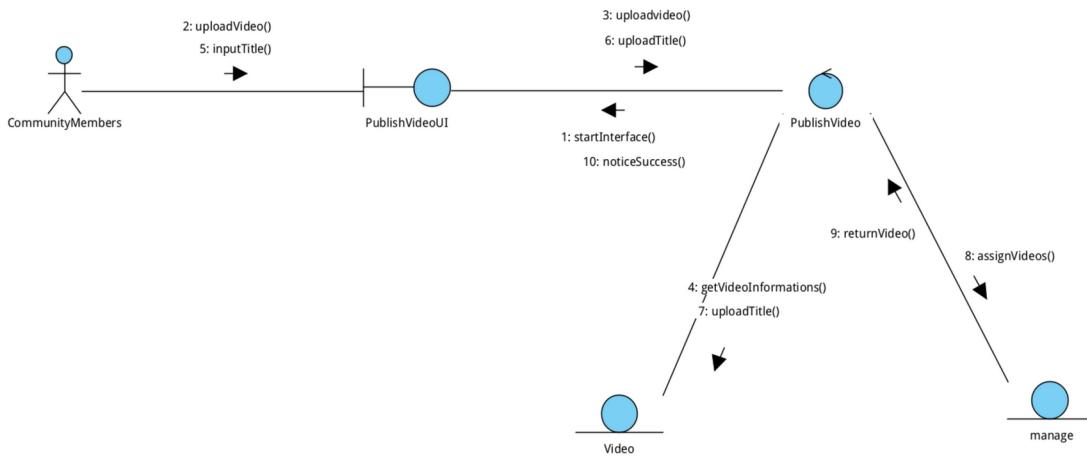
通信图：playVideo

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 Comment 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

sd publish video

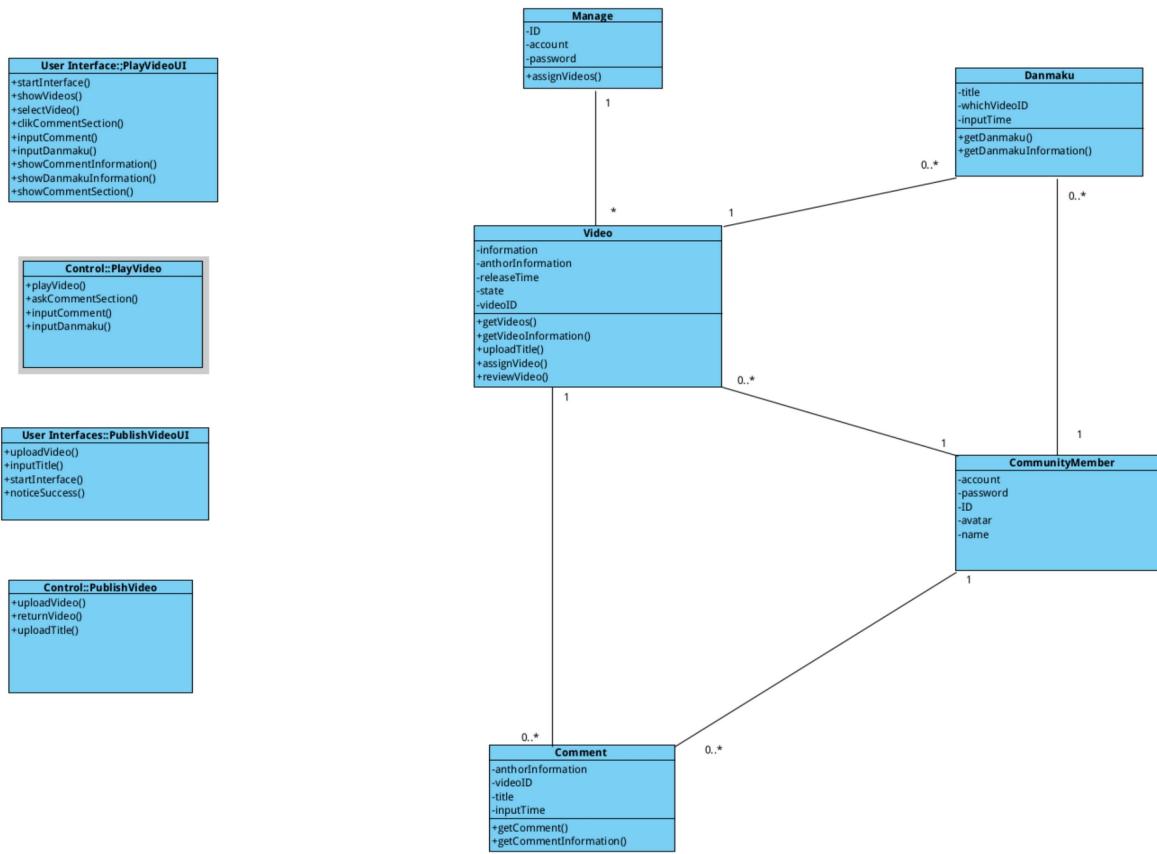


通信图：publishVideo

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



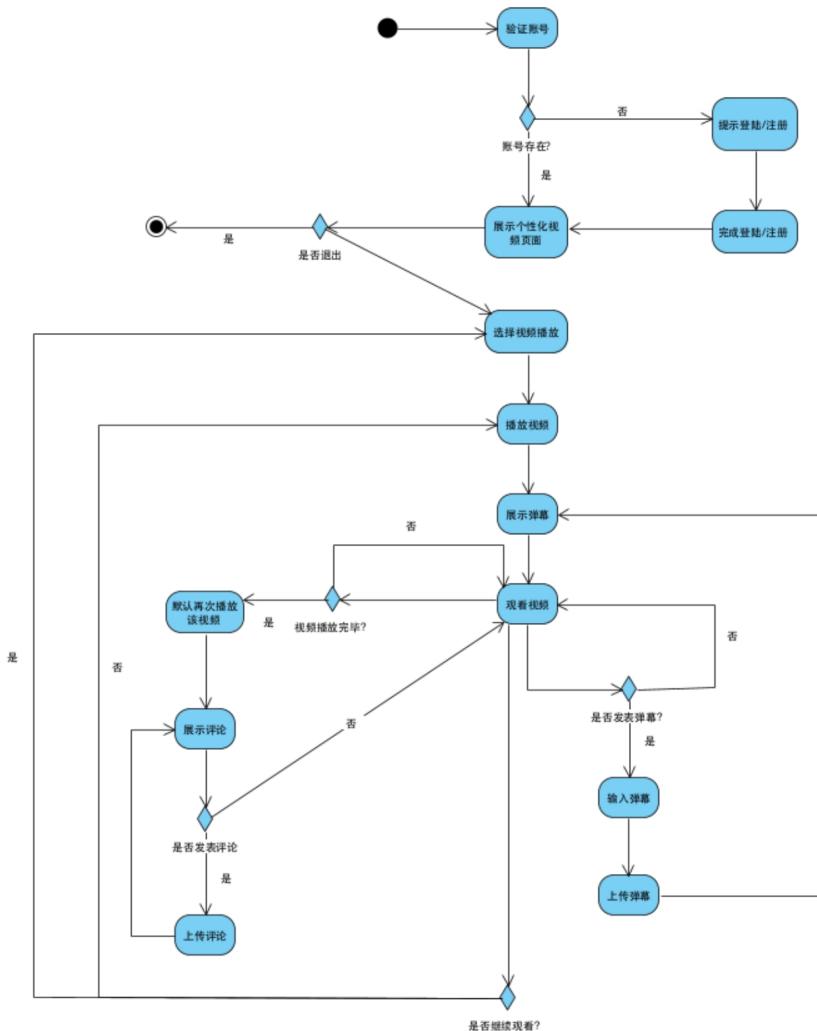
类图：VidShpere

4.2. 交互建模

4.2.1 活动图

1. 观看视频

act [play video]

