

**《软件测试》**

**期末课程设计**

**恰运动—运动爱好者的运动与社交平台**

**集成测试文档**

小组成员

2253407 柳阳

1951931 刘筱科

2254302 商一诚

2254305 夏济舟

目录

[1. 引言 4](#_Toc8309)

[1.1 目的 4](#_Toc6111)

[1.2 测试范围 4](#_Toc11378)

[1.3 测试环境 4](#_Toc25975)

[1.4 参考文件及工作产品 4](#_Toc24728)

[2. 集成策略 5](#_Toc7508)

[2.1 进入标准 5](#_Toc15029)

[2.2 集成内容 5](#_Toc3589)

[2.3 集成策略 6](#_Toc24361)

[2.4 集成顺序 7](#_Toc25142)

[3. 测试过程描述 7](#_Toc11921)

[3.1 软件集成测试 7](#_Toc24102)

[3.2 子系统集成测试 8](#_Toc4583)

[4. 测试通过/失败标准 8](#_Toc28116)

[4.1 模块验收标准 8](#_Toc30829)

[4.2 集成测试验收标准 9](#_Toc12723)

[5. 工具 9](#_Toc7708)

[5.1 测试工具 9](#_Toc26084)

[5.2 其它工具 9](#_Toc9843)

[6. 挂起，恢复条件 9](#_Toc8256)

[6.1 挂起 9](#_Toc14042)

[6.2 恢复 10](#_Toc956)

[7. 责任人 10](#_Toc21746)

[8. 集成测试用例与报告 11](#_Toc15345)

[8.1 用例分析与设计 11](#_Toc27077)

[8.1.1 后端类间集成测试 12](#_Toc12943)

[附件：测试报告 24](#_Toc27747)

**1. 引言**

**1.1 目的**

本测试大纲旨在明确益寻系统的集成测试流程，确保测试活动有序、完整地开展。集成测试的核心目标是验证系统中不同组件、模块及子系统之间的集成与交互是否正常，确保各组件在集成后能够正确交换信息、协同工作，并满足整体功能和性能需求。

通过全面的集成测试，我们能够提前发现并修复潜在问题，有效降低系统故障风险，提升系统的可靠性、稳定性和可用性。此外，集成测试的结果与问题记录将作为后续问题跟踪、修复以及开发和测试团队优化系统集成和整体质量的重要参考依据。

**1.2 测试范围**

本测试针对软件的集成测试，以预先完成的单元测试为基础，不进行系统测试，不涉及硬件。本次测试的主要内容有：

（1）测试模块间接口的连通性

（2）测试各个子功能的集成结果能否满足父功能的预期要求

（3）测试一个模块的功能是否会对另一模块产生负面影响

（4）测试全局数据结构是否合理

（5）测试模块的误差是否会累加，以致超出接受范围

本次测试的主要方法为黑盒测试，必要集成测试的回归测试。本计划书主要面向项目负责人、集成测试负责人及设计师。

**1.3 测试环境**

|  |  |  |
| --- | --- | --- |
| 序号 | 描述 | 配置 |
| 1# | 操作系统 | Windows 10 |
| 2# | 浏览器 | Chrome/Firefox/Edge |
| 3# | 开发工具 | IntelliJ IDEA 2021.2.2 |
| 4# | 测试工具 | Junit, Cypress |

