WeChat 项目文档

项目小组 WeChat 小组

小组成员 张键 张慧芝 李露露 李雪怡 陈旭卓 陈瑞

联系方式 19923235594

重庆师范大学软件工程系

摘要

本项目源于主流社交软件功能冗余、广告泛滥、隐私泄露等用户痛点，针对 3.8 亿追求“纯粹、高效、轻量化” 社交体验的用户需求，立项开发无广告、去冗余、强核心的一体化即时通信平台 WeChat。

项目核心目标是打造 “打开即聊、聊完即退” 的工具型产品，聚焦跨端文本 / 语音 / 视频聊天、朋友圈互动及点对点数字支付三大核心功能，覆盖 25-40 岁职场人群、18-24 岁学生群体及 40 岁以上用户，目标占据 15% 极简需求市场份额（约 5700 万用户），确立细分领域领导地位。

开发过程以 “功能硬边界” 为原则，优先保障核心功能稳定与合规性，采用 “稳态核心→可观测→灰度→扩容” 的架构节奏，通过精准营销与低迁移成本方案吸引用户，同步建立风险防控机制应对市场竞争、获客成本及合规风险。

项目最终将交付一款跨 iOS、Android、Windows 等多平台的应用，实现核心功能一键触达、操作效率提升 40%、隐私最小化授权等成效，通过用户留存率、客户满意度及市场份额验证成功，后续可探索高级会员订阅等合规变现路径，重塑轻快、安全、可信的社交体验。

关键词：极简社交，即时通信，无广告，核心功能，多端适配

日期	修改	描述	作者
2025 年 9 月 18 日	0.1.0	初始版本	
2025 年 10 月 13 日	0.1.1	大改选题，重新分析	
2025 年 10 月 19 日	0.2.1	更新参与者名称及用况图	
2025 年 10 月 23 日	0.2.2	完成愿景文档	
2025 年 10 月 30 日	0.3.1	用况建模	
2025 年 11 月 22 日	0.4	完善和修改用况模型,并且使用况描述更精简化	
2025 年 12 月 11 日	0.5	健壮性分析，交互建模	
2025 年 12 月 22 日	0.6	架构设计	
2025 年 12 月 30 日	0.7	详细设计	

目录

第1章 立项

1.1. 项目起源与提案

当前社交软件市场正面临深刻的用户体验危机。主流平台在追求“超级 APP”生态扩张的过程中，逐渐偏离了社交工具的核心价值，陷入了功能堆砌与广告泛滥的发展困境，这导致了核心社交体验被稀释，用户不得不面对复杂的操作流程、无处不在的信息干扰以及日益增长的隐私担忧。

市场调研数据清晰地揭示了用户的强烈不满与明确期待。2024 年的行业报告指出，高达 82%的用户对现有聊天软件的广告植入感到困扰，76%的用户认为频繁更新的非核心功能，如小游戏和资讯板块，显著增加了操作的复杂性，更值得关注的是，有 68%的用户明确表示愿意转向一款“仅保留核心功能、无广告”的极简社交软件。这并非零星的抱怨，而是代表了约 3.8 亿中国用户群体对“纯粹、高效、轻量化”社交体验的共同渴望，构成了一个清晰且尚未被满足的市场缺口。

在此背景下，我们正式提出“WeChat”项目。本项目的核心提案是开发并运营一款严格遵循“无广告、去冗余、强核心”定位的一体化即时通信平台。产品将建立清晰的“功能硬边界”，果断摒弃所有广告推送与非核心衍生服务，专注于提供跨端的文字、语音、视频聊天，以及朋友圈互动和点对点数字支付等核心社交功能。我们的目标是打造一款让用户能够“打开即聊、聊完即退”的工具，重新定义轻快、安全、可信的社交体验。

1.2. Business Case

本项目的商业价值立足于一个明确的市场机遇：在主流软件忽视的“极简社交”细分领域，存在一个快速增长且潜力巨大的蓝海市场。当前，微信、QQ 等主导者在该领域的渗透率不足 5%，且尚无用户规模超千万的纯核心功能聊天软件。我们计划通过精准的差异化定位，切入这一市场空白，目标是覆盖至少 15%的极简需求用户，即约 5700 万用户基数，从而确立我们在该细分市场的领导地位。

我们的核心战略是通过极致的用户体验驱动增长。产品以“零广告干扰”、“功能硬边界”和“跨端一致体验”作为核心卖点，直接回应目标用户的核心痛点。这将有力支撑我们的市场进入策略：通过精准的营销传播“纯净社交”概念，并辅以“一键导入通讯录”等低迁移成本方案，快速吸引并转化那些对现有平台感到疲惫的用户群体。我们预期，这种以用户体验为中心的口碑效应将成为用户获取和留存的主要驱动力。

在财务可持续性方面，项目初期将优先追求用户规模和市场份额，而非短期盈利。这并不意味着缺乏商业模式构想。长期来看，可行的变现路径包括探索无广告的高级会员订阅、与核心体验无缝融合的合规增值服务（如大型文件传输券），以及在严格遵循“数据最小化”原则下，未来可能开展的 B 端企业级协同工具授权，这些潜在的收益来源将确保项目在占据市场后实现健康的财务回报。

本项目面临的主要风险包括来自巨头的竞争反应、用户获取成本高于预期，以及复杂的合规性要求，尤其是在数据隐私和金融支付领域。我们已经制定了相应的缓解策略：通过极致的产品专注度构建竞争壁垒；依靠清晰的价值主张和口碑营销控制获客成本；并从产品设计之初就将合规性嵌入架构，与可靠的第三方支付伙伴合作，以管控法律与金融风险。

综上所述，本项目旨在抓住一个被现有市场参与者忽视的、具有充分用户需求验证的宝贵商机。我们提议立即立项并启动 WeChat 的开发，集中资源打造稳定、高可用的核心功能。项目的成功将以用户留存率、客户满意度以及最终的市场份额作为关键衡量指标。此项目不仅具备明确的商业价值，更是重塑社交体验、回归沟通本质的一次重要实践。

第2章愿景

2.1. 问题陈述

1. 问题一

要素	描述
问题	现有聊天软件广告（启动页/朋友圈/聊天界面）泛滥，冗余功能占用界面，核心功能查找繁琐。
影响	核心操作效率降 40%，用户每日关 3-5 次弹窗，32% 用户因广告过多有卸载意愿。
结果	WeChat 零广告植入，仅保留核心功能（聊天/支付/朋友圈），核心功能一键触达，效率提升 40%。
优点	体验纯粹无干扰，操作高效便捷，适配极简需求用户。

2. 问题二

要素	描述
问题	主流软件频繁更新非核心功能（虚拟形象/群直播等），功能过载导致使用门槛高。
影响	中老年用户学习成本高，60 岁以上仅 45%能熟练使用核心功能（转账/朋友圈设置）。
结果	WeChat 更新（每年 1-2 次），不新增非核心功能，界面直观，上手即会。
优点	学习成本低，全年龄段适配，尤其解决中老年用户使用难题。

3. 问题三

要素	描述
问题	现有软件衍生功能强制索取隐私数据（位置/浏览记录等），用户被迫授权。
影响	隐私泄露风险提升，2024 年 65%社交软件隐私投诉与此相关。
结果	WeChat 遵循隐私最小授权，仅获取核心功能必需权限，无过度数据收集。
优点	隐私安全有保障，契合用户隐私诉求，增强产品信任度。

2.2. 市场分析

1. 商机

据第三方调研（2024 年社交工具用户体验报告），82% 的用户反馈“现有聊天软件广告过多（如弹窗广告、朋友圈广告）”，76% 的用户认为“频繁更新的非核心功能（如小游戏、资讯板块）增加了操作复杂度”，68% 的用户表示 “愿意尝试‘仅保留核心功能、无广告’的极简社交软件”—— 用户对“去冗余、纯社交” 的需求已形成明确市场缺口。当前主流聊天软件（如微信、QQ）均朝着 “超级 APP” 方向发展，不断叠加生活服务、娱乐、办公等衍生功能，导致核心社交体验被稀释；而小众社交工具多聚焦“垂直社交”（如兴趣社交、职场社交），未针对“大众基础社交的极简需求”开发产品，本产品可凭借“聚焦核心、剔除冗余”的定位，抢占“大众极简社交”的市场。

2. 市场统计

核心市场统计数据如下（数据来源：2024 年中国社交软件行业报告、第三方用户调研机构问卷结果）：

用户需求规模：中国社交软件用户规模达 11.2 亿，其中 “对广告敏感、追求极简体验”的用户群体约 3.8 亿，占比 34%；该群体中，25-40 岁职场人群占比 52%（注重专注力）、18-24 岁学生群体占比 28%（反感信息干扰）、40 岁以上用户占比 20%（追求简单操作），形成明确的目标用户基数。

现有产品体验痛点数据：

广告干扰：82%的用户反馈“主流软件广告影响使用”，其中 70%的用户表示“愿意为无广告的社交软件支付每月 5-10 元的费用”；

功能冗余：68%的用户“从未使用过主流软件中的内置游戏、资讯板块”，但此类功能占据约 30%的界面空间；

操作效率：用户使用主流软件完成“发起转账”的平均步骤为 4.2 步，完成“设置好友朋友圈权限”的平均步骤为 3.8 步，而用户期望的“理想步骤”为 1-2 步。

竞争格局与市场空间：当前主流社交软件（微信、QQ）在“极简社交”领域的市场渗透率不足 5%，尚无用户规模超千万的“纯核心功能”聊天软件；据行业预测，若本产品能覆盖“极简需求用户”的 15%（约 5700 万用户），即可成为细分领域头部产品，且该细分市场的年增长率预计达 25%，高于整体社交软件市场 12%的年均增长率。

2.3. 涉众与用户

1. 涉众

涉众	涉众类型	简要描述
用户（User）	年轻技术使用者、中老年标准用户、数字支付高频用户	广大网民群体，涵盖不同年龄、职业、技术熟练度，通过产品完成通信、社交、支付等操作，是产品核心服务对象
项目团队（Developers）	项目经理、设计师、测试人员、编码员、维护者等	负责产品设计、开发、测试、维护的技术人员，实现通信、好友管理、群聊等功能的技术支撑
发起人（Sponsors）	项目发起人、产品负责人、投资人	为项目提供资金、资源支持，是业务领域的最终决策者
合作伙伴（Partners）	云服务商、第三方 SDK 提供商、表情素材方	提供数字支付接口和服务的机构，是为产品提供关键能力支持的外部合作方
权威（Authorities）	网络安全监管部门、法律顾问	负责互联网通信、数据隐私、支付安全监管的政府部门或行业机构

2. 关键涉众代表角色

DSDM 角色	代表涉众类型	职责
业务愿景制定	产品总监	定义并维护“轻量化、安全、核心通讯”的产品愿景。

者		
业务大使	从重度用户中招募的专职代表、产品经理	全程参与开发过程，提供需求细节，并参与用况建模，对功能验收拥有决策权。
业务顾问	外聘的隐私法律专家、通讯领域专家、支付合规顾问	在特定领域提供专业意见
技术协调员	首席架构师或技术经理	定义技术愿景，做出技术栈决策，统筹技术资源

3. 用户（按使用频率与深度）

在涉众中，用户（Users）是一类特殊的核心群体，他们直接与系统交互，决定系统的可接受性。根据使用频率与深度，可以划分为以下三类典型用户：

用户类型	使用频率	使用深度	行为特征
重度用户	每日高频使用，消息量大、使用多端设备	深度参与群聊、文件传输、跨平台协作	依赖通信工具进行工作或学习；常在移动端与 PC 端切换
中度用户	每日使用多次，沟通以熟人圈为主	聊天、语音通话、朋友圈浏览发布	主要用于日常交流与任务沟通
轻度用户	每周或偶尔使用	文字聊天、图片分享、偶尔支付	偶尔沟通、关注隐私

4. 用户环境分析

用户使用环境直接决定产品的适应性与稳健性，需从设备与系统、网络条件、使用场景三个维度，明确环境约束对功能设计的要求，确保产品在多样化场景下满足核心体验。在设备与系统方面，用户覆盖手机、平板、PC 三类终端，操作系统包含 Android、iOS、Windows，需实现多设备消息无缝同步，如 PC 端已读消息同步至手机端，同时适配不同终端操作逻辑；网络条件上，需应对 Wi-Fi、4G/5G 及低带宽场景，低带宽下自动压缩消息体积、支持离线同步，避免消息延迟或丢失；使用场景涵盖通勤、办公、家庭等，办公场景需强化文件加密与群协作效率，通勤场景优化语音转文字功能，家庭场景提升视频通话清晰度，满足不同场景下的差异化需求。

2.4. 关键涉众和用户的需要

1. 关键涉众需要

涉众类别	需要描述	优先级（MoSCoW 规则）	对应产品能力	验证标准
发起人（Sponsors）	产品快速上线市场、上线后用户量大、核心用户留存率高、成本可控，优先保障核心功能投入、避免冗余开发	Must have（Mo）	用户增长与留存、资源管控	用户统计数据、留存曲线、成本核算表
项目团队	需求边界清晰、技术约束明	Must have	需求管理、	需求变更记

（Developers）	确、有可量化的性能指标	（Mo）	技术架构、测试验证	录、故障统计报告
合作伙伴（Partners）	多端消息同步的低时延与稳定性、接口合规、通过应用商店审核，无违规下架风险	Must have（Mo）	多端同步、支付接口对接	云服务报告、支付成功率统计
权威（Authorities）	数据处理合规合法、通信内容无违规信息、具备违规消息检测与拦截能力、交易记录可追溯	Must have（Mo）	数据合规、违规检测	合规审计证书、消息处理日志

2. 关键用户需要

用户类型	需要描述	优先级（MoSCoW 规则）	对应产品能力	验证标准
重度用户	跨端同步时延≤1 秒、支持≥200MB 文件传输、无广告	Must have（Mo）	多端同步、文件传输、群管理	时延测试报告、传输成功率统计
中度用户	操作简单、语音通话无卡顿、无强制弹窗	Must have（Mo）	音视频通话、朋友圈	通话质量报告、朋友圈界面审计
轻度用户	隐私设置简单、快速上手核心功能、无冗余引导	Must have（Mo）	隐私设置、基础聊天	设置操作测试、用户上手调研

2.5. 产品概述

1. 产品定位陈述

for	追求纯粹、高效、轻量化社交体验的大众用户
who	需要在通信、数字支付、管理朋友圈动态等核心社交场景中摆脱广告弹窗、非必要功能干扰，降低操作复杂度与隐私泄露风险的用户
the	社交聊天即时通讯应用 WeChat
That	聚焦三大核心功能，剔除广告与冗余衍生服务，以“零干扰、高直达、轻操作” 为核心优势，让用户无需筛选干扰信息、无需学习复杂功能，即可快速完成沟通、支付、社交管理等需求

2. 完整的产品概述

2.1. 能力概览

WeChat 是一款专注于核心社交需求的纯粹、轻量级即时通讯应用。它摒弃了“超级 APP”的功能冗余，通过建立清晰的“功能硬边界”，为用户提供无广告、无干扰、高直达的社交体验。系统的核心能力有以下几部分：

1.实时通信：支持一对一的文本、语音、视频通话，提供扫码/手机号添加好友、备注分组、朋友圈权限设置等操作，聚焦于沟通本身，确保连接稳定、传输安全、界面干净。

2.可信数字支付：提供内嵌的、流程闭环的单笔订单支付功能。从发起到入账，全程自动化风控校验，并与在线网银系统安全对接，确保交易可追溯、对账一致。

3.清爽朋友圈：支持动态的发布、好友间的点赞评论互动、以及全面的隐私与历史内容管理，它去除了所有商业化信息流和复杂插件，只保留最纯粹的分享与互动乐趣。

2.2. 客户的效益

WeChat 服务于追求纯粹社交体验的大众用户，其核心效益与特性的关联如下所示：

客户效益	对应产品特性
沟通效率提升	一对一文本消息；图片/文件直传（断点续传、PC 端拖拽）；通信核心元素极简布局（无广告弹窗）
操作成本降低	数字支付；通信界面极简设计
隐私安全保障	数字支付（无衍生服务、不收集非必要数据）；安全合规特性（传输/存储加密、审计留痕）；朋友圈动态（无第三方导流、纯好友互动）
体验纯粹无扰	通信界面（无广告弹窗、功能推荐标识）；朋友圈动态（无广告、无第三方导流）；数字支付（无理财/信贷衍生服务）；取舍原则（核心路径零广告、剔除冗余功能）
社交管理清晰	朋友圈动态（我的动态管理、纯好友互动）；通讯录界面（仅显示好友列表与分组）
支付安全高效	数字支付（转账/收款/账单查询）；支付合规特性（与网银安全对接、交易留痕）