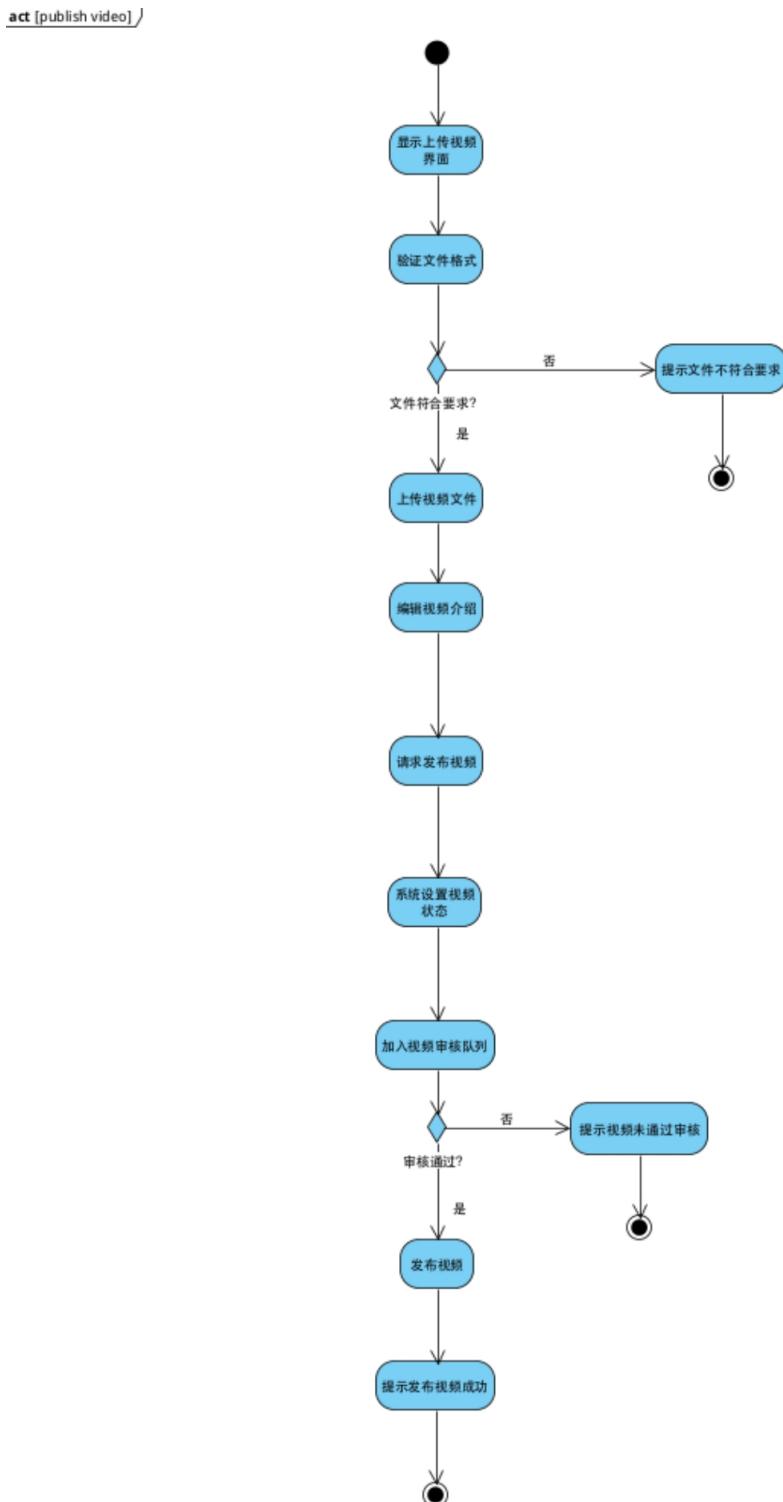


活动图：playVideo

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

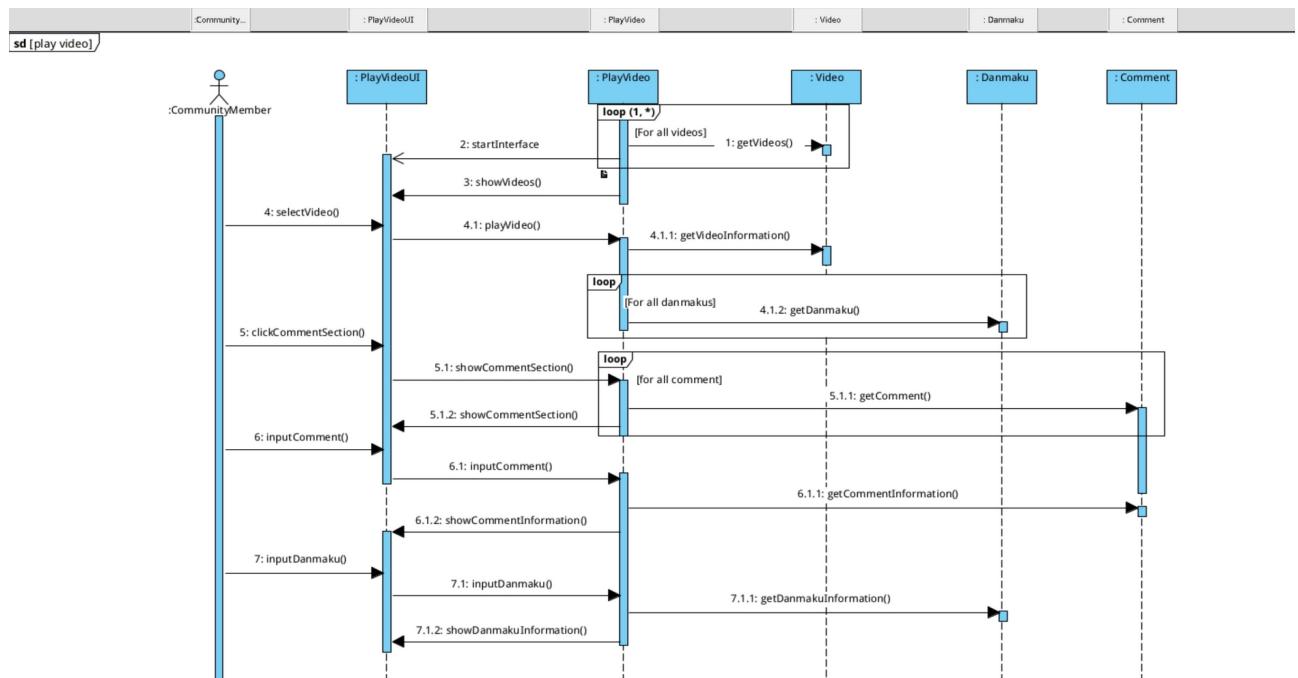


活动图：publishVideo

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):