**1.4 参考文件及工作产品**

开始测试涉及以下文档：

* 《恰运动需求分析规约》
* 《恰运动概要设计规约》
* 《恰运动详细设计规约》
* 《恰运动单元测试报告》

执行测试前涉及的任务：

* 用例设计完成并通过评审
* 测试脚本开发完成
* 测试环境搭建完成
* 测试过程及缺陷管理流程确定

**2. 集成策略**

**2.1 进入标准**

编码完成，单元测试完成。集成测试计划完成，工具及人员安排到位。

**2.2 集成内容**

（1）函数集成

函数间的接口，函数是否调用正常，在本项目中主要是指不同层间函数的集成，如 controller 和 service 层。

（2）功能集成

不同函数间组合实现的业务功能

（3）数据集成

数据间传递是否正确，对于传入值的控制范围是否一致

（4）子系统集成

不同子系统之间的集成。在本项目中，如获取用户好友申请列表时调用UserService模块。

**2.3 集成策略**

为了使测试人员能够更快地定位和修复故障，同时在整个集成过程中保持较高的可控性和可追溯性，本系统的集成测试采用了自底向上的集成方式。这种集成方式的目标是使测试人员更容易定位软件故障的位置，并在测试过程中尽早发现和解决问题。自底向上的集成测试从最底层的组件开始，逐步将上层组件集成进来，逐步构建整个系统。通过这种方式，测试人员可以先对各个独立的组件进行单元测试，并在组件间的集成过程中验证其正确的交互和接口。这种方法允许测试人员在测试进程中重点关注关键和复杂的组件，有助于更早地发现和解决潜在的问题，并提高整个集成测试的效率和质量。对于一个给定层次的模块，其子模块（包括对应的下属模块）事前已经完成组装并经过测试，所以不再需要编制桩模块用于替代尚未完成或不可用的子模块。

集成测试的主要步骤：

|  |  |  |  |
| --- | --- | --- | --- |
| 活动 | 输入 | 输出 | 职责 |
| 制定并审核集成测试计划 | 概要设计规约等 | 集成测试计划 | 制定测试计划 |
| 测试用例的分析、设计和评审 | 集成测试计划  概要设计规约 | 测试用例 | 设计测试用例并评审 |
| 实施测试 | 集成测试用例、测试过程 | 测试脚本  测试环境 | 开发测试脚本，搭建  测试环境 |
| 测试驱动或桩 | 开发驱动或桩 |
| 执行测试 | 测试脚本 | 测试结果 | 记录结果，跟踪缺陷 |
| 测试的分析与评估 | 集成测试计划  测试结果 | 测试分析评估报告 | 会同开发人员评估测试结果，得出测试报告 |

**2.4 集成顺序**

* **软件集成顺序**

自底向上的集成方式。

* **软件/硬件集成顺序**

无。

* **子系统集成顺序**

略。

**3. 测试过程描述**

**3.1 软件集成测试**

在恰运动项目中集成测试的主要过程活动如下：

1. 设计《集成测试用例》。

自底向上集成测试的步骤：

* 确定被测模块：根据概要设计规格说明，明确哪些模块需要进行集成测试。在熟悉被测模块性质的基础上，对它们进行分层，并确定可以并行测试的同一层次上的模块。然后，制定测试进度计划，排除测试活动的先后关系。
* 模块集成：按照时间线序关系逐步将软件单元集成为模块。在集成过程中，对集成产生的问题进行测试和修复，确保各模块之间的接口和交互正常工作。
* 子系统集成：将各个模块集成为子系统，并测试子系统的功能和性能是否正常工作。在这一阶段，重点验证子系统之间的协作和集成情况，确保整体系统的稳定性。
* 最终用户系统集成：将各个子系统集成为最终用户系统，并进行全面测试。在这一阶段，验证各子系统在最终用户系统中的兼容性和互操作性，确保最终用户系统能够正常工作。