2.3. 假设与依赖

以下假设均为支撑 “无广告、去冗余、强核心” 的产品愿景而存在，任一假设失效将直接动摇愿景可行性；以下依赖均是产品落地与愿景达成的必要前提，需明确其对愿景的影响关联。

1. 假设

   • 目标用户对 “纯粹、高效、轻量化” 的核心诉求具有稳定性，短期内不会因市场趋势变化而转向对 “超级 APP 生态” 的依赖，仍持续排斥广告干扰与功能冗余。

   • 主流移动操作系统（iOS 14.0+/Android 8.0+）及 PC 系统（Windows）的底层 API、推送服务（APNs/FCM）将长期保持兼容与稳定，不会通过技术限制削弱 “跨端一致、核心功能一键触达” 的体验。

   • 目标市场的通信、数据隐私及数字支付相关监管政策，不会出台颠覆性新规（如强制要求植入合规广告、限制极简功能形态），仍允许 “数据最小化”“无衍生服务” 的产品形态存在。

   • 现有网络基础设施（移动网络 / Wi-Fi）能持续支撑 “低延迟、弱网自适应” 的通信需求，用户设备（内存≥2GB、存储≥100MB）能满足产品轻量化运行要求，不会因硬件或网络限制导致核心功能（音视频通话、文件直传）体验降级。

   • 用户对数字支付的核心需求集中于 “点对点安全转账”，无大规模转向理财、商户收单等衍生服务的强烈诉求，认可 “支付仅服务社交场景” 的极简设计。

- 市场竞争格局中，主流社交软件不会快速跟进 “极简核心” 定位，仍保持 “功能堆砌、广告变现” 的发展路径，留给本产品足够的市场窗口期以抢占 “极简社交” 细分领域。

2. 依赖

- 移动应用商店（Apple App Store/Google Play）的审核政策，需允许 “无广告、无衍生服务” 的极简功能形态上架，且不强制要求添加与核心愿景相悖的功能模块（如信息流、广告入口）。
- 第三方支付服务商（银行、合规支付机构）的接口与清算系统，需支持 “点对点转账、无衍生金融服务” 的极简支付模式，且保持接口稳定性（支付成功率≥99.5%）、费率合理与合规追溯能力。
- 云服务提供商（AWS/Google Cloud/Azure 等）的基础设施，需满足 “高可用（核心服务可用性≥99.9%）、低延迟（文本消息送达≤2s）、可扩展” 的技术要求，且服务等级协议（SLA）能保障核心功能（通信、存储）的持续运行。
- 合规与安全相关的第三方服务（敏感词检测、内容审核、数据加密 SDK），需支持 “轻量化集成、无冗余权限索取” 的模式，能在不增加产品复杂度的前提下满足合规要求。
- 内部基础技术组件（账号体系、安全风控中间件、运维监控平台），需按照 “极简架构、低耦合” 的原则设计，能支撑 “稳态核心→灰度扩容” 的节奏，不强制引入与核心功能无关的技术模块。
- 表情、素材等合作方提供的资源，需符合 “极简风格、无广告导流” 的要求，不通过素材植入推广信息或强制跳转。

2.4. 取舍与竞争

1. 取舍

议题	选择	放弃	业务理由
变现与体验	核心路径零广告、无推荐流	广告/商业导流位、信息流运营	与“极简、零干扰”定位一致，降低学习与操作成本
隐私与个性化	数据最小化、必要即采	个性化推荐、行为画像	降低合规与信任风险，匹配目标人群偏好
跨端一致	手机/平板/PC 体验与状态一致	端上花哨差异化功能	重度用户多端切换，高一致提升效率
实时 vs 鲁棒	端到端低延迟+弱网降质/离线队列	绝对实时（零丢零抖）	现实网络有抖动，先保证“可达与恢复”
安全合规	传输/存储加密、审计留痕、内容治理	全量默认 E2EE（影响治理）	需满足不同法域合规与审计要求
范围控制	“消息+群聊(小)+音视频+支付(点到点)”	小程序、直播、资讯、表情商店等	与“功能硬边界”一致，避免稀释核心
架构节奏	稳态核心→可观测→灰度→扩容	同期大规模生态开放	先把可靠性/可运维打牢再扩生态

2. 竞争

- 超级 App 型（QQ ）

  对手优势：生态完整、网络效应强。

  我们的差异：零广告、功能硬边界、跨端一致，降低心智负担。
  策略：一键导入通讯录与聊天记录、纯净社交卖点、弱网鲁棒与桌面端效率路径。

- 隐私优先（Signal/Telegram 等）

  对手优势：强隐私心智与加密口碑。

  我们的差异：在合规与审计前提下，提供“足够强”的隐私保护 + 轻社交与点对点支付闭环（无信息流）。

  策略：数据最小化、细粒度权限、清晰隐私承诺与可审计机制；支付仅做“好友直付”。

- 协作型工具（Slack/Teams 等）

  对手优势：团队协作深度强。

  我们的差异：面向大众的“低干扰、高直达”场景，不做工作流平台。

  策略：PC 端快捷操作/文件直传体验做深，但不扩企业后台与流程。

- 垂直小众 IM

  对手优势：某点功能极致。

  我们的差异：普适的一体化核心社交体验。

  策略：把“稳定、低时延、弱网可用”做到极致，形成质量口碑。

注：后续拓展特性的开发与上线，均需基于本章节 “取舍原则” 执行，遵循 “不新增广告、不干扰核心操作、不强制捆绑” 的底线，所有拓展功能默认隐藏，用户可自主选择启用 / 关闭，核心通信、社交、支付路径始终保持 “打开即聊、聊完即退” 的轻量化体验。

2.6. 产品特性

1. 范围内特性

分为“通信/社交/支付”三组；每条含范围、不做/边界、验收锚点。

A. 通信

1.一对一文本消息与状态

范围：发送/撤回/转发/多端已读、离线同步、失败自动重试。

不做/边界：不引入频道/公共群信息流，不做富媒体模板消息。

验收锚点：端到端送达 P50≤2s；弱网离线 100% 补投；已读跨端同步≤1s。

2.图片/文件直传（≤200MB）

范围：断点续传、图片快速预览、PC 端拖拽；病毒/敏感内容扫描。

不做/边界：超大文件分发、云盘化版本管理。

验收锚点：P95 成功率≥99.5%；典型场景首帧预览≤1s（缓存命中）。

3.一对一语音/视频通话

范围：来电/忙线状态、回呼、弱网自适应码率。

不做/边界：群组通话、直播。

验收锚点：接通 P95<3s；掉线重连≤3s；卡顿率（P95）达标。

4. 好友管理

范围：扫码/手机号添加、备注/分组、朋友圈可见范围设置。

不做/边界：附近的人、推荐联系人。

验收锚点：关键操作≤2 步；误触/误发率控制与可撤销性校验用例通过。

B. 社交

发布动态

范围：仅好友内容、发布/评论/删除/历史可见性管理。

不做/边界：推荐信息流、广告位、话题广场。

验收锚点：发布成功率≥99.5%；内容治理 + 审计留痕覆盖关键动作。

C. 支付

点对点转账 / 收款（好友/二维码）与账单查询

范围：好友转账、二维码收款、账单查询、风险拦截与限额策略；与在线网银系统安全对接。

不做/边界：理财/信贷、商户收单、聚合码生态。

验收锚点：支付处理 P95<3s；对账差异率<0.1%；授权/清算签名与证书有效。

2. 推迟特性

1. 小程序 / 第三方插件生态、开放平台与 Bot 市场：仅开放与核心场景（如文件处理、高效沟通）强关联的轻量插件，不构建开放式生态；插件需通过 “无广告、无冗余权限” 审核，默认关闭，用户可自主启用并一键卸载。
2. 直播 / 频道 / 信息流、内容推荐与话题广场：仅保留 “好友间一对一直播” 轻量功能，无公共频道、信息流及内容推荐；功能默认隐藏，不占用核心界面，避免信息干扰。
3. 群直播与超大群（>500 人）工具中心：聚焦 “高效协作补充”，仅新增群文件分类、批量权限设置等轻量工具，不开发复杂管理后台；超大群功能需用户主动申请开通，默认限制 500 人内群聊规模。
4. 表情商店 / 皮肤主题付费、内置小游戏：表情商店仅提供极简风格免费表情，付费表情采用 “单次购买永久使用” 模式（无订阅制）；皮肤仅保留 2 类基础护眼主题，不开发内置小游戏，避免功能冗余。
5. 企业组织架构与管理后台、SSO / 域集成：明确不纳入大众版拓展规划，仅作为独立 “企业精简版” 另行评估；大众版始终聚焦个人及熟人社交，不偏离核心定位。
6. 强默认 E2EE 全量覆盖：延续 “V1 合规优先” 策略，后续仅新增 “私聊可选 E2EE 加密” 功能，群聊保持合规审计所需机制，兼顾隐私需求与多法域合规要求。
7. 高阶 AI 功能（语音转文字多语实时字幕、智能摘要 / 检索）：仅开发服务于核心通信的轻量 AI 工具，需用户主动触发（如手动开启语音转文字），无自动推荐、行为画像功能，不增加操作复杂度。
8. 国际化多语言与多法域专项支持：待国内市场稳定、核心功能验证成熟后逐步推进；多语言适配不新增冗余界面，保持与国内版一致的极简操作逻辑，同步遵循目标法域合规要求。

所有后续拓展特性的启用，需满足以下前提：

核心功能稳定性达标（核心服务可用性≥99.9%，用户满意度≥90%）；

目标用户覆盖达成（极简需求市场份额≥15%）；

经用户调研验证，拓展功能需求与核心体验强相关（需求匹配度≥80%）；

不新增广告推送、不强制授权非必要权限、不改变核心操作路径。

2.7. 其他产品需求

1. 可应用标准

产品必须符合以下所列的各类标准与法规：

①法律和规章：遵守欧盟《通用数据保护条例》（GDPR）中关于用户数据处理、知情同意和“被遗忘权”的规定。遵守中国《网络安全法》和《个人信息保护法》中关于数据本地化（如适用）和用户信息保护的要求。

②通信标准：使用标准的 TCP/IP 协议簇进行网络通信。即时消息传递采用基于 XMPP 或类似开放标准的私有协议，确保可扩展性和互操作性潜力。

③平台兼容标准：移动应用需遵循 Apple iOS《App Store 审核指南》和 Google Android《用户数据政策》。

④质量与安全标准：数据传输全程使用 TLS 1.2 及以上版本进行加密。代码开发过程中需遵循 OWASP 移动应用安全十大关键风险的建议。静态代码扫描需通过预定义的安全质量门禁。

2. 系统需求

定义为支持 WeChat App 应用所必需的系统环境：

支持的操作系统：Apple iOS 14.0 及以上版本。Google Android 8.0 (APILevel 26) 及以上版本。

硬件配置（最低要求）：内存至少 2GB RAM。

存储空间：安装后至少需要 100MB 可用空间。

网络：支持 Wi-Fi 或移动数据连接。

摄像头与麦克风：用于媒体共享和未来的音视频通话功能。

伴生软件与外部依赖：依赖 Apple Push Notification Service (APNs) Firebase Cloud Messaging (FCM) 用于消息推送。后端服务部署于主流云服务平台（如 AWS、Google Cloud 或 Azure）。

3. 性能需求

产品需在以下性能指标上满足要求，以确保良好的用户体验：

①响应时间：在正常网络条件下，应用冷启动时间不超过 3 秒。文本消息的端到端送达延迟中位数应小于 2 秒。

②吞吐量与并发：系统后端应支持单个发布版本内至少 10,000 名用户同时在线。支持每秒处理 1,000 条消息的峰值负载。

③可靠性：核心消息服务月度正常运行时间不低于 99.9%。在客户端与服务器网络连接临时中断恢复后，应能自动重新同步消息，且无丢失。

④资源利用：在典型使用情况下，应用前台运行时的内存占用不应超过 150MB。应用在后台静默运

行时的电量消耗不应显著影响设备正常续航。

2.8. 术语表

1. 参与者与角色

术语	说明	补充信息
网民（用户）	使用系统进行通信/轻社交/支付的最终用户	与“用户（User）”同义
主叫网民	发起一对一会话邀请的网民（瞬时角色）	仅在该会话建立流程内有效
目标网民	接收邀请的一方（瞬时角色）	又称“被叫网民/被叫用户”
聊天运营管理员	监控可用性、处理举报、查询留痕	不直接从事资金清算
安全与合规管理员	内容与交易的合规治理、冻结/解冻、报送	负责合规审计与稽核证据
数字支付运营管理员	支付通道健康、对账/差错处理、限额策略	与银行/三方支付对接
在线网银系统	授权、清算与对账清单的外部系统	受信安全通道对接（签名/证书/双向 TLS）
管理团队	对业务结果与里程碑负责的内部管理层	参与取舍与版本决策
项目/开发团队	研发/测试/架构/运维等实现团队	负责交付质量与可运维性

2. 会话与通信

术语	说明	补充信息
会话	双方之间的持续通信上下文	支持文本/语音/视频模式
会话邀请	主叫方向目标方发起的会话请求	包含模式/可用性协商
会话信道	建立后用于传输数据的逻辑通道	需支持状态同步与重连
已读同步	多端对消息阅读状态的一致性	目标：跨端≤1 秒达成一致
跨端一致	手机/平板/PC 三端行为和状态一致	含 UI 关键路径与快捷操作一致
弱网降质	带宽/时延不佳时自动降低质量	自适应码率/压缩/首帧优先
离线消息	离线期间累积的待投递消息	上线后按序补投并去重
断线重连	连接异常后的自动恢复	与幂等/去重策略协同

呼叫状态	来电、忙线、回呼等状态	用于音视频拨测与回呼策略

3. 消息与传输

术语	说明	补充信息
送达状态	消息生命周期的状态标识	发送中/已送达/失败
幂等性	重复请求仅产生一次有效结果	依赖消息 ID 与去重策略
去重	避免同一消息被多次处理	与重试/队列协同
重试窗口	自动重投的时间与次数限制	与幂等与去重配合
断点续传	中断后从已完成位置继续传输	适用于图片/文件直传
首帧预览	图片/视频首帧的快速展示	提升弱网与大图场景体验
消息队列	用于削峰与可靠投递的中间件	保障顺序/重试/回退
超时控制	超过时限自动判失败并处理	触发重试/降级/回滚

4. 支付与金融

术语	说明	补充信息
点对点转账	好友之间直接转账	含二维码收款
授权/清算	银行侧确认与资金结算流程	与在线网银系统对接
对账	交易流水与银行清单核对	关注差异识别与补偿
差错处理	对账差异或异常交易的修正流程	需留痕与可追溯
限额策略	基于风险的金额/频次限制	按用户/场景配置
交易留痕	交易相关的审计记录	用于稽核与合规举证
风险拦截	对可疑交易的规则/模型拦截	与限额/黑白名单联动

5. 安全与合规

术语	说明	补充信息
数据最小化	仅收集完成目的所必需的数据	与零干扰定位一致
内容治理	对违规内容的检测、处置与留痕	支持举报闭环与审计
审计日志	关键操作的可追溯记录	保留期≥既定天数（如 180 天）
合规报送	向监管/平台的周期性报告	满足法域时限与格式

TLS	传输层加密协议（⩾1.2）	建议双向认证
E2EE	端到端加密	与合规/治理要求需权衡

6. 平台与生态

术语	说明	补充信息
APNs/FCM	iOS/Android 推送服务	依赖其可用性与策略
SDK	第三方功能包	需合规评审与最小权限
应用商店审核	App 上架与更新的审核	遵循商店政策与数据条款
云 SLA	云服务商可用性承诺	直接影响整体 SLA

7. 产品范围与版本

术语	说明	补充信息
功能硬边界	只做核心，拒绝冗余功能的边界	保持零干扰与低学习成本
范围内特性（V1.0）	s 本版本承诺交付的能力集合	通信/社交/支付/平台基础
推迟特性	明确延后的能力集合	条件成熟后再纳入
纯净朋友圈	无广告/推荐的信息发布与互动	仅好友可见范围控制
文件直传	⩽既定尺寸的点对点文件传输	支持断点续传与查毒