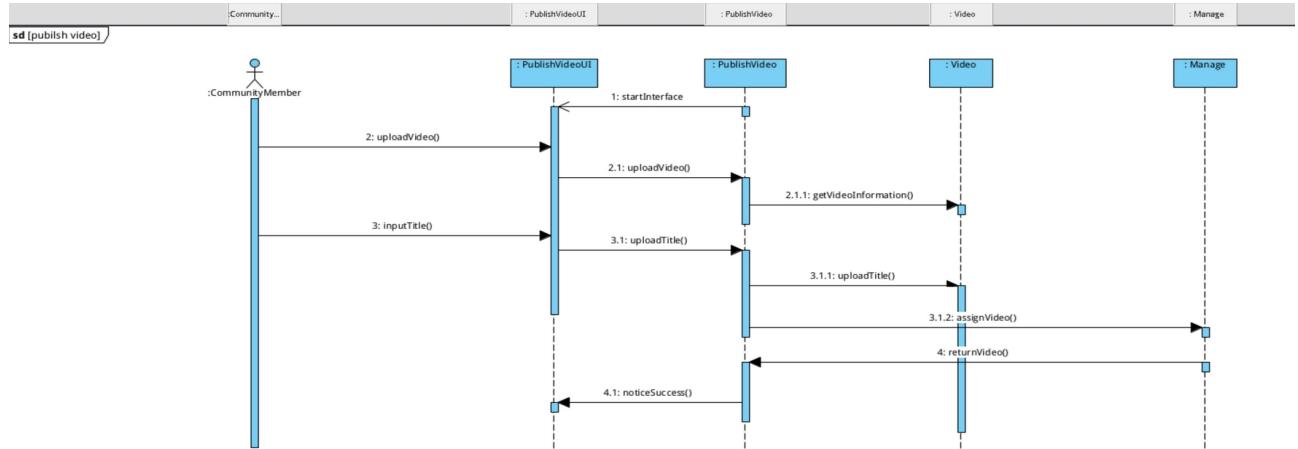
context PlayVideoUI

pre: playVideo()

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

PlayVideoUI.showDanmakuInformation()

2. 发布视频



序列图：publish video

1. 下面是 PublishVideoUI.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 PublishVideoUI.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.assignVideo()

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(), assignVideo()

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

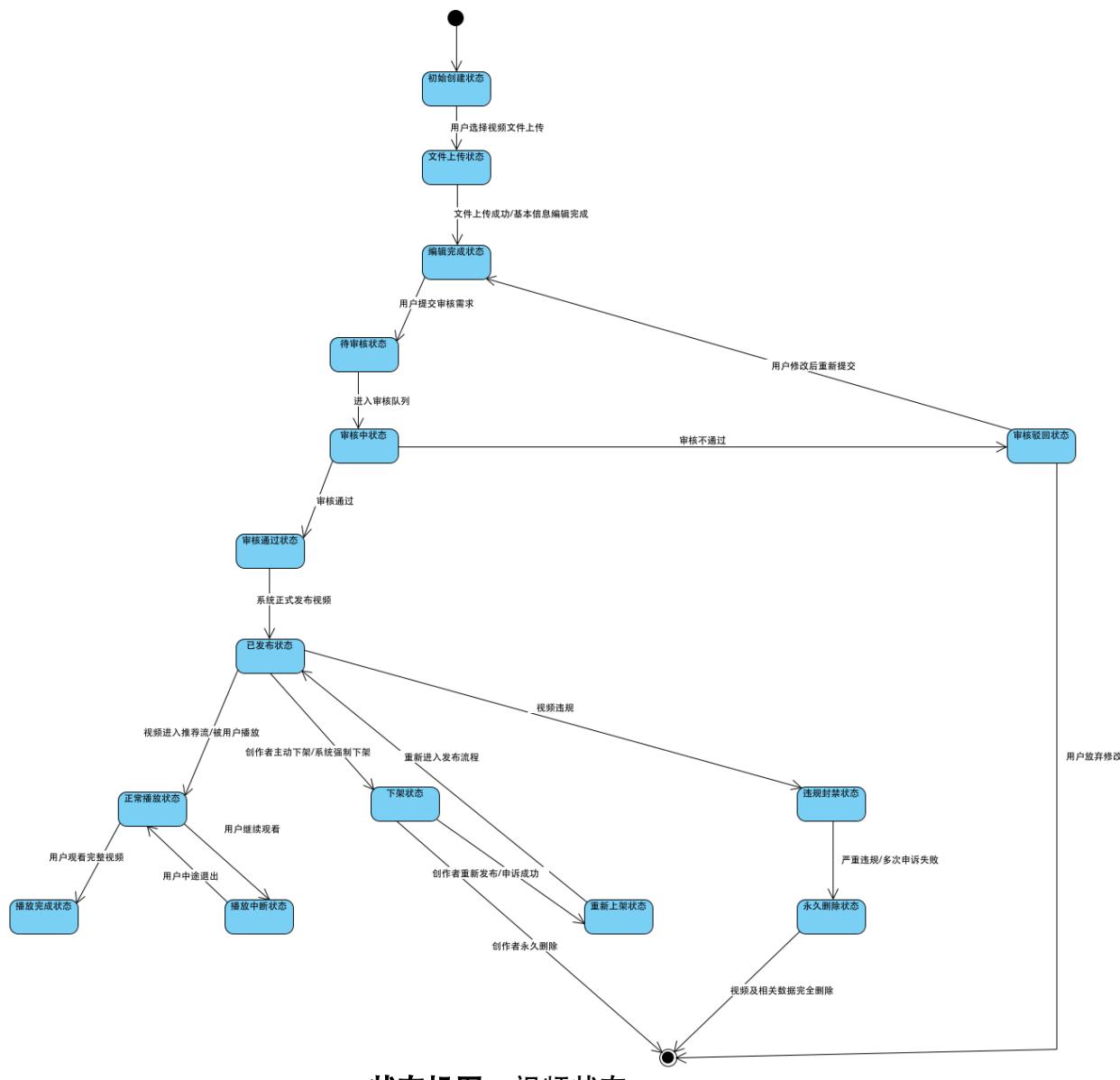
2. 创建评论对象并保存

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 | 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 |
| 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 管理员审核通过 |
| 审核中状态 | 进入管理员审核队列 | 系统正式发布视频 |
| 审核通过状态 | 管理员审核通过 | 用户修改后重新提交 |
| 审核驳回状态 | 审核不通过 | 1. 进入推荐流 2. 进入播放 3. 被下架 4. 视频违规 |
| 已发布状态 | 系统正式发布视频 | 1. 用户观看完整视频 2. 用户中途退出 - 用户继续观看 |
| 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 | - |
| 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 重新进入发布流程 |
| 重新上架状态 | 创作者重新发布 | 严重违规/多次申诉失败 |
| 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