1. 集成测试：组织人员按照1中的《集成测试用例》测试系统集成度。

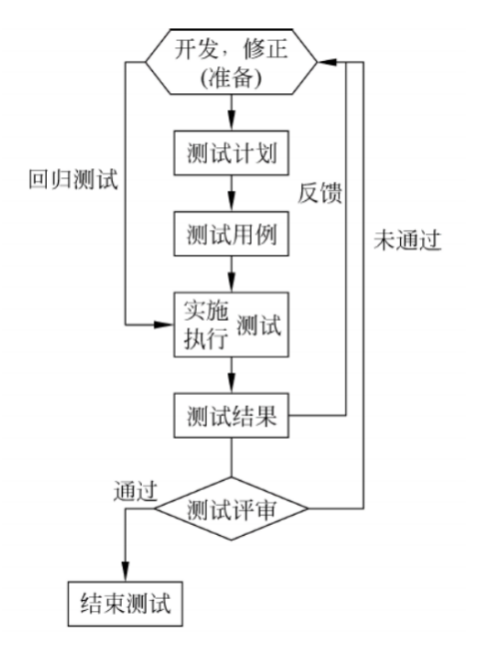
* 测试人员按照测试用例逐项进行测试活动，并且将测试结果填写在测试报告上。
* 测试过程中发现Bug，将Bug填写在PingCode（缺陷跟踪工具）上发给集成部经理（Bug状态为NEW）。
* 对应责任人接到BugHerd的Bug信息。
* 对于明显的并且可以立刻解决的Bug，将Bug发给开发人员（Bug状态为ASSIGNED）。对于目前无法解决的，将这个Bug放到下一轮次进行修改。（Bug状态为RESOLVED）。

1. 问题反馈：反馈Bug给开发人员。

* 开发人员接到发过来的Bug立刻修改。
* 测试人员接到BugHerd（缺陷跟踪工具）通过Email发过来的Bug更改信息，应该逐项复测，填写新的测试报告。

1. 回归测试：重新测试修复Bug后的系统。重复3，直到4回归测试结果达到系统验收标准。
2. 集成测试总结报告：完成以上4步后，综合相关资料生成报告。
3. 进入系统测试。

整体集成测试过程活动流程图如下。



**3.2 子系统集成测试**

完成子系统间的集成。

**4. 测试通过/失败标准**

**4.1 模块验收标准**

接口：接口提供的功能或者数据正确。

功能点：验证程序与产品描述、用户文档中的全部说明相对应，一致性。

流程处理：验证程序与产品描述、用户文档中的全部说明相对应，一致性。

外部接口：验证程序与产品描述、用户文档中的全部说明相对应，一致性。

**4.2 集成测试验收标准**

首先，集成测试用例中所设计的功能测试用例必须全部通过，性能及其他类型测试用例通过 95%以上。 在未通过的测试用例中，不能含有 “系统崩溃“和“严重错误“错误，”一般错误“小于 1%。 测试结果与 测试用例中期望的结果一致，测试通过，否则表明测试未通过。

**5. 工具**

**5.1 测试工具**

Junit、Cypress

**5.2 其它工具**

Github

**6. 挂起，恢复条件**

**6.1 挂起**

集成测试挂起条件是指在集成测试过程中，由于某些特定条件未满足或存在客观限制，导致测试活动需要暂时中止或延期的情况。常见的挂起条件包括：

1. 优先级调整  
   当出现更高优先级的测试任务时，测试团队需暂停当前集成测试工作，优先处理紧急任务，以确保关键项目需求的及时满足。
2. 资源不足  
   若测试过程中出现人力资源或测试设备短缺，导致测试进度受阻或效率显著降低，测试团队应及时与项目管理方协调，通过资源调配或计划调整来保障测试工作的顺利开展。
3. 缺陷数量超标  
   在首轮集成测试阶段，若测试人员发现单个功能模块存在三个及以上缺陷（包括功能性缺陷和操作性问题），应中止集成测试并将该模块退回单元测试组进行修复和复测。
4. 突发紧急事件  
   当发生系统级重大故障（如系统崩溃、严重安全漏洞等）时，测试团队须立即暂停测试工作，协同相关技术团队进行紧急处置，待系统恢复稳定后再继续测试。