第3章 用况建模

3.1. 术语表

名称	定义	补充信息
网民	使用本系统进行通信与支付的最终用户。	与“主叫网民/目标网民”为同一参与者类型在会话中的不同角色；
主叫网民	在通信用例中发起会话邀请的网民（会话内的瞬时角色）。	仅在会话期间有效；与“网民”同类型；常用字段：caller_id、邀请时间、会话 ID。
目标网民	被主叫选定并接收邀请的一方。	与主叫同为“网民”类型的对等体；常用字段：callee_id、可达/忙闲状态。
聊天运营管理员	负责通信可用性与质量监控、处理用户举报、查看会话留痕。	不处理支付账务与通道对账；需要处置与查询权限。
安全与合规管理员	监控敏感内容与可疑交易，执行合规审核、冻结/解冻、报送。	生成/审阅合规报告；不负责运营 SLA 与日常运行指标。
数字支付运营管理员	负责支付通道健康、限额/策略配置、对账与补偿处置。	关注差异交易处理与运营报表；不处理聊天内容或举报。
在线网银系统	银行侧授权/清算与对账清单提供者，对接支付请求并返回结果。	外部系统，通过受信安全通道（证书/签名）连接；
朋友圈管理员	负责朋友圈内容合规审核、举报处理、历史动态管理的运营角色	与聊天运营管理员协作，专注于朋友圈场景的运营与合规

3.2. Wechat 用况的描述

1. 简要描述

谁来使用本用况模型	网民（平台用户）、安全与合规管理员、聊天运营管理员、数字支付运营管理员、朋友圈管理员、在线网银系统（外部）与（可选）商户系统
人们要求系统做什么	即时沟通（文本/语音/视频）与安全、可追溯的数字支付；在异常情况下能够审计、合规处置与账务纠偏
到底是什么	一个为用户提供安全便捷的社交沟通与支付结算服务的综合性数字平台，构建沟通与交易的闭环体验
本系统的目的	让用户在可信条件下完成一对一通信与单笔支付；同时让管理员能进行风控、合规与运营维护
本系统并不会	暴露底层 UI 细节或硬件驱动流程；替代银行清结算职能；绕过监管要求
我们的产品	7×24 可用；接口与外部网银系统安全互联；所有关键操作可审计、可追溯

2. 参与者类别

图 1 给出了 WeChat 用况模型中的所有参与者，下面小节给出了这些参与者的简略描述。

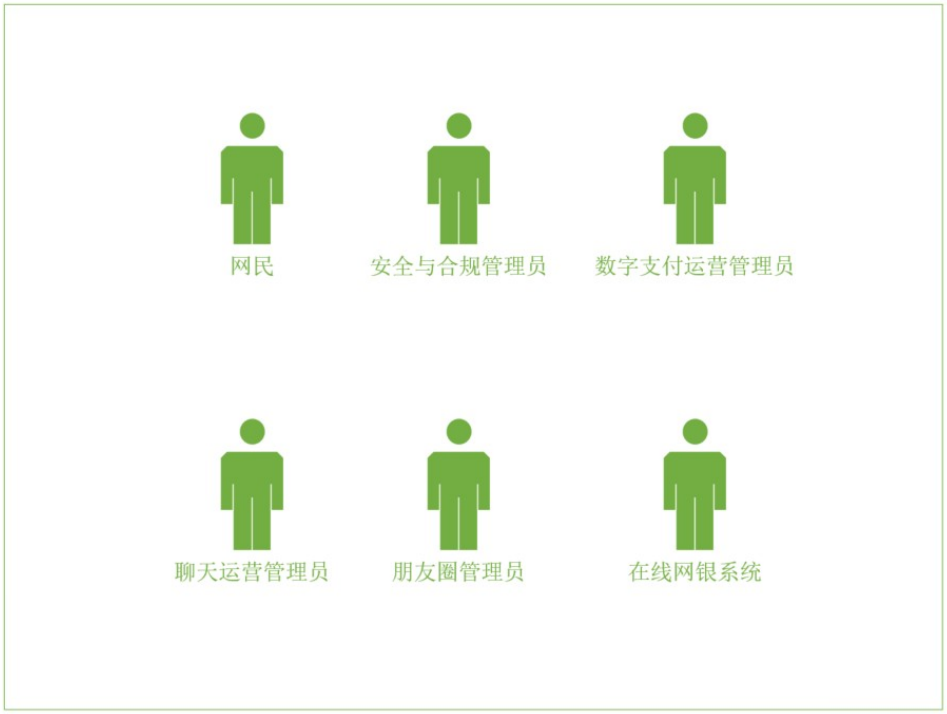


图 1　　WeChat 中的参与者

1.网民（Customer/User）

在客户端发起与接收通信；在支付场景中提交支付请求、接收结果与回执。主叫/被叫均视为此参与者。

2.安全与合规管理员（Security & Compliance Admin）

关注敏感内容、违规与可疑交易；查看告警、处置冻结/解冻、生成合规报告。

3.数字支付运营管理员（Payment Ops Admin）

关注支付通道健康、对账差异与补偿；配置限额与风控策略；查看运营报表。

4.聊天运营管理员（Chat Ops Admin）

监控通信可用性与质量告警；查询会话留痕；处理用户举报并回访。

5.朋友圈管理员（Moments Admin ）

负责朋友圈内容合规审核、举报处理、历史动态管理的运营角色

6.在线网银系统（Bank System）

为本系统提供扣款/转账授权、返回同步/异步结果、输出清单用于对账。

3.3. WeChat 的主要用况

图 2 给出了 WeChat 用况模型中的主要用况。下面小节则给出这些用况的简略描述。

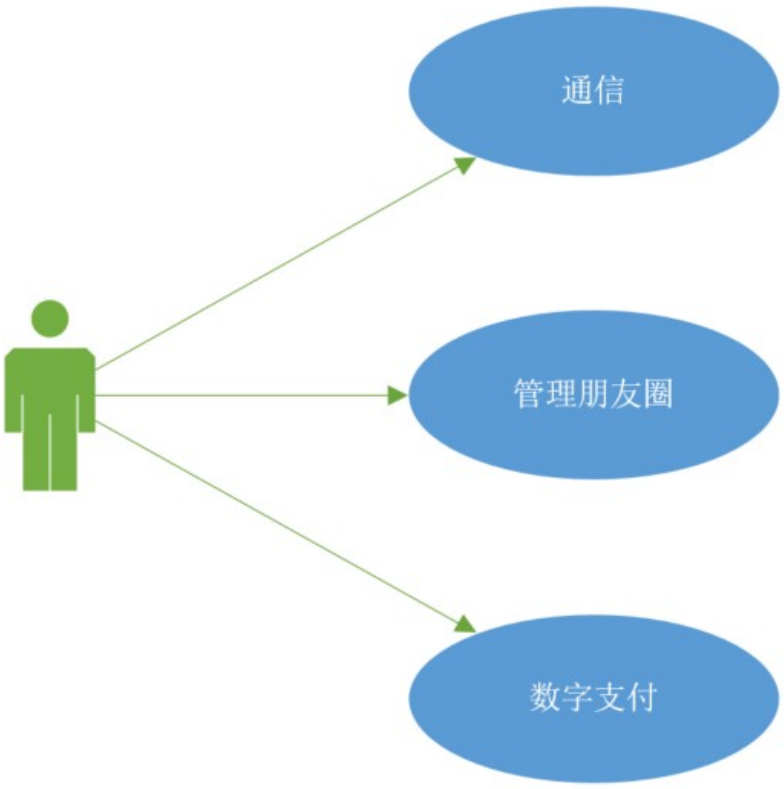


图 2　WeChat 主要用况

数字支付

网民对单笔订单完成支付；系统执行风控与授权、结果入账、异步对账/补偿。

通信

网民与网民之间建立一对一会话（文本/语音/视频），包含好友添加、备注分组、权限设置等好友管理操作，支持敏感词过滤、举报与会话留痕。

管理朋友圈

网民发布、互动、管理纯好友动态，实现无广告、高隐私的内容分享。

3.4. 补充用况

合规处置

对通信、朋友圈、支付场景的违规内容 / 交易进行记录、上报与冻结 / 解冻（由安全与合规管理员、聊天运营管理员、朋友圈管理员驱动）。

运营监控与对账

监控通信、支付、朋友圈的服务质量，生成对账与运营报告（由聊天运营管理员、数字支付运营管理员、朋友圈管理员驱动）。

3.5. 用况描述——通信(UC-1)

1. 简要描述

该用况描述了网民如何通过系统与另一网民建立一对一实时通信，支持文本消息、语音通话和视频通话三种通信模式，并确保通信过程的安全性和合规性。

2. 用况图

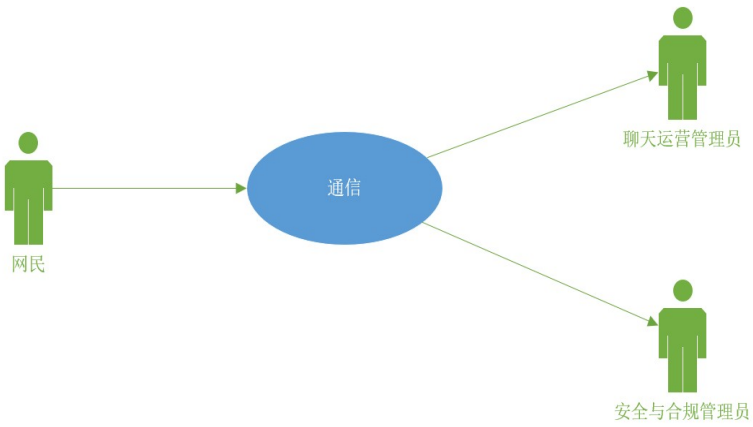


图 3　通信用况的用况图

3. 前置条件

- 通信双方网民均已成功登录系统。
- 通信双方网民互为好友或存在于可通信列表中。为确保此条件满足，用户可能需先执行 {添加好友} 子流。
- 系统网络连接正常且可用。
- 网民设备的相关硬件（麦克风、摄像头等）可用且已授权。

4. 基本流

{好友管理}

1.若发起方与目标方非好友关系：执行{添加好友}子流，添加成功后进入{建立连接}阶段；若添加失败，用况终止或重新发起添加。

2.若发起方与目标方已为好友关系：进入{建立连接}阶段。

{建立连接}

1. 用况开始时，参与者网民从联系人列表中选择目标网民。

2. 参与者网民选择通信模式（文本、语音或视频）。

3. 若选择文本模式：系统直接建立文本通信信道，通知双方 “文本连接成功”，进入{进行通信}子流。

4. 若选择语音或视频模式：系统向目标网民发送带对应模式的会话邀请,系统等待目标网民响应，最多 30 秒。

若目标网民接受邀请，系统建立对应（语音 / 视频）通信信道并通知双方 “语音 / 视频连接成功”，进入{进行通信} 子流。

若目标网民拒绝邀请或超时未响应，执行对应的备选流，用况终止。

{进行通信}

5. 根据通信模式执行相应子流：

a. 如果通信模式为文本：则执行{处理文本消息交换}

b. 如果通信模式为语音：则执行{维护语音通信}

c. 如果通信模式为视频：则执行{维护视频通信}

{结束通信}

6. 当任一网民选择结束通信时，系统执行子流 {结束通信会话}。

7. 用况终止。

5. 子流

5.1. 处理文本消息交换

1. 系统建立文本消息传输通道。

2. 当参与者网民发送文本消息时：

a. 系统接收消息并执行实时敏感词过滤。

b. 如果消息包含敏感词，系统用"***"替换敏感词并记录审核日志。

c. 系统将处理后的消息实时投递给目标网民。

d. 系统在双方界面显示消息。

3. 持续执行直到收到结束通信信号。

4. Resume at the next step.

5.2. 维护语音通信

1. 系统建立高质量语音数据通道；

2. 系统开始捕获、编码、传输双方的语音数据；

3. 系统实时监控网络质量，动态调整音频参数；

4. 系统处理回声消除和背景降噪；

5. 如果检测到网络质量下降，系统自动调整码率优先保证通话连贯性；

6. 持续执行直到通信结束；

7. 返回到调用点；

5.3. 维护视频通信

1. 系统建立结合视频和语音的数据通道；

2. 系统开始捕获、编码、传输双方的视频和语音数据；

3. 系统实时监控网络和性能，动态调整视频分辨率；

4. 系统支持前后摄像头切换功能；

5. 如果检测到网络质量下降，系统优先保证语音质量；

6. 持续执行直到通信结束；

7. 返回到调用点；

5.4. 结束通信会话

1. 系统断开通信信道；

2. 系统记录会话日志，包括：

   a. 开始时间和结束时间；

   b. 通信参与者；

   c. 通信模式；

   d. 会话时长；

3. 系统生成合规审核记录；

4. 系统释放相关资源；

5. 返回到调用点；

5.5. 添加好友

1.参与者网民通过搜索 ID、昵称或从系统推荐列表中找到目标网民。

2.参与者网民向目标网民发送好友邀请。

3.系统向目标网民推送好友邀请通知，并等待其响应（无超时限制）。

4.若目标网民接受邀请：

   a. 系统将双方添加到彼此的好友列表中。
   b. 系统通知参与者网民“好友添加成功”。
   c. 子流成功结束。

5.若目标网民拒绝邀请：

   a. 系统通知参与者网民“对方已拒绝您的好友邀请”。
   b. 子流终止。

6.返回到调用点。

6. 备选流

6.1. 处理邀请被拒绝

At {建立连接} if 目标网民拒绝邀请

1.系统通知参与者网民"对方已拒绝您的通信邀请";

2.用况终止;

6.2. 处理邀请超时

At {建立连接} if 目标网民在 30 秒内未响应邀请

1.系统通知参与者网民"对方无应答";

2.用况终止;

6.3. 处理目标网民正忙

At {建立连接} if 检测到目标网民正在其他通信会话中

1.系统通知参与者网民"对方正忙，请稍后再试";

2.用况终止;

6.4. 处理网络质量下降

At {进行通信} if 系统检测到网络质量严重下降

1.系统自动降低音频码率以保证通话连贯性;

2.系统自动降低视频分辨率或建议网民停止视频;

3.系统在界面显示"网络连接质量不佳"提示;

4.通信流程从检测到问题的地方继续;

6.5. 处理通信中断

At {进行通信} if 通信连接意外中断

1.系统自动尝试重新建立连接;

2.如果 30 秒内无法重新连接，系统记录通信中断事件;

3.系统通知双方网民"通信连接已中断";

4.执行子流 {结束通信会话};

5.用况终止;

6.6. 处理内容违规

At {进行通信} if 系统检测到多次敏感内容

1.系统记录违规事件并生成合规报告;

2.系统向安全与合规管理员发送通知;

3.系统可临时限制该网民的通信功能;

4.通信流程从检测到违规的地方继续;

6.7. 处理网民举报

At {进行通信} if 任一网民举报对方

1.系统记录举报信息并保存通信记录；

2.系统向聊天运营管理员发送举报通知；

3.通信流程从举报发生的地方继续；

6.8. 处理网民无响应

At {进行通信} if 网民在 60 秒内无操作

1.系统发出提示音提醒网民；

2.如果继续无响应，系统显示"会话即将超时结束"警告；

3.如果 120 秒内仍无响应，执行子流 {结束通信会话}；

4.用况终止；

6.9. 处理添加好友失败

At {添加好友} if 搜索用户不存在或因隐私设置无法被查找

1.系统通知参与者网民“未找到相关用户或该用户不允许被添加”。

2.子流终止。

7. 后置条件

- 通信会话已被完整记录，包括开始时间、结束时间、通信模式和参与者信息；
- 必要的安全与合规审核记录已生成（如涉及敏感内容或用户举报）；
- 系统资源已被正确释放；
- 用户通信状态已更新；

8. 特殊需求

1. 性能要求

a. 文本消息投递延迟：小于 1 秒；

b. 语音通话延迟：低于 400 毫秒；

c. 视频通话延迟：低于 500 毫秒；

d. 系统可用性：99.5%以上；

2. 安全要求:

a. 所有通信内容必须进行实时敏感词过滤；

b. 语音和视频通信必须采用加密传输；

c. 用户隐私数据必须得到保护；

3. 合规要求:

a. 通信记录必须保存至少 90 天；

b. 敏感内容触发必须实时记录并报告；

c. 用户举报必须及时处理并记录；

3.6. 用况描述——数字支付(UC-2)