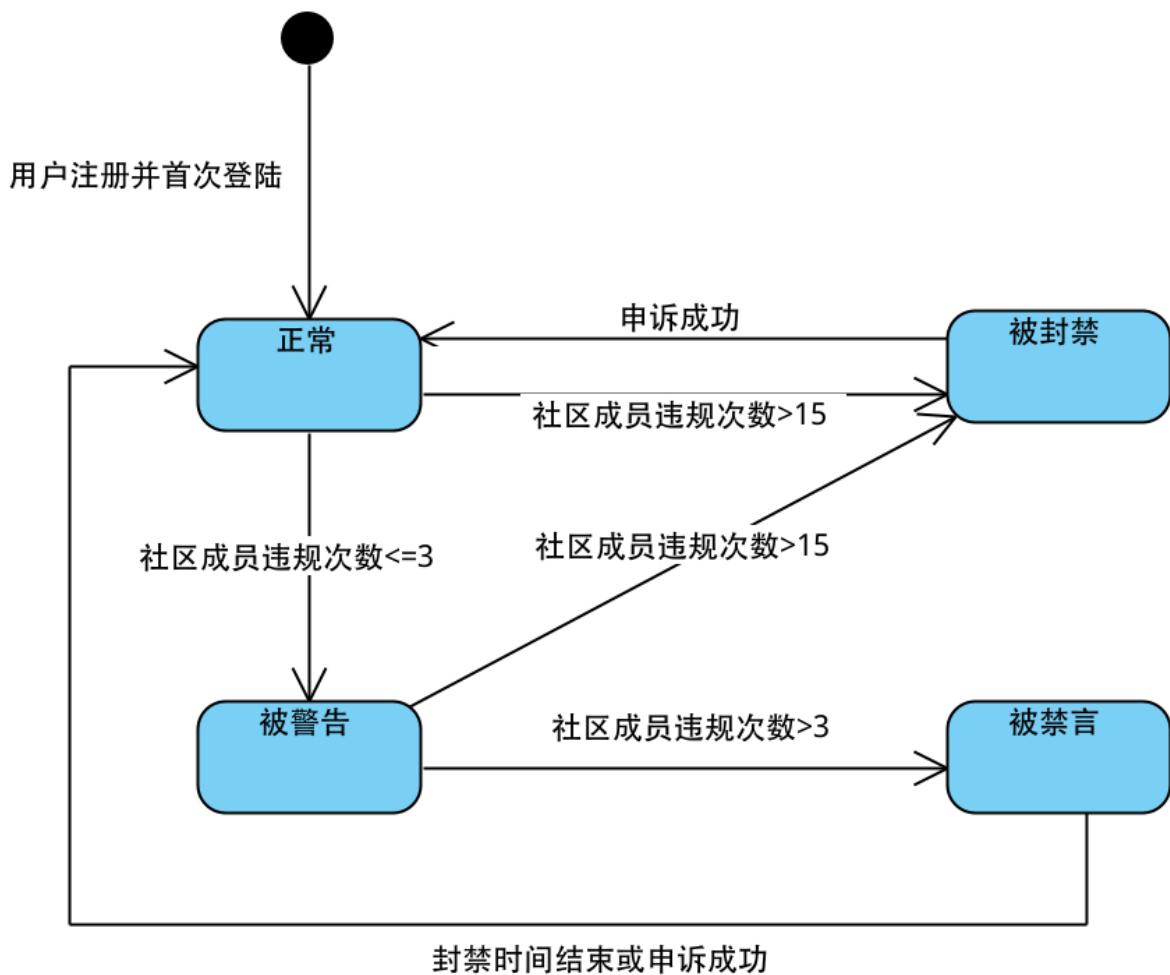
14. 违规封禁状态 - 视频因严重违规被封禁

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

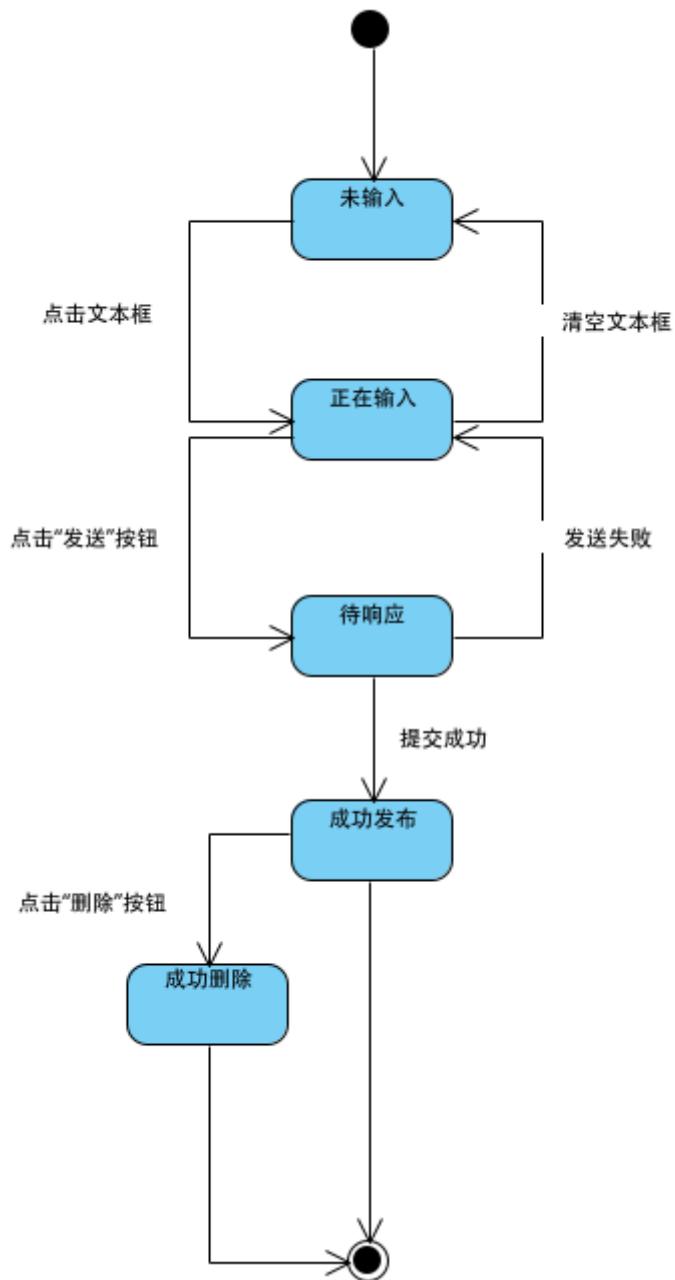
5. 被警告 → 被封禁：

- 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 |
| 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 | 删除成功完成 | 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. 架构风格与核心模式(架构风格：选一个/结合)

14.12 常见的架构风格

在现有的系统里，经常会看到一些原型架构风格。其中每一个都与某一类系统配合得很好。如果我们有一项具有相似特性的应用，就可以通过使用对应的架构而节省工作量，或者至少把这种架构用作设计的出发点。下面列出了几类系统。

- 批处理转换。在整套输入集合上只会执行一次的数据转换（14.12.1节）。
- 连续型转换。当输入发生变化时会连续执行的数据转换（14.12.2节）。
- 交互式界面。由外部交互所支配的系统（14.12.3节）。 [256]
- 动态仿真。模拟现实世界对象演化的系统（14.12.4节）。
- 实时系统。由严格的时间约束所支配的系统（14.12.5节）。
- 事务管理器。与存储和更新数据相关的系统，经常要包括对不同物理位置的并发访问（14.12.6节）。

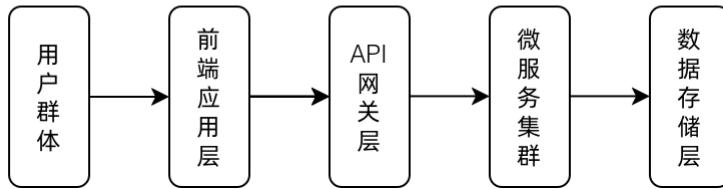
这并不是已知系统和架构的完整清单，只是列出了常见的形式。某些问题需要一种新的架构，但大多数问题可以使用现有风格，或者至少使用现有风格的变体。许多问题组合了这些架构的方面。

Soni 等人确认了软件架构的 4 个不同方面，如表 13-1 所示。

表 13-1 Soni 等人确认的软件架构的 4 个不同方面

| 架 构 类 型 | 元 素 示 例 | 关 系 示 例 |
|---------|------------|---------|
| 概念性架构 | 组件 | 连接符 |
| 模块架构 | 子系统、模块 | 导出、导入 |
| 代码架构 | 文件、目录、库 | 包含、包括 |
| 执行架构 | 任务、线程、对象交互 | 使用、调用 |

- 核心架构组合：分层架构 + 微服务架构 + 事件驱动架构（EDA）
- 前端架构模式：MVC 架构（Model-View-Controller）
- 部署架构模式：分布式部署架构（跨可用区）
- 选型理由：C++ 语言在高性能、低延迟场景具备天然优势，分层架构实现职责隔离，微服务拆分支持核心服务独立扩容（如视频转码、实时互动），事件驱动架构适配弹幕 / 评论实时推送，三者结合满足全龄化、高并发、低延迟的核心业务需求。