**6.2 恢复**

1. **准入条件满足**  
   待测模块或系统必须符合集成测试的准入标准，包括通过必要的单元测试、接口验证等前置条件。
2. **高优先级任务完成**  
   若因更高优先级的任务导致测试暂停，则需在该任务完成后，重新评估并恢复集成测试。
3. **复测通过且产品可运行**  
   若因缺陷问题退回修复，需在单元测试组完成复测并确认产品功能稳定、可正常执行后，方可重新进入集成测试。
4. **测试资源就绪**  
   人员、测试环境、设备等关键资源必须到位，确保测试能够顺利执行。
5. **突发事件已解决**  
   若因系统崩溃、安全漏洞等紧急情况暂停测试，需在问题完全修复且系统恢复稳定后，方可继续测试。

**7. 责任人**

* 测试负责人：柳阳，商一诚

控制并完成测试任务与测试过程，决定测试人员提交上的Bug是否需要修改。

* 测试设计人员：夏济舟

设计集成测试用例。

* 测试人员：柳阳，商一诚

按照用例进行测试活动。

* 开发人员：韩坤甫

程序Bug修改，程序员间协调。

* 用户代表：无

**8. 集成测试用例与报告**

**8.1 集成测试接口分析**

**MessageService模块接口**

|  |  |  |  |
| --- | --- | --- | --- |
| 标识符 | 名称 | 调用层次数 | 调用函数次数 |
| TC\_MS\_001 | public List<MessageUserDTO> getChatHistory(Integer chatId, Integer userId) | 1 | 1 |
| TC\_MS\_002 | public ResultMsg deleteMsg(Integer userId,Integer messageId) | 1 | 1 |

**UserService模块集成接口**

|  |  |  |  |
| --- | --- | --- | --- |
| 标识符 | 名称 | 调用层次数 | 调用函数次数 |
| TC\_US\_001 | public LoginResponseDTO login(String userName, String password) | 1 | 1 |
| TC\_US\_002 | public UserInfoDetailDTO getUserInfo(int userId) |  |  |

**MessageService模块和ChatService模块集成接口**

|  |  |  |  |
| --- | --- | --- | --- |
| 标识符 | 名称 | 调用层次数 | 调用函数次数 |
| TC\_MC\_001 | public Message sendMessage(MessageDTO messageDto, Integer userId) | 1 | 1 |

**GroupMemberService模块和ChatService模块集成接口**

|  |  |  |  |
| --- | --- | --- | --- |
| 标识符 | 名称 | 调用层次数 | 调用函数次数 |
| TC\_GC\_002 | public ResultMsg quitGroup(Integer groupId,Integer memberId) | 1 | 1 |

**FriendApplicationService模块和UserService模块集成接口**

|  |  |  |  |
| --- | --- | --- | --- |
| 标识符 | 名称 | 调用层次数 | 调用函数次数 |
| TC\_FU\_001 | public List<ApplicationResponseDTO> getAllFriendApplication(Integer userId) | 1 | 1 |

**8.2 用例分析与设计**

该部分主要集中在对前后端接口进行集成测试，我们选择了具有代表性的接口进行测试。为了方便进行接口测试，我们使用 Cypress作为测试工具。通过Cypress，我们可以使用脚本模拟前端操作并进行测试，以确保接口的功能和正确性。集成测试是对前后端接口的完整性和交互的验证。

### 8.1.1.1 / getChatHistory

**设计标识符**：TC\_SS\_001  
**被测特性**：

1. **聊天记录获取**：成员可获取完整聊天历史（含时间倒序分页）
2. **权限控制**：非群成员禁止访问聊天记录
3. **数据完整性**：返回消息包含发送者基本信息（头像、昵称等）
4. **敏感信息过滤**：自动过滤已撤回/删除的消息（可选扩展）

**测试方法**：  
采用 **等价类划分** 和 **分页边界测试**：

* **有效输入**：
  + 用户是群成员 + 合法chatId + 合理分页参数
* **无效输入**：
  + 非群成员访问/不存在的chatId/越界分页

### **测试项标识**

| 测试项标识符 | 测试项描述 | 优先级 |
| --- | --- | --- |
| TC\_SS\_001\_001 | 群成员获取聊天历史 | 高 |
| TC\_SS\_001\_002 | 非群成员访问历史 | 高 |
|  |  |  |