1. 简要描述

网民通过系统完成一笔单笔订单的数字支付；系统完成请求接收、风控与合规校验、向在线网银系统发起授权、结果入账与通知，并保证可追溯与对账一致性。

2. 用况图

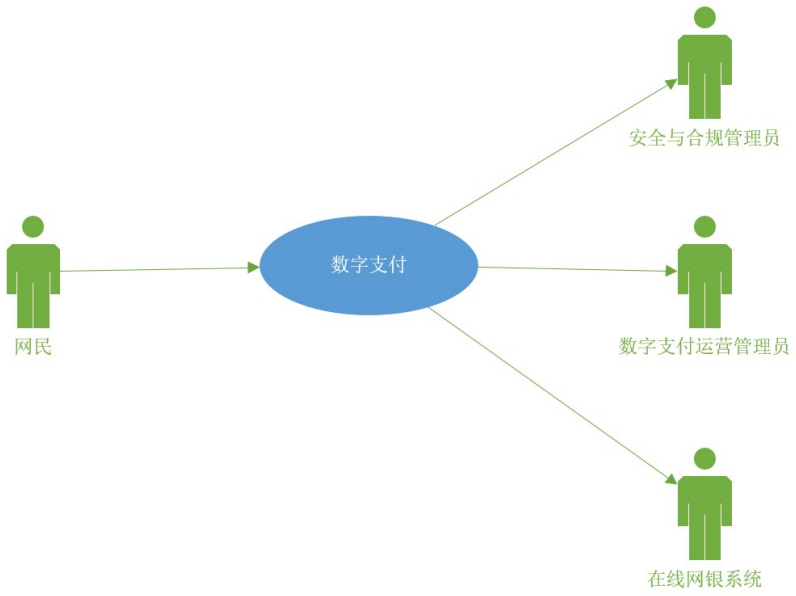


图 4 数字支付用况的用况图

3. 前置条件

- 网民已登录且支付账户处于可用状态（通过必要的 KYC/实名校验）；
- 目标订单存在、未支付、未过期，金额与收款方信息完整有效；
- 支付方式（银行卡/账户余额等）已绑定且允许本次交易限额；
- 系统与在线网银系统网络连通、签名/证书有效。

4. 基本流

{接收支付请求}

1.系统接收网民提交的支付请求（订单号、金额、收款标识、支付方式等）。

2.系统创建支付交易并置状态为“待授权”，记录审计上下文（设备、IP、时间）。

{校验与风控}

3.系统执行子流{风控与合规校验}。若通过，继续；否则转入备选流 {风控拒绝}/{合规疑似}。

{发起银行授权}

4.系统根据支付方式选择路由，调用子流 {银行授权处理}，向在线网银系统发起扣款/转账授权请求。

5.系统等待授权结果，最长 60 秒。

{结果处理}

6.系统接收授权结果：

a. 如果返回"成功"：系统执行 {结果入账与通知}

b. 如果返回"处理中"：系统标记交易为"待确认"，执行 {异常补偿与对账}

7.系统向网民展示支付结果。

8.用况终止。

5. 子流

5.1. 风控与合规校验

1. 系统验证订单信息的完整性和有效性，包括订单号、金额、收款方标识。
2. 系统检查交易金额是否在用户单日及单笔限额范围内。
3. 系统验证交易频率是否符合安全规则，防止高频异常交易。
4. 系统检查设备指纹、地理位置信息和 IP 地址是否存在异常模式。
5. 系统查询用户账户是否在黑名单或高风险名单中。
6. 系统执行反洗钱规则检查，识别可疑交易模式。
7. 如果发现中等风险行为，系统要求用户进行额外身份验证。
8. 如果发现高风险行为，系统触发人工复核流程并生成合规工单。
9. 系统记录所有风控检查的审计轨迹和决策依据。
10. Resume at the next step.

5.2. 银行授权处理

1. 系统根据支付方式选择对应的银行通道和接口协议。
2. 系统组装银行授权请求报文，包括交易金额、商户信息、用户标识。
3. 系统对请求报文进行数字签名和加密处理。
4. 系统与在线网银系统建立安全加密通道。
5. 系统发送授权请求到银行系统并启动响应计时器。
6. 系统接收银行系统的同步响应或受理回执。
7. 如果银行要求额外认证，系统转发挑战需求并收集用户认证结果。
8. 系统解析银行响应，提取授权码、交易状态和错误信息。
9. 系统记录完整的银行交互报文摘要和时序信息。
10. 系统更新交易状态为"已授权"或根据响应设置相应状态。
11. Resume at the next step.

5.3. 结果入账与通知

1. 系统依据银行授权结果更新交易状态：
2. 如果授权成功：更新为"支付成功"
3. 如果授权失败：更新为"支付失败"并记录失败原因
4. 系统执行原子记账操作，确保资金账户余额准确更新。
5. 系统生成唯一的电子回执号和交易凭证。
6. 系统记录完整的账务分录，包括借方、贷方账户和金额。
7. 系统向参与者网民发送支付结果通知，包括成功/失败状态和交易详情。
8. 系统向商户系统发送支付成功通知。
9. 如果交易涉及风险或合规事件，系统向安全与合规管理员发送告警通知，安全与合规管理员收到告警后，可根据事件类型决定是否介入人工处置，处置结果同步更新交易状态。
10. 系统持久化所有交易日志、审计信息和操作痕迹。
11. 系统更新相关业务系统的订单状态为"已支付"。
12. Resume at the next step.

5.4. 异常补偿与对账

1. 系统生成对账任务，设置对账时间窗口和重试策略。
2. 系统从在线网银系统拉取指定时间段的交易清单。
3. 系统将本地交易记录与银行交易清单进行逐笔比对：
4. 识别"银行成功而本地未入账"的交易
5. 识别"银行失败而本地误入账"的交易
6. 识别金额不一致的交易记录
7. 对于比对发现的差异交易，系统执行相应的补偿操作：
8. 如果是"银行成功而本地未入账"：执行补记账操作并补发通知
9. 如果是"银行失败而本地误入账"：执行冲正操作并通知相关方
10. 如果是金额不一致：暂停交易并生成人工处理工单
11. 系统更新交易状态为最终确定状态。
12. 系统生成对账报告，包括总交易数、成功数、差异数和处理结果。
13. 系统将对账结果通知数字支付运营管理员。
14. 系统归档完整的对账记录和补偿操作日志。
15. Resume at the next step.

6. 备选流

6.1. 风控拒绝

At {校验与风控} if 命中高风险/黑名单/限额：

1.系统将交易置“拒绝”，保存拒绝原因与证据；

2.通知网民“风险拦截，交易未受理”；

3.发送告警给安全与合规管理员；

4.用况终止。

6.2. 余额不足 / 账户受限

At {发起银行授权} if 网银返回资金不足或账户限制：

1.记录失败码并将交易置“失败”；

2.通知网民失败原因；

3.用况终止。

6.3. 银行系统超时

At {发起银行授权} if 60 秒内未获结果：

1.系统按退避策略重试至上限；

2.若仍超时，置交易为“待确认”，记录对账待跟踪条目；

3.进入 {异常补偿与对账}；

4.返回{结果处理}继续。

6.4. 二要素/3-DS 挑战

At{发起银行授权} if 网银要求额外认证：

1.系统转发挑战需求并收集认证结果；

2.若认证通过，返回{发起银行授权}继续授权；

3.若认证失败或超时，置“失败”，用况终止。

6.5. 用户取消

At {接收支付请求} if 网民在授权前撤销：

1.系统将交易置“已取消”并解除幂等锁；

2.用况终止。

6.6. 重复支付防重

At {接收支付请求} if 检测到相同订单号与幂等键已成功：

1.系统直接返回已成功结果与原回执；

2.用况终止。

6.7. 合规疑似（需人工复核）

At {校验与风控} if 命中可疑模式（反洗钱/套现）：

1.置交易为“冻结待核查”，生成合规工单；

2.通知安全与合规管理员；

3.用况暂止，待人工处置，人工处置需在 48 小时内完成，处置结果通过系统工单流转，若结论为‘合规通过’则执行 {结果入账与通知}；若结论为‘违规’则置交易为‘失败’并通知网民

6.8. 回调丢失/账务不一致

At {结果处理} if 异步对账发现银行已成功而本地未落账：

1.触发 {异常补偿与对账}进行补记账与通知；

2.记录“补偿成功/失败”；

3.返回{结果处理}结束。

6.9. 网银系统不可用

At{发起银行授权}if 通道故障：

1.切换备用通道或置交易“系统繁忙”；

2.通知运营管理员；

3.用况终止或进入 {银行系统超时} 流程。

6.10. 网民长时间无操作

At {接收支付请求} if 15 分钟未完成授权：

1.系统关闭交易为“超时关闭”；

2.用况终止。

7. 后置条件

- 交易状态为“成功”或“待确认/冻结/关闭”等终结性或可追踪状态；
- 账务分录、回执、审计日志与对账条目完整可追溯；
- 若触发合规事件，相关报告/工单已生成并通知到位；
- 与在线网银系统的交互记录可核验。

8. 特殊需求

1. 性能要求：

   a. 支付交易处理时间：小于 3 秒；

   b. 系统并发处理能力：支持每秒 1000 笔交易；

   c. 系统可用性：99.9%以上；

   d. 支付结果通知延迟：小于 1 秒；

2. 安全要求：

   a. 所有支付数据传输必须采用加密传输；

   b. 支付密码输入必须使用安全键盘；

   c. 大额交易必须进行多重身份验证；

d. 用户支付信息必须安全存储；

3. 合规要求：

a. 支付交易记录必须保存至少 5 年；

b. 单笔交易超过一定金额必须报告监管部门；

c. 可疑交易必须实时记录并报告；

d. 用户隐私数据必须得到保护；

4. 审计要求：

a. 所有支付操作必须生成审计日志；

b. 系统管理员操作必须记录操作痕迹；

c. 支付状态变更必须记录变更历史；

3.7. 用况描述——管理朋友圈(UC-3)

1. 简要描述

网民通过系统发布、互动及管理个人朋友圈动态，支持纯好友社交场景下的内容分享与隐私控制，无广告、无推荐流，确保体验纯粹且符合合规要求。

2. 用况图

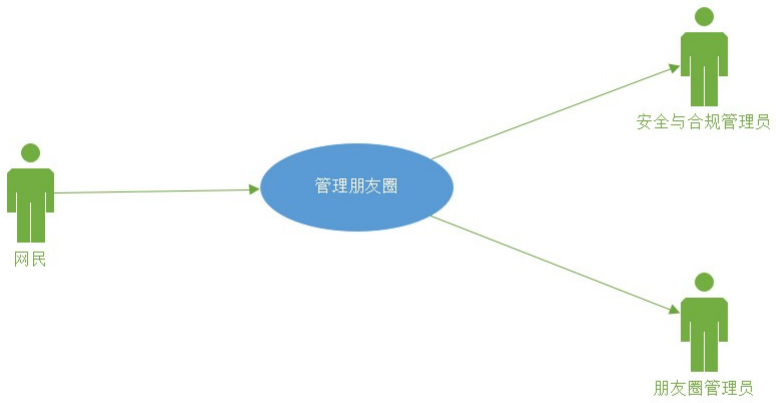


图 5 管理朋友圈用况的用况图

3. 前置条件

- 网民已登录系统，且朋友圈功能未被限制（如因违规被临时禁言）；
- 网民已知悉并同意遵守平台内容规范；
- 系统仅获取发布动态必需的权限（如相机、相册权限，支持用户手动关闭）。

4. 基本流

{发布动态}

1. 网民选择“发布朋友圈”功能，可编辑文本、上传图片（≤9 张）或短视频（≤15 秒）。
2. 网民设置动态可见范围（公开 / 部分好友可见 / 不给谁看 / 仅自己可见）。
3. 网民确认发布，系统接收动态内容并执行子流{合规检测与处理}。
4. 系统保存动态并同步至符合可见范围的好友朋友圈列表。
5. 系统向发布者反馈“发布成功”，动态按发布时间倒序展示。

{互动动态}

6.好友查看动态后可执行点赞或评论操作：

a. 点赞：系统实时同步点赞状态至发布者与好友端；

b. 评论：好友提交评论。系统接收评论内容，执行子流{合规检测与处理}，随后将处理后的评论展示给发布者及所有有权限查看该动态的好友；

7. 发布者可查看互动通知，对好友评论进行回复（仅评论双方及动态可见者可见）。

{管理动态}

8. 发布者可对自有动态执行操作：

a. 编辑：发布后 1 小时内可修改文本、可见范围（不可修改图片 / 视频）；

b. 删除：随时删除整条动态，系统同步移除所有好友端展示；

c. 管理互动：可关闭动态的点赞 / 评论功能，实时生效。

9. 用况终止。

5. 子流

5.1. 合规检测与处理

1.系统对发布内容、评论内容执行敏感词与违规信息检测。

2.若含轻微违规内容，用 “***” 替换敏感词并记录审核日志。

3.若含严重违规内容，拒绝发布并提示违规原因，生成合规报告。

4.返回检测结果至主流程。

5.2. 可见范围配置

1.系统提供分组选择、好友搜索等快捷方式，支持精准筛选可见 / 不可见好友。

2.系统保存配置方案，后续发布可直接复用历史可见范围设置。

3.Resume at the next step.

6. 备选流

6.1. 内容发布失败

At {发布动态} if 网络中断或系统异常：

1.系统自动保存动态草稿（本地保留 24 小时）；

2.提示网民“发布失败，请检查网络后重试”；

3.提示网民“发布失败，已保存为草稿，请检查网络后重试”；

4.用况终止。

6.2. 动态含严重违规内容

At {发布动态} if 合规检测未通过：

1.系统拒绝发布并显示违规提示；

2.记录违规事件并通知安全与合规管理员；

3.累计 3 次违规临时限制朋友圈功能；

4.用况终止。

6.3. 好友无查看权限

At {互动动态} if 好友不在可见范围：

1.系统显示“该内容无法查看”，不泄露权限设置细节；

2.互动功能对该好友隐藏；

3.流程继续。

6.4. 误操作删除动态

At {管理动态} if 网民误删动态：

1.系统支持“最近删除”列表（保留 7 天）；

2.网民可选择恢复动态，恢复后权限与互动记录同步恢复；

3.流程返回 {管理动态} 继续。

7. 后置条件

- 动态及互动记录（点赞 / 评论 / 回复）已完整保存，多端同步一致；
- 可见范围配置持续生效，权限变更后好友端即时更新；
- 违规内容已按规则处置，合规报告与审计日志生成完毕；
- 系统释放临时占用的存储与网络资源。

8. 特殊需求

1. 性能要求：

a. 动态发布响应时间≤2 秒；

b. 互动操作同步延迟≤1 秒；

c. 动态加载首屏时间≤1.5 秒；

2. 安全要求：

a. 动态内容传输采用加密方式

b. 隐私权限设置严格生效，无越权访问风险；

c. 本地草稿仅存储在用户设备，不自动上传云端；

3. 合规要求：

a. 动态与互动记录保存至少 90 天；

b. 违规内容检测与处置记录可追溯；

c. 不收集动态内容用于个性化推荐或商业用途。

第4章需求分析

4.1. 健壮性分析

1. 健壮性分析概述

健壮性分析（Robustness Analysis）最初源自 Jacobson 的 Objectory 方法，是一种基于用况的、轻量级的对象协作分析技术，用于在需求分析阶段“粗粒度地”确定系统内部需要哪些类及其大致职责。它强调在不陷入实现细节的前提下，通过用况事件流来发现：

1.用户与系统之间的交互界面（边界类，boundary）；

2.需要长期保存的业务信息（实体类，entity）；

3.协调边界与实体、承载用况业务流程逻辑的控制逻辑（控制类，control）。

在 WeChat 项目中，我们主要针对以下三个核心用况进行健壮性分析：

1.通信（UC-1）：一对一文本/语音/视频通信，用况描述详见第 3 章 3.5 小节；

2.数字支付（UC-2）：网民完成单笔订单的支付及异常补偿，用况描述详见 3.6 小节；

3.管理朋友圈（UC-3）：朋友圈动态发布、互动与管理，用况描述详见 3.7 小节。

在健壮性分析中需要遵守以下建模规则：

1.边界类（boundary）与实体类（entity）可以与控制类（control）通信；

2.实体类之间不直接通信（名词不能彼此对话）；

3.控制类之间可以互相通信（动词之间可以协调合作）；

4.尽量保持控制类的职责内聚，每个用况的主要业务流程由少量控制类主导。

2. 通信用况（UC-1）的健壮性分析

本节基于 UC-1 通信用况的基本流与备选流，对系统在该用况中的核心对象、对象衍型（Boundary / Control / Entity）及对象间协作关系进行分析，并结合协作图、通信图及分析类图给出系统在执行通信用况时的结构化行为视图。

2.1. 用况简述

UC-1 通信用况描述：网民从联系人列表中选择目标网民，选择文本 / 语音 / 视频模式，系统建立对应通信信道，期间执行文本消息传输、敏感词过滤、语音 / 视频媒体维护、网络质量监控及合规记录，并在任一方结束通信后完成会话日志与合规审核记录的生成。该用况包含多个子流与备选流，包括：处理文本消息交换、维护语音通信、维护视频通信、结束通信会话、添加好友，以及邀请被拒绝、网络质量下降、通信中断、内容违规、网民举报、无响应等多种异常场景。