1.1. 架构图详细描述（技术暂定）

1. **用户群体**: 包含普通用户（全年龄段）、创作者、管理员三类角色，通过 Web 端、移动端、管理后台等终端访问系统。

2. 前端应用层:

- Web 端（React 18）: PC 端视频浏览、创作发布、电商下单，适配桌面端分辨率。
- 移动端（React Native 0.73）: iOS/Android 双端兼容，支持标准模式与老年模式切换。
- 管理后台（Vue 3）: 管理员内容审核、用户管理、系统配置入口。
- 通信协议: 前端与后端通过 HTTP/HTTPS (RESTful API)、WebSocket (实时互动) 通信，数据格式为 JSON。

3. **API 网关层**: Nginx 1.24 + Kong 3.6 组合，负责请求路由、限流（单 IP 每秒最大 100 次请求）、认证（JWT 校验）、静态资源缓存（视频封面、图标），网关与 C++ 微服务通过 HTTP/HTTPS 协议通信，无需额外适配。

4. **C++ 微服务集群**: 基于 Boost.Asio + gRPC 构建，包含 8 个核心微服务，通过 etcd 实现服务注册发现，Sentinel-C++ 实现熔断降级，线程模型采用“IO 线程池 + 业务线程池”分离设计（IO 线程处理网络通信，业务线程处理计算密集型任务）。

5. **数据存储层**: 各类存储组件通过内网与 C++ 微服务通信，不直接暴露外网；视频文件通过 CDN 分发至用户端，C++ 微服务通过原生客户端驱动访问各类存储。

核心数据流向示例

• 视频播放流程: 用户（移动端）→ 前端应用（加载视频元数据请求）→ API 网关（路由 + 认证）→ C++ 视频处理服务（查询视频地址）→ Redis 缓存（热点视频地址命中）→ CDN（分发视频文件）→ 前端应用（播放视频）→ C++ 实时互动服务（拉取弹幕 / 评论）→ 前端应用（展示互动内容）。

• 弹幕提交流程: 用户（Web 端）→ 前端应用（提交弹幕）→ API 网关（校验身份）→ C++ 实时互动服务（接收弹幕）→ Redis（更新互动计数）→ Kafka（异步持久化消息）→ MongoDB

(弹幕持久化) → C++ 实时互动服务 (WebSocket 广播弹幕) → 前端应用 (实时展示弹幕)。

1.2. 核心技术栈选型 (暂议)

| 技术类别 | 选型方案 | 实战配置 | 选型理由 (C++ 适配优势) |
|---------|--|--|---|
| 前端技术 | React 18、React Native 0.73、Vue 3 | 状态管理 Redux Toolkit, 路由 React Router 6, 老年模式集成百度语音 SDK | 前端与后端解耦, 通过标准协议通信, 不依赖 C++ 开发语言 |
| 后端语言 | C++20 | 编译器 GCC 12.2, 标准库 libstdc++, 编译选项 -O2 -std=c++20 -pthread | 高性能 (CPU 密集型场景比 Java 快 30%+), 内存占用低, 适合视频转码、AI 审核等低延迟需求 |
| 开发框架 | Boost.Asio 1.84、gRPC 1.60 | 网络通信基于 Boost.Asio, 服务间调用 gRPC (Protobuf 3.25), 配置中心 etcd 3.5 | 轻量级无依赖, Boost.Asio 支持高性能异步 IO, gRPC 跨语言通信适配微服务架构 |
| API 网关 | Nginx 1.24 + Kong 3.6 | Nginx 集群 3 节点, Kong 集群 2 节点, 启用限流、熔断插件 | 动态路由配置便捷, 无需修改适配 C++ 微服务, 支持每秒 10 万 + 请求转发 |
| 关系型数据库 | MySQL 8.0 | 连接池 MySQL Connector/C++ 8.0, 主从架构 (1 主 2 从), 连接池大小 1000 | C++ 原生驱动, 性能优于 JDBC, 支持事务与复杂查询, 适配结构化数据存储 |
| 非关系型数据库 | MongoDB 6.0 | 分片集群 (3 分片 + 3 副本), C++ 驱动 mongo-cxx-driver 3.7 | 官方原生驱动, 高并发写入吞吐量 ≥ 1 万 / 秒, 适配弹幕 / 评论等非结构化数据 |
| 缓存系统 | Redis 7.0 | 集群模式 (3 主 3 从 + 3 哨兵), C++ + 客户端 hiredis 1.2.0 | 轻量级高性能, 查询延迟 ≤ 10ms, 支持分布式锁, 适配 C++ 高并发场景 |
| 消息队列 | Kafka 3.6 | 集群 3 节点, C++ 客户端 librdkafka 2.3.0 | 低延迟 (<5ms), 高吞吐 (≥ 10 万消息 / 秒), 持久化保障消息不丢失 |
| 文件存储 | 阿里云 OSS | C++ SDK aliyun-oss-cpp-sdk 3.20.0, 跨区域备份 (华东 → 华北) | 原生支持 C++, 支持分片上传 (最大 5GB 文件), CDN 联动加速 |
| 视频处理 | FFmpeg 6.0 (C++ 绑定) | 编译 FFmpeg 静态库, 支持多线程转码, 分辨率适配 480P/720P/1080P | C++ 直接调用 FFmpeg 内核, 性能优于 JNI 调用, 适配视频转码、封面生成 |
| 监控系统 | Prometheus 2.45 + C++ 客户端 prometheus-cpp 1.1.0, Grafana 10.2 | 监控指标采集间隔 15 秒 | 原生 C++ 客户端, 低开销 (CPU 占用 ≤ 1%), 适配微服务性能监控 |
| 链路追踪 | Jaeger 1.50 | C++ 客户端 jaegertracing-cpp 1.8.0 | 跨服务链路可视化, 支持 gRPC/HTTP 链路追踪, 便于 C++ 微服务问题排查 |

5.2. 详细架构设计

1. 分层架构设计

1.1. 表现层 (UI 层)

- 核心职责：用户交互入口，负责界面展示、用户操作接收与反馈，适配多端与全龄化需求。
- 关键设计 (C++ 适配关联)：
 - 多端适配：核心组件库共享 80%，差异化部分单独开发，前端与 C++ 后端通过标准 JSON 格式通信，无需额外适配。
 - 全龄化适配：老年模式支持大字体、高对比度、语音交互，语音指令通过 HTTP 接口提交至 C++ 用户管理服务处理。
 - 实时互动支撑：WebSocket 客户端采用 Socket.IO，与 C++ 实时互动服务通过标准 WebSocket 协议通信，自动重连机制保障稳定性。
 - BFF 层：Node.js + Express 聚合 C++ 微服务接口，减少前端请求次数，C++ 微服务提供标准化 RESTful 接口供 BFF 层调用。

1.2. 应用层 (C++ 微服务设计)

