微信简化版需求文档 v1.0

技术目标：实现支持10万+在线用户的IM系统，覆盖90%高频面试考点

一、核心功能需求

1. 用户服务（分布式架构基础）

- 注册/登录
  - 支持手机号+验证码注册（Redis缓存验证码，5分钟过期）
  - JWT令牌鉴权（需实现续签机制）
  - 面试重点：布隆过滤器防止缓存穿透
- 好友关系
  - 实现单向好友模式（类似微信）
  - 好友申请/审批流程
  - 技术难点：
  - ```
    /* 分库分表示例：user_id % 8 */CREATE TABLE user_relation_0 (
        id BIGINT PRIMARY KEY,
        user_id BIGINT COMMENT '用户ID',
        friend_id BIGINT COMMENT '好友ID',
        status TINYINT COMMENT '0-待确认 1-已好友'
    ) ENGINE=InnoDB PARTITION BY HASH(user_id % 8);
    ```

2. 即时通讯（高并发核心）

- 消息类型
    ```

消息类型	技术方案
单聊文本	Protobuf编码 + Netty长连接
群聊消息	Redis Pub/Sub + 写扩散
大文件	分片上传（MD5秒传）

- 消息可靠性
  - 实现ACK确认机制（类似TCP）
  - 离线消息存储（Redis SortedSet + MySQL冷备）
  - 面试考点：
  - ```
    // 消息重传设计public void resendMessage(Channel channel, Message msg) {
        if (channel.isActive()) {
            channel.writeAndFlush(msg).addListener(future -> {
                if (!future.isSuccess()) { // 失败后加入重试队列
                    retryQueue.add(new RetryEntry(msg, 3));
                }
            });
        }
    }
    ```

3. 朋友圈（分布式存储实战）

- 发布流程
  a. 写入MySQL主表（分用户ID散列）
  b. 异步写入Redis好友Feed流（推模式）
  c. 冷数据定期归档到HBase
- 浏览权限
  - 实现"仅三天可见"（Redis ZSET按时间戳排序）
  - 技术亮点：
    ```

- ```
  # 伪代码：朋友圈权限检查def check_visible(post_user, viewer, post_time):
      if is_blacklist(post_user, viewer):
          return Falseif post_user.privacy == '3days':
          return time.now() - post_time < 3 * 86400return True
  ```

二、非功能性需求（面试核心）

1. 性能指标

场景	目标QPS	允许延迟	实现手段
单聊消息发送	50,000+	<100ms	Netty零拷贝+多级缓存
万人群消息扩散	5,000/s	<1s	批量写Redis+异步持久化
朋友圈刷新	10,000+	<300ms	多级缓存+本地缓存

2. 容灾要求

- 消息丢失率：<0.001%（Kafka副本数=3）
- 服务可用性：99.99%（K8s滚动升级+跨AZ部署）

3. 安全要求

- 端到端加密：使用Signal协议库
- 防XSS：消息内容HTML转义
- `StringEscapeUtils.escapeHtml4(rawContent);`

三、技术演进路线

Phase 1：基础版（2周）

- 实现单聊+群聊（Netty+MySQL）
- 基础消息存储

Phase 2：优化版（1周）

- 引入Redis缓存消息
  ```

- 实现消息ACK机制

Phase 3：分布式版（2周）

- 分库分表（ShardingSphere）
- 引入Kafka削峰

Phase 4：进阶版（1周）

- 实现朋友圈Feed流
- 接入Prometheus监控

四、面试展示建议

1. 重点展示架构图：
2. ```
   graph TDA[客户端] --> B[Nginx负载均衡]B --> C[Netty网关集群]
       C --> D[Kafka消息队列]
       D --> E[消息处理服务]
       E --> F[Redis集群]
       E --> G[MySQL分库]
   ```
3. 准备三个问题深度回答：
   - "如何解决消息顺序性问题？"
   - （答案：通过分布式SeqID+客户端缓冲）
   - "万人群聊的性能瓶颈在哪里？"
   - （答案：Redis的PUB/SUB单分片吞吐上限）
   - "朋友圈刷新为什么不用拉模式？"
   - （答案：读放大问题导致DB压力）
4. 携带压测报告：

- ```
  JMeter测试结果：
  单机Netty节点：1.2万连接/秒
  消息投递延迟：P99 < 80ms
  Kafka吞吐量：12万条/秒
  ```

这样的需求设计既具备真实项目完整性，又精准命中面试官关注的分布式、高并发、一致性等技术点。建议在README中突出技术决策的权衡过程（比如为什么选择推模式而不是拉模式），这比单纯的功能描述更有价值。