本节的健壮性分析仅聚焦 UC-1 所需的内部分析类及其协作关系，为后续的交互建模和类图设计提供结构化视图。

2.2. 通信用况协作图

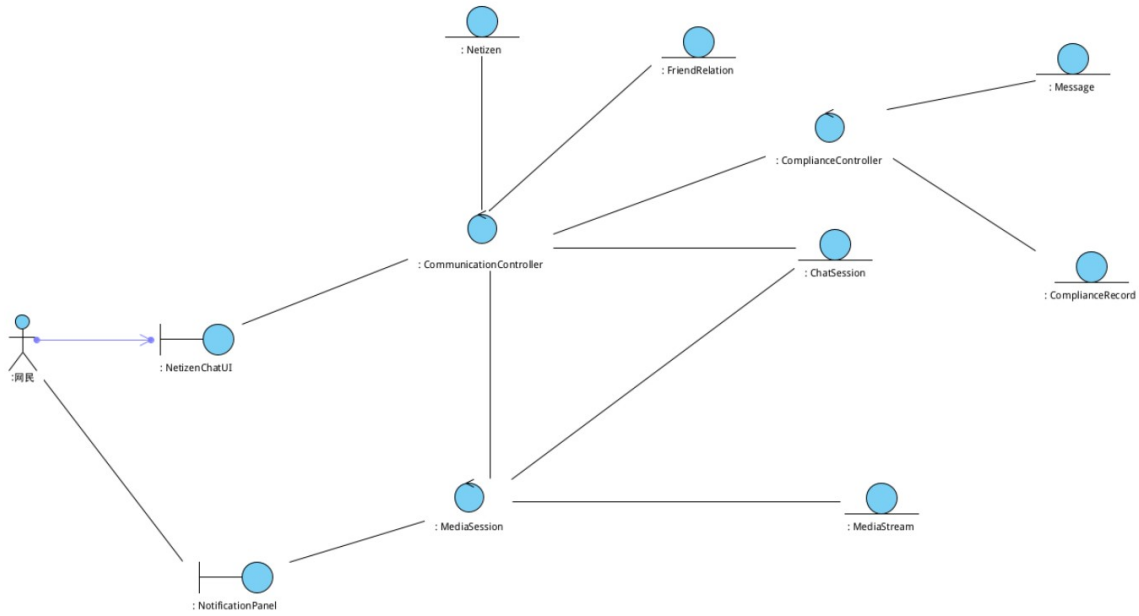


图 6 通信用况协作图

2.3. 通信用况通信图

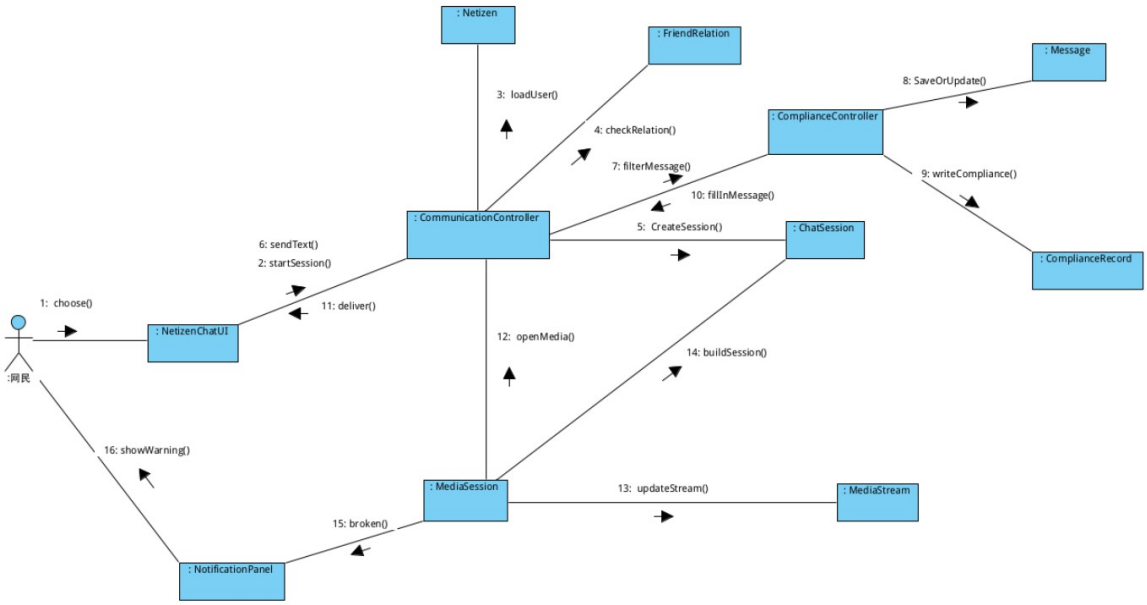


图 7 通信用况通信图

通信图的执行过程如下：

选择好友与通信模式（1）

网民在 NetizenChatUI 中选择目标好友并指定文本/语音/视频模式，触发通信流程。

发起建立会话请求（2）

UI 将 startSession 请求发送给核心控制器 CommunicationController。

用户信息与好友关系校验（3–5）

控制器分别向 Netizen 与 FriendRelation 实体查询双方信息和关系状态；

校验通过后创建新的 ChatSession，并初始化通信状态。

发送文本消息（6）

网民通过 UI 输入消息并发送，UI 将 sendText 请求传递给控制器处理。

敏感词过滤与合规记录（7–10）

控制器调用 ComplianceController 执行敏感词过滤（7）；

过滤后内容写入 Message（8），若触发违规则记录到 ComplianceRecord（9）；

ComplianceController 再将过滤结果返回给控制器（10）。

消息分发到前端界面（11）

控制器调用 deliver 将最终消息投递回双方的 NetizenChatUI，完成一次文本消息收发。

语音/视频媒体会话处理（12–14）

若为语音/视频模式，控制器调用 MediaSessionController 建立媒体通道（12）；

媒体控制器创建或更新对应的 MediaStream（13），并与当前会话绑定（14）。

媒体异常检测与用户提示（15–16）

当媒体流检测到网络错误或连接中断时，媒体控制器向 NotificationPanel 发送异常通知（15）；通知面板随后调用 showWarning 将异常状态展现给网民（16）。

2.4. 分析类说明

根据用况事件流，识别出以下三类分析类：

（1）Boundary（边界类）

类名	说明
NetizenChatUI	网民通信界面。负责展示联系人列表、消息列表，提供“选择好友”“发送消息”“发起语音/视频”“结束通信”等操作入口。
NotificationPanel	异常提示界面。用于显示网络中断、语音/视频异常、对方忙碌等状态提醒。

（2）Control（控制类）

类名	说明
CommunicationController	通信用况的核心流程控制器，负责建立会话、校验好友关系、转发消息、控制通信模式切换与结束。
ComplianceController	敏感词过滤与违规处理的业务控制器，负责过滤文本消息并记录合规信息。
MediaSessionController	语音/视频媒体通话的控制器，负责媒体流的建立、绑定、重连与异常检测。

（3）Entity（实体类）

类名	说明
Netizen	网民账号信息（ID、昵称、在线状态）。
FriendRelation	好友关系记录，用于判断双方是否允许通信。
ChatSession	会话实体，记录通信模式、参与者、时间与状态。
Message	文本消息记录，包括内容、时间戳及过滤标记。
ComplianceRecord	违规消息或敏感词触发记录。
MediaStream	语音/视频媒体流信息，包括连接参数与状态。

2.5. 通信用况分析类图

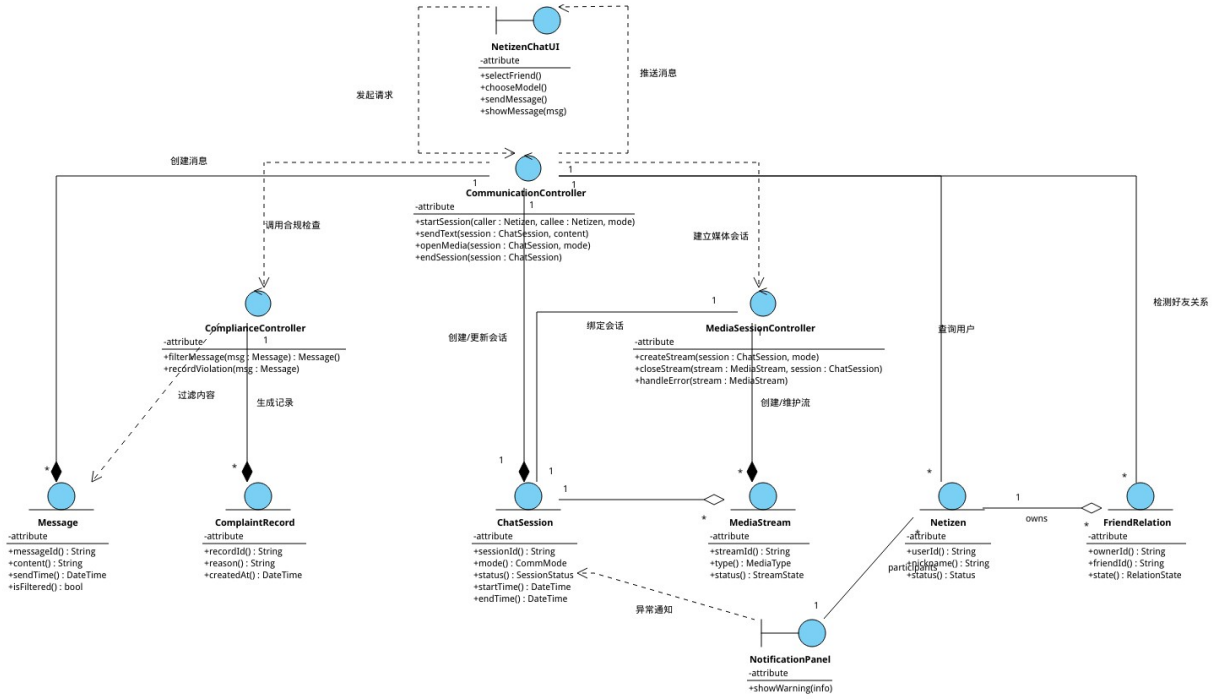


图 8 通信用况分析类图

2.6. 小结

通信用况的健壮性分析明确了系统在执行 UC-1 时所需的主要分析类及其角色划分，并给出了对象间的静态关系和动态协作过程。通过协作图、通信图与分析类图的结合，本用况的业务流与系统内部结构得到了清晰的建模描述，为后续的架构设计与顺序图建模奠定了基础。

3. 数字支付用况（UC-1）的健壮性分析

3.1. 用况简述

UC-2 数字支付用况描述：网民通过系统完成一笔单笔订单的支付。系统需接收支付请求、执行风控与合规校验、向在线网银系统发起授权、处理结果入账与通知，并保证交易可追溯与对账一致性。该用况包含多个子流与备选流，包括：风控与合规校验、银行授权处理、结果入账与通知、异常补偿与对账，以及风控拒绝、余额不足、银行超时、用户取消、重复支付、合规疑似等多种异常场景。

3.2. 数字支付用况协作图

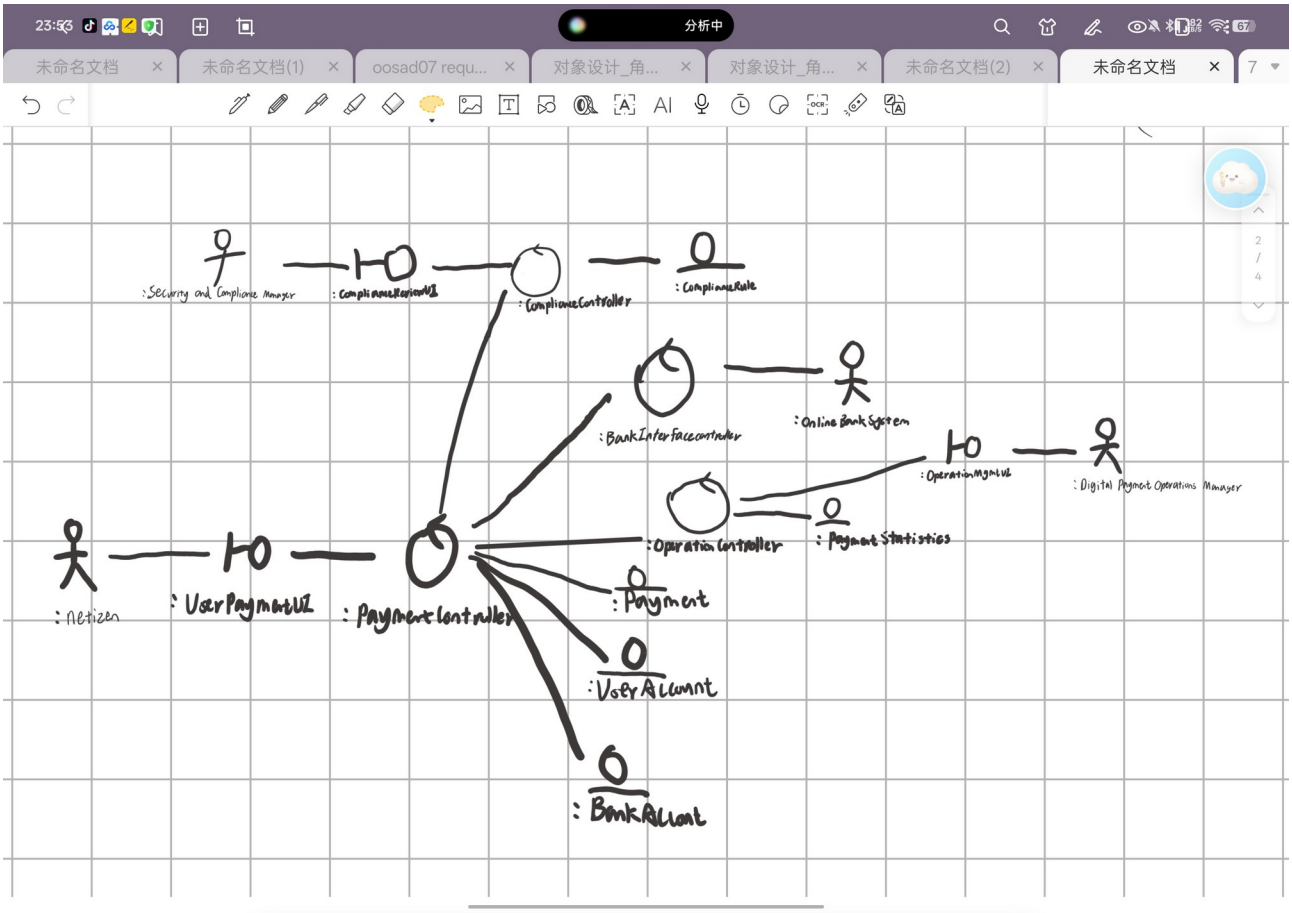


图 9 数字支付协作图

说明：

协作图展示了数字支付流程中各对象之间的协作关系，包括支付请求的接收、风控校验、银行授权、结果处理、异常补偿等关键步骤。每个对象代表一个边界类、控制类或实体类，它们之间通过消息传递实现业务逻辑的流转。