- 核心职责：封装核心业务流程，衔接表现层与领域层，包含 8 个独立部署的 C++ 微服务，采用 “Boost.Aasio + 线程池 + 消息队列” 架构，服务间通过 gRPC 通信（Protobuf 序列化），对外提供 HTTP/HTTPS 接口。
- 线程模型设计：每个微服务采用 “IO 线程池 + 业务线程池” 分离模式，IO 线程池（4 线程）处理网络 IO（HTTP/WebSocket/gRPC），业务线程池（CPU 核心数 ×2）处理计算密集型任务（视频转码、AI 审核、数据统计），避免 IO 阻塞影响业务处理。
- 核心微服务列表：

| 服务名称 | 核心职责 | 关键接口 (RESTful) | 依赖服务 | 部署配置 |
|--------|-----------------------------|--|-----------------------|---|
| 用户管理服务 | 用户注册 / 登录、权限控制、信息维护、违规管理 | 1. 注册: POST /api/v1/users 2. 权限校验: GET /api/v1/users/auth 3. 信息更新: PUT /api/v1/users/{userId} | 认证服务、MySQL、Redis | 3 实例，CPU 2 核，内存 4GB，IO 线程 4，业务线程 8 |
| 视频处理服务 | 视频上传 / 转码 / 封面生成、元数据管理、状态更新 | 1. 上传: POST /api/v1/videos/upload 2. 视频查询: GET /api/v1/videos/{videoId} 3. 转码回调: POST /api/v1/videos/transcode/callback | OSS、Kafka、Redis、MySQL | 4 实例，CPU 8 核，内存 16GB（转码耗资源），IO 线程 4，业务线程 16 |
| 实时互动服务 | 弹幕 / 评论 / 点赞 / 收藏提交与查询、实时推送 | 1. 弹幕提交: POST /api/v1/interactions/danma ku | Redis、MongoDB、Kafka | 5 实例，CPU 4 核，内存 8GB，IO 线程 4，业务线程 8 |

| 服务名称 | 核心职责 | 关键接口 (RESTful) | 依赖服务 | 部署配置 |
|--------|---------------------------------|---|---------------------------------|---|
| 内容审核服务 | AI 初审？人工复审、违规记录管理 | 2. 评论查询: GET /api/v1/interactions/comm ents 3. 点赞操作: POST /api/v1/interactions/like 1. 审核提交: POST /api/v1/reviews/video 2. 人工复审: PUT /api/v1/reviews/manual 3. 违规查询: GET /api/v1/reviews/violations/ {userId} | TensorFlow C++ API、MySQL、视频处理服务 | 3 实例, CPU 8 核, 内存 16GB (AI 推理耗资源), IO 线程 4, 业务线程 16 |
| 推荐服务 | 个性化视频推荐、热点视频计算 | 1. 推荐列表: GET /api/v1/recommendations 2. 行为上报: POST /api/v1/recommendations/ behavior | MySQL、Redis、Spark C++ API | 3 实例, CPU 4 核, 内存 8GB, IO 线程 4, 业务线程 8 |
| 激励服务 | 播放 / 点赞数据统计、创作者收益计算 / 展示、提现申请处理 | 1. 数据统计: GET /api/v1/rewards/stats 2. 收益查询: GET /api/v1/rewards/balance 3. 提现申请: POST /api/v1/rewards/withdraw | MySQL、Redis、Kafka | 2 实例, CPU 2 核, 内存 4GB, IO 线程 4, 业务线程 8 |
| 认证服务 | JWT 生成 / 校验、token 刷新、权限配置管理 | 1. token 生成: POST /api/v1/auth/token 2. token 刷新: POST /api/v1/auth/refresh 3. 权限配置: PUT /api/v1/auth/permissions | Redis、MySQL | 2 实例, CPU 1 核, 内存 2GB, IO 线程 4, 业务线程 4 |

1.3. 领域层

- 设计原则：基于 C++20 面向对象 + 泛型编程，核心模型采用值语义，避免指针悬空，通过智能指针 (std::unique_ptr/std::shared_ptr) 管理资源，模型与存储解耦，支持序列化 / 反序列化 (JSON/Protobuf)。

- 核心领域模型：

- 用户模型 (User) :

| 字段名 | 类型 (C++ 映射) | 约束 | 说明 |
|----------|-------------|-----------|----------------|
| userId | std::string | 主键 (UUID) | 唯一标识 |
| username | std::string | 非空, 唯一 | 登录账号 (3-20 字符) |
| password | std::string | 非空 | AES-256 加密存储 |

| 字段名 | 类型 (C++ 映射) | 约束 | 说明 |
|----------------|------------------------------|----------|------------------------------------|
| phone | std::optional std::string | 唯一 | 手机号 (可选, 需验证) |
| email | std::optional std::string | 唯一 | 邮箱 (可选, 需验证) |
| role | enum class | 非空 | 用户角色 (user/creator/merchant/admin) |
| status | enum class | 非空 | 账号状态 (normal/frozen/forbidden) |
| violationCount | int32_t | 非空, 默认 0 | 违规次数 (≥15 次冻结) |
| preference | nlohmann::json | 非空 | 偏好设置 (界面模式、暗黑模式等) |
| createTime | std::string | 非空 | 创建时间 (yyyy-MM-dd HH:mm:ss) |
| updateTime | std::string | 非空 | 更新时间 (yyyy-MM-dd HH:mm:ss) |

- 视频模型 (Video) :

| 字段名 | 类型 (C++ 映射) | 约束 | 说明 |
|--------------|------------------------------|-----------|---|
| videoId | std::string | 主键 (UUID) | 唯一标识 |
| userId | std::string | 非空 | 上传者 ID (关联 User) |
| title | std::string | 非空 | 视频标题 (1-50 字符) |
| description | std::optional std::string | 可选 | 视频描述 (≤500 字符) |
| tags | std::vector std::string | 可选 | 话题标签 (最多 10 个) |
| fileUrl | nlohmann::json | 非空 | 多分辨率视频地址 (480P/720P/1080P) |
| coverUrl | std::string | 非空 | 封面图地址 (OSS 存储) |
| duration | int32_t | 非空 | 视频时长 (秒, ≤300) |
| format | std::string | 非空 | 视频格式 (MP4/WebM) |
| size | int64_t | 非空 | 视频大小 (字节) |
| resolution | std::string | 非空 | 原始分辨率 (如 1920×1080) |
| status | enum class | 非空 | 视频状态 (pending/reviewing/published/rejected/takenDown) |
| playCount | int64_t | 非空, 默认 0 | 播放次数 |
| likeCount | int64_t | 非空, 默认 0 | 点赞次数 |
| commentCount | int64_t | 非空, 默认 0 | 评论次数 |
| createTime | std::string | 非空 | 创建时间 |
| updateTime | std::string | 非空 | 更新时间 |

- 互动模型 (Interaction) :

| 字段名 | 类型 (C++ 映射) | 约束 | 说明 |
|---------------|------------------------------|------------------|-------------------------------------|
| interactionId | std::string | 主键 (UUID) | 唯一标识 |
| userId | std::string | 非空 | 操作人 ID (关联 User) |
| videoId | std::string | 非空 | 关联视频 ID (关联 Video) |
| type | enum class | 非空 | 互动类型 (danmaku/comment/like/collect) |
| content | std::optional std::string | 可选 | 互动内容 (仅 danmaku/comment 有值) |
| timestamp | std::optional<int32 _t> | 可选 | 视频时间点 (仅 danmaku 有值, 秒) |
| status | enum class | 非空, 默认 normal | 状态 (normal/deleted) |
| createTime | std::string | 非空 | 创建时间 |