3.3. 数字支付用况通信图

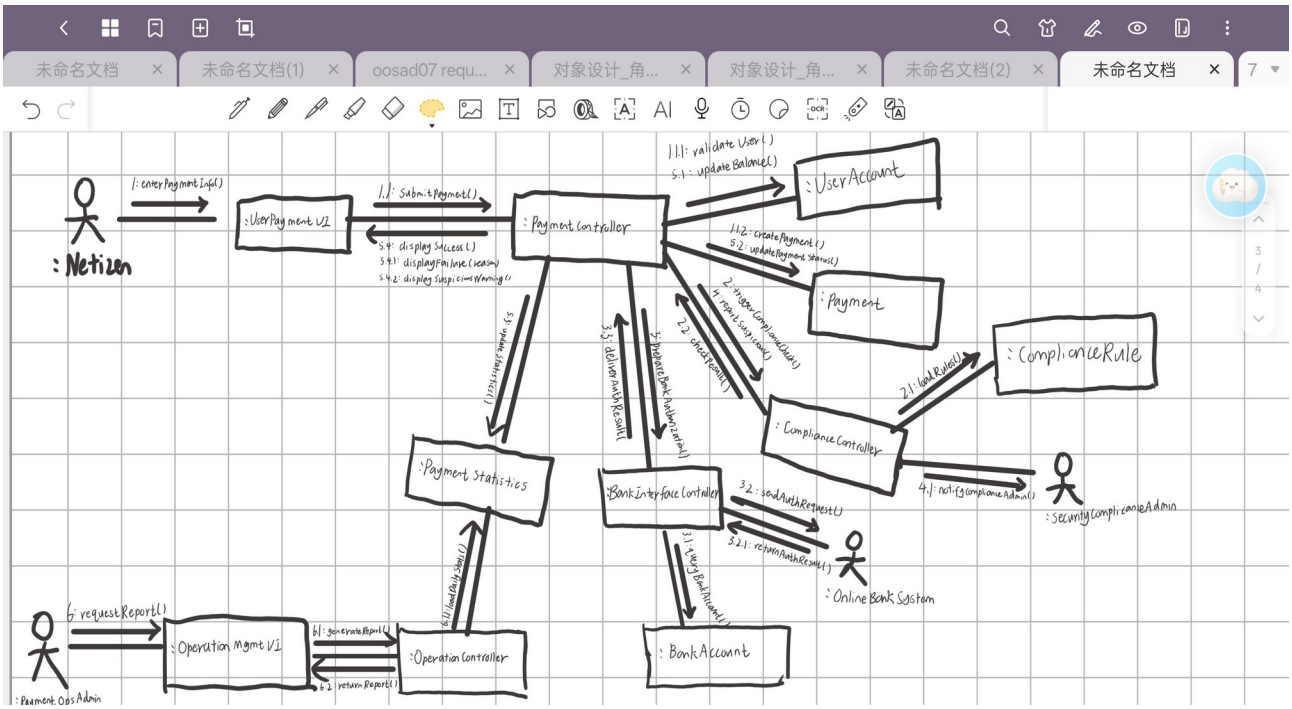


图 10 数字支付通信图

通信图执行过程如下：

提交支付请求 (1-3)

网民在 PaymentUI 选择支付方式并确认支付

UI 将 submitPayment 请求发送给 PaymentController 控制器查询 Order 实体验证订单信息

风控与合规校验 (4-7)

PaymentController 调用 RiskControlController 执行风险检查 (4)

RiskControlController 查询 Netizen 状态、交易历史、设备信息等 (5)

若触发可疑模式，调用 ComplianceController 生成合规工单 (6)

返回风控决策结果给 PaymentController (7)

银行授权处理 (8-12)

PaymentController 调用 BankGatewayController 发起银行授权 (8)

BankGatewayController 组装银行报文并加密签名 (9)

与银行系统建立安全通道并发送请求 (10)

接收银行响应，解析授权结果 (11)

更新 PaymentTransaction 状态并返回结果 (12)

结果入账与通知 (13-18)

PaymentController 调用 AccountingController 执行记账操作 (13)

AccountingController 更新账户余额并生成账务分录 (14)

PaymentController 通过 NotificationPanel 向网民发送结果通知 (15)

向商户系统发送支付成功通知 (16)

若涉及风险事件，向安全管理员发送告警 (17)

更新 Order 状态为"已支付" (18)

异常补偿与对账 (19-22)

对于"待确认"交易，触发 ReconciliationController 执行对账 (19)

从银行系统拉取交易清单进行比对 (20)

识别差异交易并执行补记账或冲正操作 (21)

生成对账报告并通知运营管理员 (22)

3.4. 分析类说明

根据用况事件流，识别出以下三类分析类：

（1）Boundary（边界类）

类名	说明
PaymentUI	支付操作界面。负责展示订单信息、支付方式选择、密码输入、支付结果展示等功能。
NotificationPanel	通知与提示界面。用于显示支付成功/失败结果、风险提示、验证码输入等交互。
MerchantNotificationAPI	商户通知接口。负责向商户系统异步发送支付结果通知。
AdminDashboard	管理员仪表板。展示交易监控、风险告警、合规工单等管理功能。

（2）Control（控制类）

类名	说明
PaymentController	支付流程主控制器。协调整个支付过程的各个步骤，包括请求验证、风控检查、银行授权、结果处理等。
RiskControlController	风险控制控制器。执行交易风险评估、反欺诈检查、限额验证、黑名单查询等风控逻辑。
ComplianceController	合规控制器。处理反洗钱检查、可疑交易识别、合规工单生成、监管报告等合规业务。
BankGatewayController	银行网关控制器。负责与在线网银系统的交互，包括报文组装、加密签名、通道选择、响应解析等。
AccountingController	账务控制器。处理资金记账、余额更新、账务分录生成、电子回执创建等会计操作。
ReconciliationController	对账控制器。执行交易对账、差异识别、补偿处理、对账报告生成等对账业务。
RetryController	重试控制器。管理银行超时重试、通知重发、幂等性控制

类名	说明
	等重试逻辑。

（3）Entity（实体类）

类名	说明
Netizen	网民账户信息。包括用户 ID、实名信息、KYC 状态、支付账户余额等。
Order	订单实体。记录订单号、金额、商品信息、商户 ID、订单状态等。
PaymentTransaction	支付交易记录。包含交易 ID、订单号、支付金额、支付方式、交易状态、时间戳等核心字段。
PaymentAccount	支付账户信息。包括账户余额、支付限额、绑定银行卡列表、账户状态等。
RiskProfile	风险画像。记录用户的风险评分、交易习惯、设备指纹、地理位置等风控相关数据。
ComplianceRecord	合规记录。存储可疑交易报告、合规工单、监管报送记录等合规信息。
BankResponse	银行响应记录。保存银行授权结果、授权码、错误代码、响应时间等银行交互数据。
AccountingEntry	账务分录。记录每笔交易的借贷方账户、金额、余额变化、记账时间等会计信息。
AuditLog	审计日志。记录所有支付操作的审计轨迹，包括操作人、操作时间、操作内容、IP 地址等。
ReconciliationRecord	对账记录。存储对账任务、比对结果、差异处理、对账报告等信息。

3.5. 数字支付用况分析类图

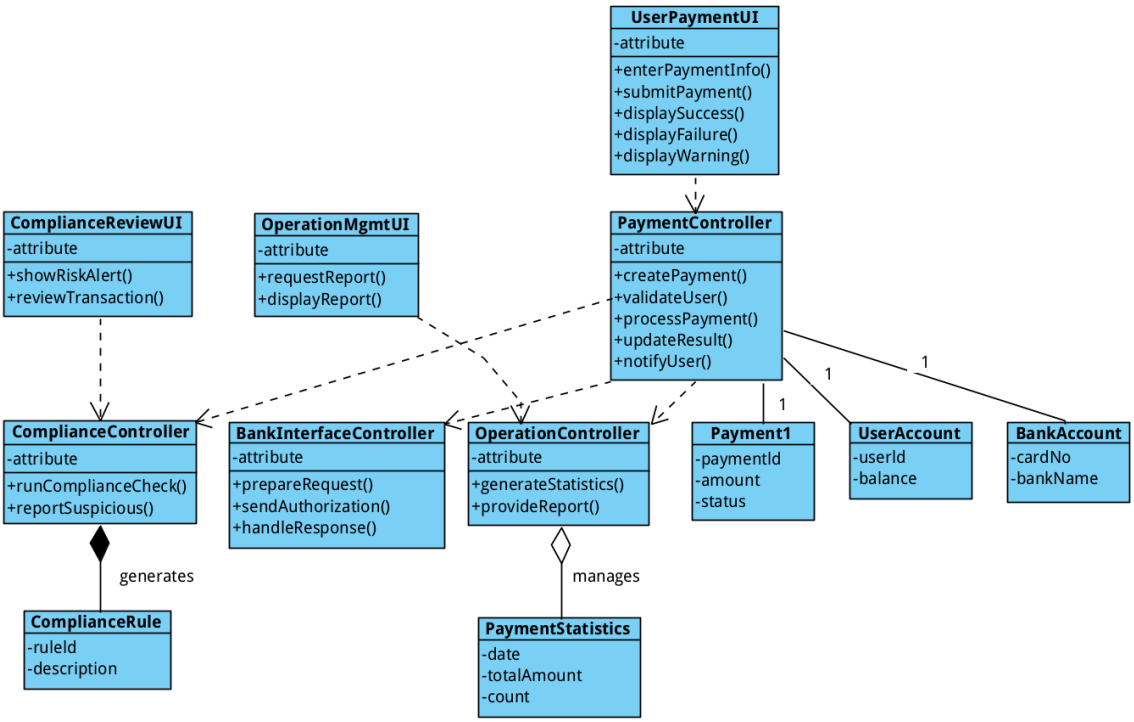


图 11 数字支付类图

3.6. 小结

数字支付用况的健壮性分析明确了系统在执行 UC-2 时所需的主要分析类及其角色划分，并给出了对象间的静态关系和动态协作过程。通过协作图、通信图与分析类图的结合，本用况的复杂支付流程与系统内部结构得到了清晰的建模描述，为后续的架构设计、顺序图建模以及具体的实现提供了坚实的基础框架。特别强调了异常处理、状态管理、审计追溯和补偿机制等关键设计点，确保支付系统的高可用性、一致性和合规性。

4. 管理朋友圈用况（UC-3）的健壮性分析

本节基于 UC-3 管理朋友圈用况的基本流与备选流，对系统在该用况中的核心对象、对象类型（Boundary/Control/Entity）及对象间协作关系进行分析，并结合协作图、通信图及分析类图给出系统执行该用况时的结构化行为视图。

4.1. 用况简述

UC-3 管理朋友圈用况描述了网民发布朋友圈动态、与好友进行互动及对自有动态执行管理操作的完整流程。系统需完成动态编辑与权限配置、合规检测、动态发布与同步、互动操作处理、动态管理及违规处置等任务，同时保障内容隐私与合规性。

4.2. 管理朋友圈用况协作图

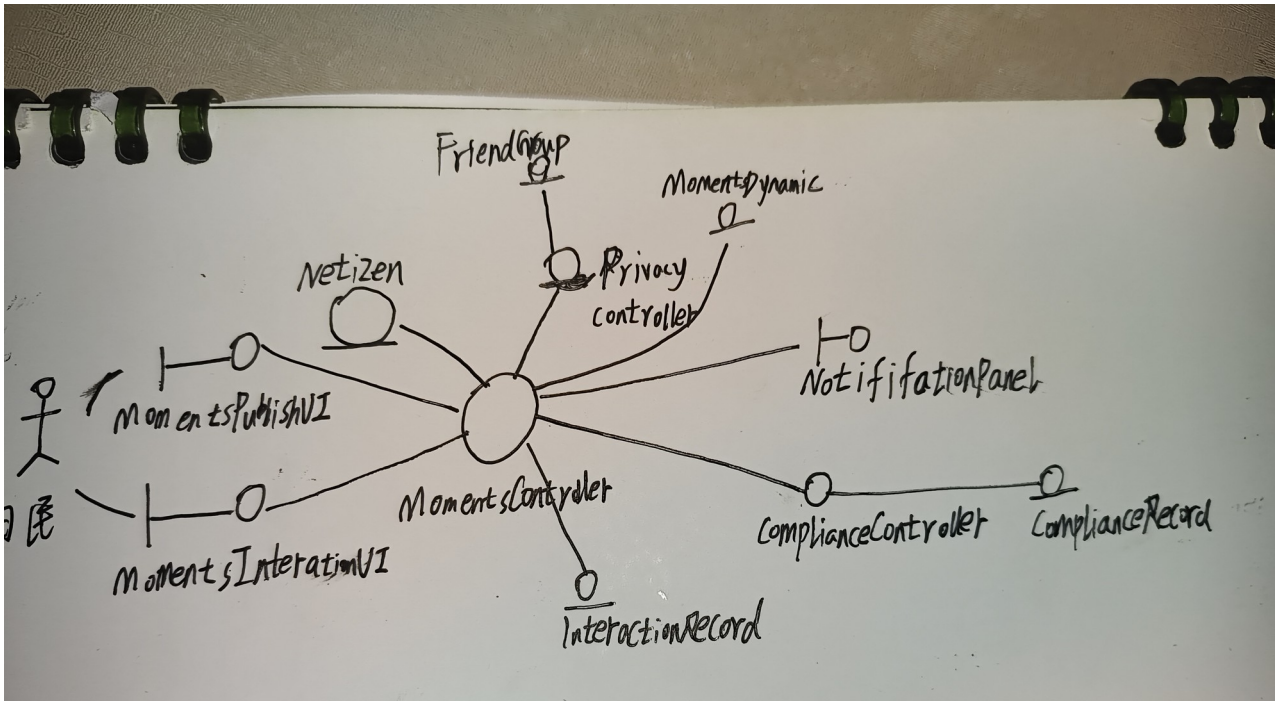


图 12 管理朋友圈协作图

4.3. 管理朋友圈用况通信图

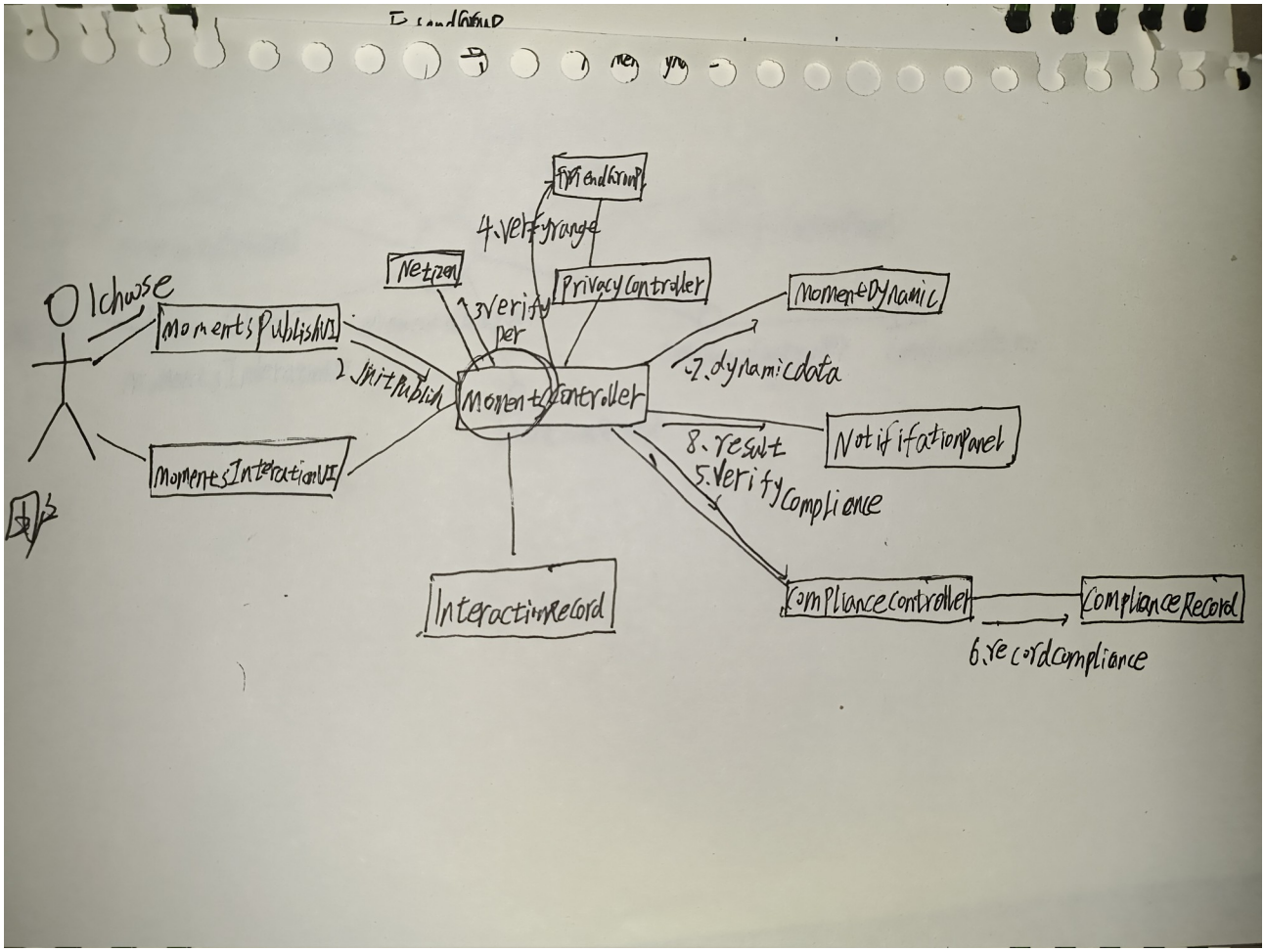


图 13 管理朋友圈通信图

通信图执行过程如下：

选择发布类型并初始化草稿（1-2）

网民在 MomentsPublishUI 中选择发布文本 / 图片 / 短视频类型，触发 initPublishDraft 请求至核心控制器 MomentsController，系统自动生成本地发布草稿。

配置可见范围并校验（3-5）

网民设置动态可见范围，UI 将请求转发至 PrivacyController；隐私控制器调用 FriendGroup 实体校验分组及好友权限，校验通过后将范围配置保存至待发布的 MomentsDynamic 实体。

提交动态并执行合规检测（6-8）

网民确认发布，UI 将 submitDynamic 请求传递给 MomentsController；控制器调用 ComplianceController 执行敏感词与违规信息检测，若检测到违规内容则记录至 ComplianceRecord。

保存并同步动态（9-10）

合规检测通过后，MomentsController 将动态内容写入 MomentsDynamic 实体完成持久化；同时调用 syncToFriends 接口，将动态同步至符合可见范围的好友 MomentsInteractionUI。

动态互动操作（11-15）

点赞：好友在互动界面执行点赞操作，请求传递至 MomentsController，控制器调用 InteractionRecord 实体记录点赞状态并同步至发布者端；

评论：好友提交评论后，控制器先调用 ComplianceController 完成评论合规检测，检测通过后将评论内容保存至 InteractionRecord，并同步给发布者及有权限查看的好友。

动态管理操作（16-19）

编辑：发布者 1 小时内发起编辑请求，MomentsController 调用 MomentsDynamic 实体更新文本及可见范围信息；

删除：发布者执行删除操作，控制器将动态删除，并同步移除所有好友端展示内容。

操作结果提示（20）

所有操作完成后，系统通过 NotificationPanel 向网民反 git config --global http.sslVerify false 馈发布 / 互动 / 管理的最终结果，若出现异常则同步推送提醒。

4.4. 分析类说明

根据用况事件流，识别出以下三类分析类：

（1）Boundary（边界类）

类名	说明
MomentsPublishUI	朋友圈发布界面。负责提供文本编辑、图片 / 短视频上传入口，支持可见范围配置，发起动态发布 / 草稿保存请求。
MomentsInteractionUI	朋友圈互动界面。展示好友动态列表，提供点赞、评论、回复等互动操作入口，同步展示互动通知。
NotificationPanel	操作结果提示界面。用于显示动态发布成功 / 失败、合规检测违规、权限校验失败等状态提醒。

（2）Control（控制类）

类名	说明
MomentsController	朋友圈用况的核心流程控制器，负责协调动态发布、互动处理、动态编辑 / 删除等全流程，对接各子控制器完成业务逻辑。
PrivacyController	隐私权限控制的业务控制器，负责校验动态可见范围配置、保障好友权限生效，防止越权访问。
ComplianceController	朋友圈内容合规控制器，负责对动态及评论内容执行敏感词过滤与违规检测，生成合规记录。

（3）Entity（实体类）

类名	说明
Netizen	网民账号信息（ID、昵称、朋友圈权限状态），用于校验发布者及互动者的账号有效性。
FriendGroup	好友分组信息，支持按分组配置动态可见范围，为隐私校验提供数据支撑。
MomentsDynamic	朋友圈动态实体，记录动态内容（文本 / 图片 / 视频）、发布时间、可见范围、状态（正常 / 已删除）等信息。
InteractionRecord	互动记录实体，保存动态的点赞、评论、回复信息，包含操作人 ID、操作时间及内容。
ComplianceRecord	朋友圈合规记录实体，记录动态 / 评论的违规类型、检测时间、处置结果等合规审计信息。

4.5. 管理朋友圈用况分析类图

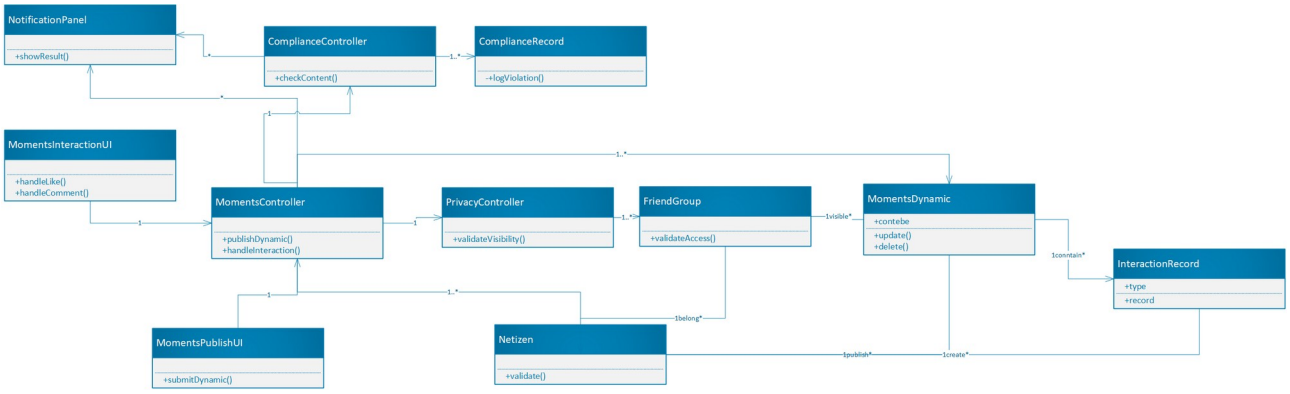


图 14 管理朋友圈类图

4.6. 小结

管理朋友圈用况的健壮性分析明确了系统执行 UC-3 时所需的核心分析类及角色划分，通过协作图、通信图与分析类图的结合，清晰呈现了动态发布、互动、管理全流程的业务逻辑与系统内部结构，为后续架构设计与详细建模提供了核心依据。

4.2. 交互建模

1. 交互建模——UC1

交互建模是在健壮性分析基础上，进一步用 UML 交互图（顺序图、通信图、交互概述图等）精确刻画系统在执行用况时内部对象之间的消息传递、控制流程与并发 / 异常处理机制。对于 UC-1，我们重点关注：

文本通信的消息收发过程；

语音 / 视频媒体会话的建立与维护过程；

（1）参与对象

Actor：Netizen（主叫 / 目标均可）

Boundary：NetizenChatUI

Control：CommunicationController、ComplianceController

Entity：Netizen、FriendRelation、ChatSession、Message、ComplianceRecord

1.1. 顺序图

（1）文本通信顺序图

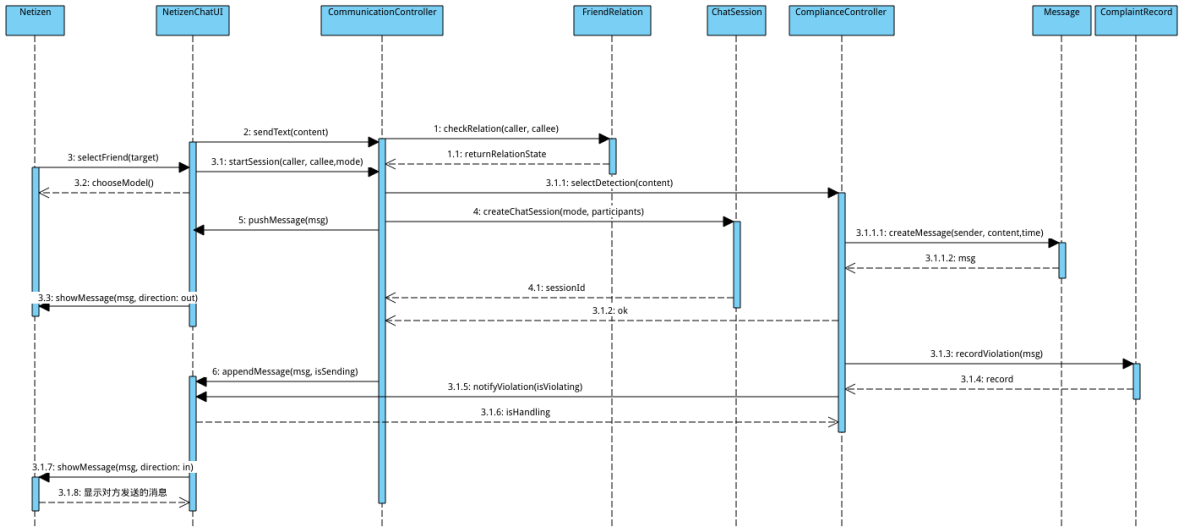


图 15 文本通信顺序图

（2）语音/视频通信顺序图

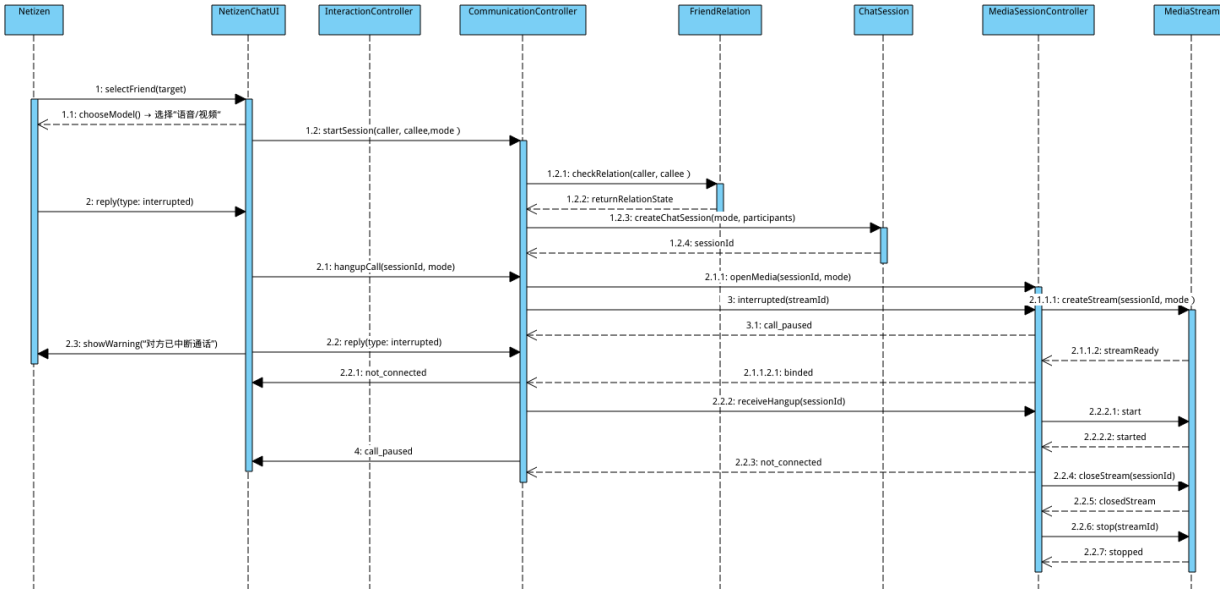


图 16　语音/视频通信顺序图

1.2. 通信图

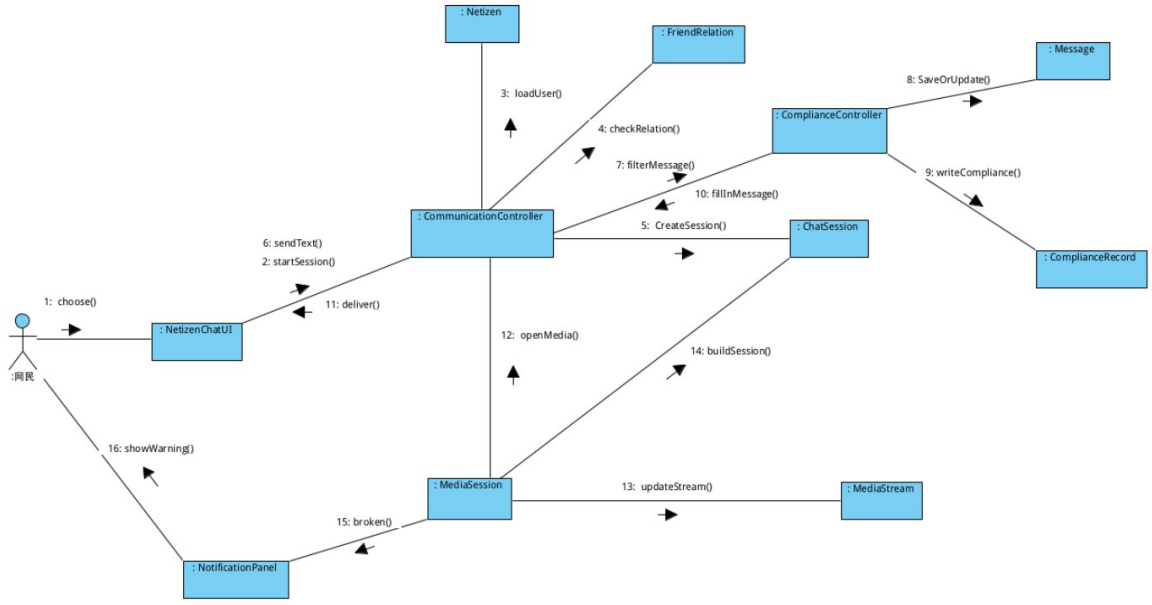
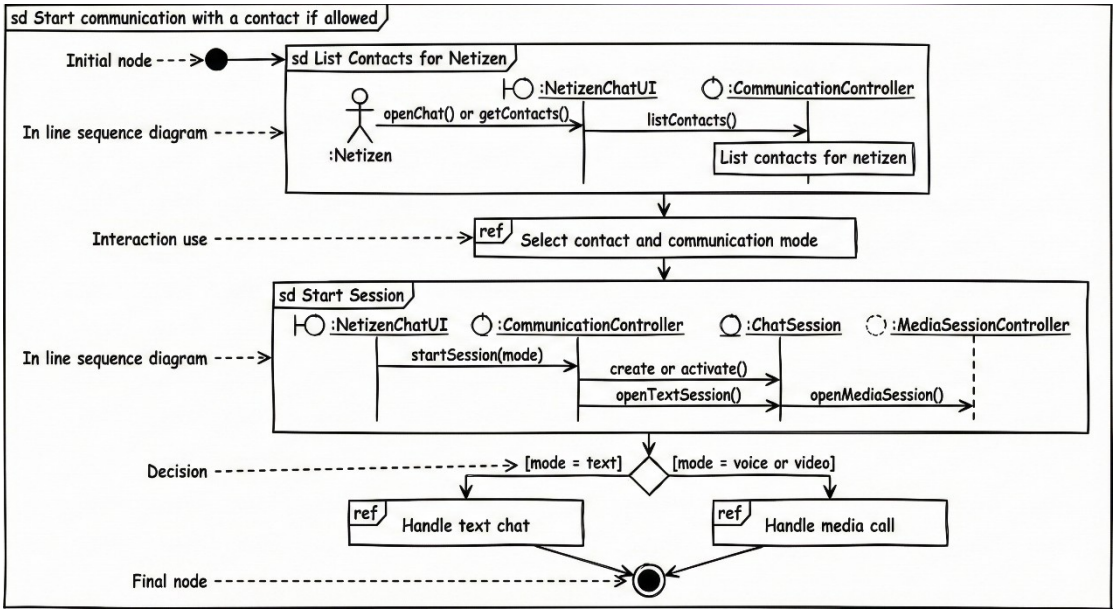