1.4. 数据层

- 核心职责：数据持久化存储与高效访问，C++ 微服务通过原生客户端驱动访问各类存储，封装 DAO (Data Access Object) 模式统一数据访问接口，支持依赖注入便于单元测试。
- 存储方案详情：
 - 关系型数据库 (MySQL / Postgresql) ：
 - 架构：1 主 2 从（主库写入，从库读负载均衡），半同步复制保障数据一致性。
 - 核心表：user (用户) 、video (视频) 、product (商品) 、order (订单) 、review (审核) 。
 - C++ 访问：采用 MySQL Connector/C++ 8.0 客户端，封装连接池（最大连接数 1000），支持事务、批量操作，索引设计优化查询性能。
 - 非关系型数据库 (MongoDB 6.0) ：
 - 架构：3 分片 + 3 副本（分片键：videoId，哈希分片），总存储 2TB。
 - 核心集合：danmaku (弹幕) 、comment (评论) 、user_behavior (用户行为) 。
 - C++ 访问：使用 mongo-cxx-driver 3.7 原生驱动，支持高并发写入，分片集群访问透明化。
 - 缓存系统 (Redis 7.0) ：
 - 架构：3 主 3 从 + 3 哨兵（主从复制，哨兵自动故障转移），单节点内存 8GB，总内存 24GB。
 - 缓存内容：热点视频元数据、用户会话、互动计数、推荐列表。
 - C++ 访问：基于 hiredis 1.2.0 封装客户端，支持连接池、管道操作，淘汰策略 LRU，防止缓存雪崩 / 穿透。

- 文件存储（阿里云 OSS）：
 - 存储空间：视频库（华东）、图片库（华东）、备份库（华北），视频文件默认 3 副本存储。
 - C++ 访问：使用 aliyun-oss-cpp-sdk 3.20.0，封装分片上传（文件>100MB 强制分片）、断点续传、生命周期管理。
- 消息队列（Kafka 3.6）：
 - 架构：3 节点集群，每个节点 CPU 2 核，内存 4GB，存储 500GB。
 - 核心主题：video-transcode（视频转码）、interaction-danmaku（弹幕推送）、interaction-comment（评论同步）、reward-statistics（收益统计）。
 - C++ 访问：基于 librdkafka 2.3.0 封装生产者 / 消费者，支持线程池异步发送 / 接收，消息格式 JSON/Protobuf。

2. 关键子系统详细设计

2.1. 实时互动子系统

- 架构模式：基于 WebSocket+Kafka 的异步处理架构，C++ 适配 Boost.Asio 封装 WebSocket 服务，支持 TCP_NODELAY 优化减少延迟。
- 核心流程：
 1. 用户提交弹幕 / 评论→前端通过 WebSocket 发送至 C++ 实时互动服务→服务验证用户身份与内容合规性。
 2. 服务同步更新 Redis 互动计数（点赞数 / 评论数）→异步发送消息至 Kafka 主题。
 3. Kafka 消费者消费消息→写入 MongoDB 持久化存储。
 4. 服务通过 WebSocket 广播弹幕 / 评论至该视频的所有在线用户（按 videoId 分组管理连接）。
- C++ 适配优势：Boost.Asio 异步 IO 模型支持高并发连接（单实例支持 1 万 + 并发 WebSocket 连接），内存占用低，延迟≤500ms，满足实时互动需求。

2.2. 视频处理子系统

- 架构模式：基于 FFmpeg C++ 绑定的多线程转码架构，结合 OSS 分片上传与 CDN 加速。
- 核心流程：
 1. 用户上传视频→前端分片上传至 C++ 视频处理服务→服务接收分片并合并→上传至 OSS 存储。

2. 服务发送转码任务至 Kafka→转码线程池消费任务→调用 FFmpeg C++ API 进行多分辨率转码（480P/720P/1080P）。
 3. 转码完成后，更新视频元数据（多分辨率地址）→同步至 MySQL 与 Redis→通知用户视频发布成功。
- C++ 适配优势：直接调用 FFmpeg 内核，避免 JNI 中间层开销，多线程转码效率比 Java 高 40%+，支持批量转码任务调度，适配高并发视频上传场景。

2.3. 内容审核子系统

- 架构模式：AI 初审 + 人工复审的二级审核流程，C++ 适配 TensorFlow C++ API 实现 AI 推理。
- 核心流程：
 1. 视频上传完成→视频处理服务提交审核请求至 C++ 内容审核服务。
 2. AI 初审：服务调用 TensorFlow C++ 模型，检测视频标题、描述、画面、音频中的违规内容→生成审核结果（pass/reject/suspect）。
 3. 结果判定：“pass”直接发布，“reject”驳回并通知用户，“suspect”进入人工复审队列。
 4. 人工复审：管理员通过管理后台处理疑似内容→更新审核结果→同步至视频处理服务。
- C++ 适配优势：TensorFlow C++ API 推理延迟比 Python 低 60%+，支持批量推理，适配每日 100 万 + 视频审核需求，AI 初审准确率 $\geq 95\%$ 。

3. 接口设计

3.1. 接口设计规范

- 协议类型：HTTP/HTTPS（RESTful API）、WebSocket（实时互动，<ws://ws.vidsphere.com>）。
- 数据格式：JSON（UTF-8 编码），日期格式 “yyyy-MM-dd HH:mm:ss”，空值返回 null，C++ 端通过 nlohmann::json 库序列化 / 反序列化。
- 接口命名：版本号嵌入 URL（v1/v2），路径采用小写字母 + 下划线，如 “/api/v1/videos/upload”。
- 状态码：HTTP 标准状态码 + 业务错误码，响应体格式统一：

json

{

```

    "code": 200, // 200 成功, 4xx 客户端错误, 5xx 服务端错误, 1xxx 业务错误

    "msg": "success",

    "data": {},

    "errorDetail": "" // 开发环境显示详细错误信息

}

```

- 错误码体系:

| 错误码 | 含义 | 适用场景 |
|------|---------|-------------------|
| 200 | 成功 | 所有接口正常响应 |
| 400 | 参数错误 | 必填参数缺失、格式错误 |
| 401 | 未授权 | 未登录、token 过期 |
| 403 | 禁止访问 | 权限不足 |
| 404 | 资源不存在 | 视频 ID / 用户 ID 不存在 |
| 500 | 服务器错误 | 服务内部异常 |
| 1001 | 视频格式不支持 | 上传格式非 MP4/WebM |
| 1002 | 视频大小超限 | 超过 500MB |
| 1003 | 敏感词违规 | 评论 / 弹幕含敏感词 |
| 1004 | 审核未通过 | 视频内容违规 |

3.2. 核心接口示例

| 接口名称 | 请求方式 | 接口路径 | 请求参数 (必填项标★) | 响应参数 | 错误码 |
|--------|------|--------------------------|--|--|-------------------------|
| 视频播放接口 | GET | /api/v1/videos/{videoId} | ★videoId: string (视频 ID) userId: string (可选, 用于个性化推荐) | { "code":200, "msg":"success", "data":{ "videoId":"vid123456", "title":"周末钓鱼 vlog", "description":"分享钓鱼技巧", "tags":["钓鱼","户外"], "fileUrl":{ "480P":" https://oss.vidsphere.com/videos/vid123456_480P.mp4 ", "720P":" https://oss.vidsphere.com/videos/vid123456_720P.mp4 " } } } | 404 (视频不存在)、500 (服务器错误) |

| 接口名称 | 请求方式 | 接口路径 | 请求参数 (必填项标★) | 响应参数 | 错误码 |
|--------|------|------------------------------|---|--|--------------------------------------|
| 弹幕提交接口 | POST | /api/v1/interactions/danmaku | <p>★videoId: string (视频 ID)</p> <p>★userId: string (用户 ID)</p> <p>★content: string (弹幕内容, ≤50 字)</p> <p>★timestamp: number (视频时间点, 秒)</p> | <pre> "danmakuId":"dm789012", "createTime":"2025-01-25 14:30:20", "content":"好厉害!", "timestamp":35, "userName":"小明" </pre> <p>★userId: string (用户 ID)</p> <p>★file: multipart/form-data (视频文件, MP4/WebM, ≤500MB)</p> <p>★title: string (标题, 1-50 字)</p> <p>description: string (可选, ≤500 字)</p> <p>tags: array<string> (可选, 最多 10 个)</p> <p>visibleScope: enum (public/fans/</p> | 1003 (敏感词违规)、403 (禁言状态)、429 (操作频繁) |
| 视频上传接口 | POST | /api/v1/videos/upload | | <pre> "videoId":"vid123456", "uploadId":"up789012", "status":"pending", "tips":"视频已提交, 等待审核" </pre> | 1001 (格式不支持)、1002 (大小超限)、403 (无发布权限) |