图 17　通信图

1.3. 交互概述图



交互概述图说明：

（1）列出联系人 —— 嵌入式顺序图 sd List Contacts for Netizen

流程从左上角的实心圆初始节点开始，进入第一个嵌入式顺序图 **sd List Contacts for Netizen**。其中包含三个参与者：

:Netizen（用户）

:NetizenChatUI（通信界面）

:CommunicationController（通信控制器）

执行步骤如下：

网民进入聊天界面并执行 openChat() 或 getContacts()。

NetizenChatUI 将请求转发给 CommunicationController。

控制器执行 listContacts() 并返回联系人列表。

图中使用了一个 ref List contacts for netizen 来表示联系人查询的内部过程。

该阶段的目标是为后续的联系人选择提供数据基础。

（2）选择联系人及通信模式 —— 交互使用 ref

在列出联系人之后，控制流进入一个独立的交互使用（ref）框：

ref Select contact and communication mode

用户在此阶段完成：

选择具体的联系人；

选择通信模式（文本模式、语音通话或视频通话）。

该步骤属于静态 UI 与用户交互逻辑，因此在交互概述图中以独立的 ref 表示。

（3）开始会话 —— 嵌入式顺序图 sd Start Session

选择联系人和模式后，流程进入第二个嵌入式顺序图 **sd Start Session**。
其中涉及的对象包括：

:NetizenChatUI

:CommunicationController

:ChatSession

:MediaSessionController

主要执行逻辑如下：

NetizenChatUI 调用 startSession(mode) 发起建立会话请求。

CommunicationController 创建或激活 ChatSession。

如果当前是文本模式，则执行 openTextSession()；

如果是语音或视频，则向 MediaSessionController 发起 openMediaSession() 以准备媒体会话。

该嵌入式顺序图体现了系统内部控制与实体对象之间的消息交互。

（4）决策节点 —— 根据通信模式分支

顺序图执行结束后，控制流进入一个菱形决策节点，用于判断当前选择的通信模式：

[mode = text]
→ 进入文本聊天流程。

[mode = voice or video]
→ 进入媒体通话流程（语音 / 视频）。

该分支将通信用况分成两类不同的行为路径。

（5）文本聊天或媒体通话 —— 两个 ref 分支

根据决策结果，控制流将进入两个不同的交互使用：

ref Handle text chat

表示系统进入文本（消息）通信过程，包含消息发送、接收、过滤、展示等逻辑。

ref Handle media call

表示系统进入媒体通话处理流程，包括语音/视频编码、媒体流建立、网络状态维护和通话控制等行为。

（6）最终节点

两个分支最终都会返回到控制流并指向最终节点（双环终结符号）。
这表示“开始通信”的用况执行完毕，系统状态进入已建立会话状态。

第5章 架构设计

5.1. 文档目的

本文档聚焦基于 Qt 框架开发的即时通讯系统 WeChat 的架构设计，明确系统整体结构、核心技术选型、分层与子系统划分、物理部署方案等关键内容，为系统开发提供宏观设计依据，确保系统满足即时通讯核心业务需求，同时具备良好的可扩展性、高可用性和安全性。

5.2. 架构设计概述

1. 明确系统设计目标

基于需求确定核心优化方向，明确架构需支撑的核心业务场景。

性能目标：支持 10 万用户同时在线，单条消息端到端响应时间≤100ms，附件传输（≤100MB）稳定无超时；

可扩展性目标：支持后续新增小程序等功能模块，无需重构核心架构；

安全性目标：实现用户身份可信认证、消息传输加密、敏感数据（手机号、聊天记录）安全存储，防范消息篡改、泄露等风险；

核心业务场景支撑：覆盖用户注册 / 登录、联系人添加 / 管理、单聊 / 群聊、消息收发 / 撤回 / 转发、附件上传 / 下载、会话同步等核心场景。

2. 架构设计原则

遵循分层、模块化、高内聚低耦合等原则，确定架构的灵活性、可维护性等优先级。

分层解耦原则：遵循表现层、业务逻辑层、数据访问层、通信层分层设计，层间通过标准化接口交互，降低耦合度；

模块化与高内聚原则：将核心功能拆解为独立模块，每个模块聚焦单一职责，减少模块间交叉依赖；

灵活性优先原则：结合即时通讯业务高频迭代、功能快速落地的特性，优先保障架构的灵活性和可适配性；

高可用与容错原则：针对消息收发、会话同步等核心场景，设计容错机制，避免单点故障影响整体服务；

符合 Qt 框架设计规范：遵循 Qt 的 MVC/MVVM 设计思想、信号与槽机制，贴合框架生态，降低团队开发与维护成本。

5.3. 系统边界与范围

界定系统核心功能边界：明确系统需实现的核心业务功能，排除非核心、暂不实现的功能。

外部系统交互说明：梳理与外部系统的接口（如第三方服务、遗留系统），明确交互协议、数据格式及依赖关系。

5.4. 逻辑架构设计

1. 逻辑架构概览

用架构图展示系统的逻辑分层、核心模块 / 子系统划分，说明各部分的核心职责。

2. 架构风格选型

基于业务场景选择合适的架构风格。

分层架构：划分表现层、业务逻辑层、数据访问层，明确各层职责。

架构风格选型依据：详细说明为何选择当前架构风格（如分层架构、MVC、微服务），对比备选风格的优劣（如 “采用 MVC 架构而非单纯分层，因前端交互复杂，需分离视图与控制逻辑”

架构风格落地规则：明确风格对应的分层 / 模块职责边界（如 “MVC 架构中 Model 仅包含业务逻辑，View 不处理业务规则，Controller 仅负责请求分发”），避免职责混淆。

3. 模块划分与职责

按 “分层 + 分区” 策略拆分子系统，明确各子系统的高层职责、边界与依赖关系，不涉及具体模块内部类设计：

核心业务子系统：对应案例 5 的核心业务模块（如业务处理、数据管理）；

支撑子系统：包括用户认证、日志审计、配置管理等通用模块；

接口子系统：负责与外部系统或前端的交互适配。

4. 领域模型设计

定义核心业务实体的高层关系（继承、关联、聚合），明确实体核心职责，不涉及具体属性与方法细节：

5. 接口设计

定义模块间、系统与外部交互的高层接口规范，不涉及具体参数与实现细节：

接口名称、功能描述；

输入 / 输出参数（字段、类型、约束）；

通信协议（如 HTTP、RPC）、调用方式（同步 / 异步）；

异常处理规则。

接口版本控制：说明接口版本管理策略（如 URL 路径版本、参数版本），适配需求变更与兼容性要求。

接口安全设计：补充接口认证（如 Token 校验）、权限控制、限流策略（如单 IP 每秒最多 5 次调用）

6. 关键流程设计

用时序图 / 活动图展示核心业务流程的高层逻辑协作关系，明确模块间交互顺序，不涉及具体代码逻辑：

7. 状态机设计

明确需状态机建模的核心业务对象，定义高层状态与转移规则，不涉及具体状态触发的代码实现：

适用对象说明：列出需要状态机建模的核心业务对象。

状态与转移定义：用状态机图展示对象生命周期的核心状态、触发事件、监护条件及效应（如 “订单待支付状态→已支付状态，触发事件为支付成功回调，效应为更新订单状态 + 扣减库存”）。

并发状态处理：若对象存在正交子状态（如 “活动同时处于运行状态和监控状态”），说明并发状态的协作规则。

5.5. 物理架构设计

1. 物理架构概览

用部署图展示硬件节点、网络拓扑、软件部署分布，说明物理层（Tier）划分（如 “客户端层、应用服务器层、数据服务器层”）。

2. 硬件资源规划

根据子系统负载特性，规划高层硬件资源分配，不涉及具体硬件参数细节：

处理器分配：根据子系统负载特性，分配对应的硬件资源（如核心业务子系统部署在高性能节点）。

存储资源规划：确定数据存储方案（如数据库分区、缓存策略），估算存储容量需求（第一本书 14 章 “估算硬件资源需求”）。

3. 软件部署方案

明确各软件组件的部署位置、实例数量与核心部署要求，不涉及具体配置参数：

4. 网络架构设计

说明网络分区（如 “内网、DMZ 区”）、通信协议（如 TCP/IP）、安全策略（如防火墙规则、端口开放限制），保障数据传输安全与稳定性。

5. 子系统交互设计

明确子系统间通信机制的高层规范，不涉及具体数据传输格式细节：

6. 并发设计

识别系统核心并发场景，定义高层并发控制策略，不涉及具体线程 / 锁实现：

7. 数据存储设计

确定数据存储策略的高层选择，不涉及具体表结构、索引设计：

8. 序列化方案设计

明确序列化技术选型与高层传输规范，不涉及具体序列化字段定义：

序列化技术选型：明确分布式场景下的序列化方式（如 Protobuf、JSON、Avro），说明选型理由（如 “微服务通信采用 Protobuf，兼顾性能与跨语言兼容性”）

数据传输规范：定义序列化数据的格式、压缩策略、校验规则（如 “传输数据采用 LZ4 压缩，添加 CRC 校验确保完整性”）。

5.6. 质量属性保障设计

1. 性能保障

性能目标定义：明确响应时间、并发量、吞吐量等核心性能指标，不涉及具体优化代码：。

优化措施：针对性能瓶颈（如数据库查询、接口调用），设计优化方案（如缓存优化、SQL 优化、异步处理）。

2. 安全保障

设计高层安全机制，不涉及具体加密算法实现：

身份认证与授权：设计用户身份校验、权限分级管理机制，控制不同角色对资源的访问权限。

数据安全：明确数据传输加密、存储加密、敏感数据脱敏策略，防范数据泄露风险。

防护措施：针对常见安全威胁（如注入攻击、跨站请求伪造），设计防护方案。

3. 可扩展性保障

说明架构对业务扩展、用户增长的支持方案（如模块拆分、接口预留、水平扩展能力）。明确架构扩展策略，不涉及具体扩展接口实现：

架构扩展点设计：列出预留的扩展接口与扩展机制（如 “新增支付方式时，通过实现统一支付接口接入，无需修改核心业务逻辑”），呼应课件中 “领域模型可扩展性” 要点。

水平 / 垂直扩展支持：说明架构如何支撑用户量增长（水平扩展）、功能新增（垂直扩展）（如 “应用服务器支持集群扩容，业务层支持按模块拆分独立部署”）。

4. 可维护性保障

设计规范（代码规范、接口规范）、监控告警机制（如日志采集、异常监控）、部署与升级流程。定义高层可维护性策略，不涉及具体规范细节：

5. 兼容性保障

明确架构兼容策略，不涉及具体兼容代码实现：

版本兼容性：说明架构对不同版本客户端、遗留系统的兼容策略（如 “接口向下兼容 V1 版本，遗留系统通过适配层对接”）。

跨环境兼容性：明确系统在不同操作系统、硬件平台的适配方案（如 “应用服务器支持 Linux/Windows，数据库兼容 MySQL 5.7+/8.0+”）。

6. 高可用与容灾设计

设计高层高可用与容灾机制，不涉及具体故障处理代码：

高可用措施：设计服务冗余、故障自动恢复、负载均衡等机制（参考第一本书 14 章 “高可用与容灾设计” 思想），如关键子系统多实例部署。

容灾方案：制定数据备份策略（如定时备份、异地备份）、故障切换流程、灾难恢复预案，明确 RTO（恢复时间目标）、RPO（恢复点目标）。

故障检测机制：说明系统故障识别方案（如心跳检测、健康检查接口、日志告警）。

降级与熔断策略：补充核心业务降级规则（如非核心功能故障时自动屏蔽）、熔断阈值（如接口失败率超过 50% 时触发熔断），避免级联故障。

5.7. 控制策略设计

根据业务特性选择高层控制模式，不涉及具体控制逻辑实现：

核心控制模式：根据业务特性选择控制策略（第一本书 14 章 “选择软件控制策略”），如：

事件驱动型控制：适用于案例 5 中基于用户操作触发的业务流程；

过程驱动型控制：适配固定步骤的业务流程（如数据校验、审批流程）。

边界条件处理：明确异常场景（如超时、数据无效、系统故障）的处理策略，定义降级、熔断机制。

5.8. 架构验证与风险说明

1. 架构验证方案

设计验证方法（如原型验证、压力测试、场景演练），验证架构是否满足性能、可用性等目标，列出验证指标与预期结果，不涉及具体测试用例：

状态机验证：说明状态机逻辑的验证方式（如场景演练、边界条件测试），确保状态转移无死锁、无遗漏（如 “测试订单从待支付→已取消→待支付的异常转移是否被禁止”）。

并发场景验证：明确并发场景的验证方案（如压力测试、并发冲突测试），验证并发策略有效性（如 “1000 用户同时下单，测试库存数据一致性与响应时间”）。

2. 风险与应对措施

梳理架构设计中的潜在风险（如技术选型风险、性能风险、依赖风险），制定对应的应对措施 ，不涉及具体风险处理代码：

风险类型	风险描述	应对措施
性能风险	高并发下响应时间过长	引入缓存、优化数据库查询、增加应用服务器实例
依赖风险	第三方中间件不可用	预留备选中间件，设计降级策略
状态机风险	状态转移逻辑遗漏，导致业务流程卡壳	梳理全量业务场景，绘制状态机图并评审，开发阶段增加状态校验
序列化风险	跨语言序列化兼容性问题	提前进行多语言序列化测试，制定统一的序列化协议规范
并发冲突风险	高并发下数据竞争导致脏写	采用乐观锁 + 重试机制，核心业务加分布式锁

5.9. 架构演进规划

短期演进：明确上线初期的架构迭代方向（如功能完善、性能调优）。

长期演进：规划架构的扩展路径（如子系统拆分细化、新技术引入），适配业务未来发展需求。

第6章详细设计

6.1. 文档目的

本文档聚焦基于 Qt 框架开发的即时通讯系统 WeChat 的详细设计，明确核心类的定义、属性、方法、协作关系，以及具体接口参数、数据结构、状态机细节等，为开发人员提供可直接落地的技术实现依据，确保系统设计与编码一致。

6.2. 核心类设计

1. 表现层核心类设计

明确表现层类的属性、方法及协作关系，聚焦界面交互与请求分发：

2. 业务逻辑层核心类设计

明确业务逻辑层类的属性、方法及协作关系，聚焦业务规则实现：

3. 数据访问层核心类设计

明确数据访问层类的属性、方法及协作关系，聚焦数据读写适配：

4. 通信层核心类设计

明确通信层类的属性、方法及协作关系，聚焦网络通信与协议处理：

5. 支撑层核心类设计

明确支撑层类的属性、方法及协作关系，聚焦通用服务实现：

6.3. 领域模型详细设计

1. 核心实体类详细设计

明确核心业务实体的完整属性、方法及关系，体现业务逻辑抽象：

1.1. User（用户实体）

核心方法：

1.2. Message（消息实体）

核心方法：

1.3. Conversation（会话实体）

核心方法：

2. 实体关系详细设计

用类图明确实体间的详细关系（关联、聚合、继承）：

- User 与 Contact：一对多关联（User 1 → * Contact），关联属性：remark（备注）、groupId（分组 ID）；
- User 与 Conversation：一对多关联（User 1 → * Conversation），关联属性：isTop（置顶）、unreadCount（未读数）；
- Conversation 与 Message：一对多聚合（Conversation 1 → * Message），Message 生命周期依赖于 Conversation；
- Message 与 Attachment：一对多聚合（Message 1 → * Attachment），Attachment 生命周期依赖

于 Message；

- Contact 与 ContactGroup：多对一关联（Contact * → 1 ContactGroup），关联属性：joinTime（加入时间）。

6.4. 接口详细设计

1. 内部模块接口详细设计

基于 Qt 信号与槽机制，明确接口的输入 / 输出参数、触发条件及异常处理：

1.1. 用户模块接口

1.2. 消息模块接口

2. 外部系统接口详细设计

明确与服务端、第三方服务的接口参数、协议格式及异常处理：

与服务端消息发送接口（TCP 协议）

协议格式：[协议头(20 字节)][消息体长度(4 字节)][Protobuf 消息体(n 字节)][校验码(16 字节)]

协议头字段：

异常处理：连接超时（错误码 10001）、服务端繁忙（10002）、消息校验失败（10003）。

6.5. 关键流程详细设计

1. 消息发送流程详细设计

1.1. 时序图

1.2. 核心代码逻辑（MessageBusinessModule::sendMessage）

6.6. 状态机详细设计

1. Message（消息）状态机详细设计

1.1. 状态枚举与触发事件

1.2. 状态转移表

1.3. 状态机实现核心代码（MessageStateMachine）

6.7. 详细设计补充说明

1. 并发控制详细实现

针对核心并发场景，明确线程管理、锁机制的具体实现：

1.1. 界面与业务线程分离：

表现层类运行在 UI 线程（主线程）；

业务逻辑层、通信层、数据访问层类运行在独立的工作线程（QThread）；

线程间通过 Qt 信号与槽（跨线程连接）通信，避免线程安全问题。

1.2. 数据并发修改控制：

本地数据库（SQLite）：使用 QMutex 锁保护数据库连接，确保同一时间只有一个线程执行写操作；

共享内存数据（如当前用户信息）：使用 QReadWriteLock 读写锁，支持多线程并发读，单线程写。

核心代码示例（SQLite 锁机制）：

2. 序列化详细实现

明确 Protobuf 序列化 / 反序列化的具体字段定义与代码实现：

2.1. 消息 Protobuf 定义（message.proto）：

2.2. 序列化核心代码（MessageCodecModule::encode）

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] Jasmin Blanchette Mark Summerfield 著.C++ GUI Qt4 编程（第二版)

[2] Kurt Bittner, Ian Spence 著. 用况建模

[3] Michael Blala James Rumbangh 著.UML 面向对象建模与设计（第 2 版）

[4] Dean Leffingwell,Don Widrig 著. 软件需求管理用例方法（第 2 版)

[5] Rebecca Wirfs-Brock, Alan McKean 著。对象设计：角色、责任和协作。

[6] James Rumbaugh, Ivar Jacobson, Grady Booch 著. UML 参考手册（第 2 版）