| 接口名称 | 请求方式 | 接口路径 | 请求参数 (必填项标★) | 响应参数 | 错误码 |
|--------|------|-----------------------------------|---|--|------------------------------|
| 商品挂载接口 | POST | /api/v1/merchants/products/attach | private, 默认 public ★merchantId: string (商家 ID) ★videoId: string (视频 ID) ★productId: string (商品 ID) ★position: enum (start/middle/end, 挂载位置) | <pre>{ "code":200, "msg":"success", "data":{ "attachId":"at345678", "videoId":"vid123456", "productId":"pid456", "position":"middle", "createTime":"2025-01-25 15:00:00" } }</pre> | 403 (非商家角色)、404 (商品 / 视频不存在) |

4. 数据设计

4.1. 数据模型关系 (ER 图描述)

- 用户 (User) 与视频 (Video) : 一对多关系 (一个用户可上传多个视频, 一个视频归属一个用户), 通过 userId 字段关联。
- 用户 (User) 与互动 (Interaction) : 一对多关系 (一个用户可发起多个互动操作, 一个互动归属一个用户), 通过 userId 字段关联。
- 用户 (User) 与订单 (Order) : 一对多关系 (一个用户可创建多个订单, 一个订单归属一个用户), 通过 userId 字段关联。
- 视频 (Video) 与商品 (Product) : 多对多关系 (一个视频可挂载多个商品, 一个商品可关联多个视频), 通过中间表 video_product (videoId+productId 联合主键) 关联。
- 视频 (Video) 与审核 (Review) : 一对一关系 (一个视频对应一条审核记录), 通过 videoId 字段关联。
- 商家 (Merchant, User 子角色) 与商品 (Product) : 一对多关系 (一个商家可发布多个商品, 一个商品归属一个商家), 通过 merchantId 字段关联。

4.2. 核心表设计

1. 用户表 (user)

| 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 (UUID) |

| 字段名 | 数据类型 | 约束 | 默认值 | 备注 (C++ 映射) |
|-----------------|---|--------|--|----------------------------|
| username | varchar(20) | 非空, 唯一 | - | std::string |
| password | varchar(64) | 非空 | - | std::string (AES 加密) |
| phone | varchar(11) | 唯一 | NULL | std::optional<std::string> |
| email | varchar(50) | 唯一 | NULL | std::optional<std::string> |
| role | enum('user','creator','merchant','admin') | 非空 | 'user' | enum class UserRole |
| status | enum('normal','frozen','forbidden') | 非空 | 'normal' | enum class UserStatus |
| violation_count | int | 非空 | 0 | int32_t |
| preference | json | 非空 | {"interfaceMode": "standard", "darkMode": false} | nlohmann::json |
| create_time | datetime | 非空 | CURRENT_TIMESTAMP | std::string |
| update_time | datetime | 非空 | CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP | std::string |

2. 视频表 (video)

| 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 (UUID) |
| user_id | varchar(32) | 非空, 外键 (user.user_id) | - | std::string |
| title | varchar(50) | 非空 | - | std::string |
| description | text | - | NULL | std::optional<std::string> |
| tags | json | - | NULL | std::vector<std::string> |
| file_url | json | 非空 | - | nlohmann::json (多分辨率地址) |
| cover_url | varchar(255) | 非空 | - | std::string |
| duration | int | 非空 | - | int32_t (秒) |
| format | enum('MP4','WebM') | 非空 | 'MP4' | std::string |
| size | bigint | 非空 | - | int64_t (字节) |
| resolution | varchar(20) | 非空 | - | std::string |
| status | enum('pending','') | 非空 | 'pending' | enum class |

| 字段名 | 数据类型 | 约束 | 默认值 | 备注 (C++ 映射) |
|--------------------|--|----|--|--------------------------|
| | reviewing','published','rejected','takenDown') | | | VideoStatus |
| attach_product_ids | json | - | NULL | std::vector<std::string> |
| visible_scope | enum('public','fans','private') | 非空 | 'public' | enum class VisibleScope |
| play_count | bigint | 非空 | 0 | int64_t |
| like_count | bigint | 非空 | 0 | int64_t |
| comment_count | bigint | 非空 | 0 | int64_t |
| create_time | datetime | 非空 | CURRENT_TIMESTAMP | std::string |
| update_time | datetime | 非空 | CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP | std::string |

3. 互动表 (interaction)

| 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 (UUID) |
| 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 |
| type | enum('danmaku','comment','like','collect') | 非空 | - | enum class InteractionType |
| content | varchar(500) | - | NULL | std::optional<std::string> |
| timestamp | int | - | NULL | std::optional<int32_t> (秒) |
| status | enum('normal','deleted') | 非空 | 'normal' | enum class InteractionStatus |
| create_time | datetime | 非空 | CURRENT_TIMESTAMP | std::string |

4.3. 数据存储策略

- 数据分区：

- MySQL：用户表按 user_id 范围分区（每 100 万用户一个分区）；订单表按 create_time 按月分区（每月一个分区），C++ 端通过 DAO 层透明化访问分区表。
- MongoDB：评论 / 弹幕集合按 videoId 哈希分片，C++ 驱动自动路由访问对应分片。

- 备份策略：
 - MySQL：每日全量备份 + 每小时增量备份，C++ 微服务通过 OSS SDK 访问备份文件。
 - Redis：RDB+AOF 持久化，C++ 客户端支持故障后数据恢复。
 - OSS 文件：跨区域备份，C++ SDK 支持备份文件读写。
- 数据生命周期：
 - 弹幕 / 评论：MongoDB 中保留 1 年，超过自动归档至 OSS，C++ 端通过 DAO 层适配归档数据访问。

第6章 详细设计

后记

正文内容，方正仿宋，小四，首行缩进。正文内容，方正仿宋，小四，首行缩进。正文内容，方正仿宋，小四，首行缩进。

参考文献

- [1] YOUNG.RSS 是什么? [EB/OL]. <http://jingpin.org/what-is-rss/>.
- [2] 杨博, 彭博.RSS 提要分析与阅读器设计[R].成都: 四川大学计算机学院, 2007: 42-43.
- [3] 逸出络然.RSS 技术的原理[EB/OL].<http://yclran.blog.163.com/blog/static/979454962009111034111558/>.
- [4] 佚名.Qt 是什么[EB/OL]. <http://qt.nokia.com/title-cn>.
- [5] 佚名.Model/View Programming[EB/OL]. <http://doc.trolltech.com/4.6/model-view-programming.html>.
- [6] [加拿大]Jasmin Blanchette[英]Mark Summerfield 著 闫锋欣,曾泉人,张志强译.
- [7] C++ GUI Qt4 编程 (第二版) [M].电子工业出版社: 2008:182-206,291-305.
- [8] 佚名.XML Processing[EB/OL]. <http://doc.trolltech.com/4.6/xml-processing.html>.
- [9] Michael Blala James Rumbangh 著.UML 面向对象建模与设计 (第 2 版) [M].北京: 人民邮电出版社,2006:136-235.
- [10]胡海静,王育平,等. XML 技术精粹[M]. 北京: 机械工业出版社, 2001:17-19.