### **测试用例**

#### **TC\_SS\_001\_001**

**测试项描述**：成功获取聊天历史  
**优先级**：高  
**前置条件**：

* 群聊ID=1中有2条历史消息
* 用户ID=1是群成员

| 输入 | 预期结果 | 数据结构验证 |
| --- | --- | --- |
| chatId=1, userId=1 | 返回{data: [{id:1, content:"Hi", sender:{id:1, name:"Alice", avatar:"url"}}, ...], total:2} | 每条消息必须包含sender对象 |

#### **TC\_SS\_001\_002**

**测试项描述**：非群成员访问拒绝  
**优先级**：高  
**前置条件**：用户ID=1不在群聊ID=1的成员列表中

| 输入 | 预期结果 | 安全日志 |
| --- | --- | --- |
| chatId=1, userId=1 | 返回{code:403, msg:"无访问权限"} | 记录非法访问尝试 |

### 8.1.1.2 / deleteMsg

**设计标识符**：TC\_SS\_002  
**被测特性**：

1. **成功删除消息**：当用户有权限删除消息时，正确返回成功状态。
2. **删除失败**：当用户无权限或消息不存在时，抛出异常（如"撤回该信息失败"）。
3. **参数异常**：当userId或messageId无效时，抛出参数校验异常。

**测试方法**：  
本测试采用 **等价类划分** 和 **异常场景测试** 方法设计测试用例。

* **等价类划分**：
  + userId：空值、不存在、存在（有效等价类）
  + messageId：空值、不存在、存在（有效等价类）
* **异常场景**：
  + 重复删除同一条消息
  + 非消息创建者尝试删除

### **测试项标识**

| 测试项标识符 | 测试项描述 | 优先级 |
| --- | --- | --- |
| TC\_SS\_002\_001 | userId和messageId有效，且用户有删除权限 | 中 |
| TC\_SS\_002\_002 | userId或messageId无效 | 高 |

### **测试用例**

#### **TC\_SS\_002\_001**

**测试项描述**：成功删除消息（用户有权限）  
**优先级**：中  
**前置条件**：消息存在且用户为消息创建者

| 用例序号 | 输入 | 期望结果 |
| --- | --- | --- |
| 1 | userId=1, messageId=1 | 返回{status: 1, msg: "操作成功"} |

#### **TC\_SS\_002\_002**

**测试项描述**：删除消息失败（无权限或消息无效）  
**优先级**：高  
**前置条件**：用户已登录

| 用例序号 | 输入 | 期望结果 |
| --- | --- | --- |
| 1 | userId=1, messageId=999 | 抛出异常，提示"撤回该信息失败" |
| 2 | userId=null, messageId=1 | 抛出参数校验异常（如"用户ID不能为空"） |

### 8.1.1.3 / login

**设计标识符**：TC\_SS\_003  
**被测特性**：

1. **成功登录**：当用户名和密码正确时，返回用户信息及有效Token。
2. **用户不存在**：当用户名未注册时，提示"用户不存在"。
3. **密码错误**：当密码不匹配时，提示"密码错误"。
4. **参数校验**：当用户名或密码为空时，抛出参数异常。

**测试方法**：  
采用 **等价类划分** 和 **安全性测试** 方法设计用例：

* **有效等价类**：
  + username：已注册用户（格式合规）
  + password：与用户名匹配的密码
* **无效等价类**：
  + username：空值、未注册用户、特殊字符注入
  + password：空值、错误密码、长度超限

### **测试项标识**

| 测试项标识符 | 测试项描述 | 优先级 |
| --- | --- | --- |
| TC\_SS\_003\_001 | 用户名和密码正确 | 高 |
| TC\_SS\_003\_002 | 用户名不存在 | 中 |
| TC\_SS\_003\_003 | 密码错误 | 高 |
|  |  |  |

### **测试用例**

#### **TC\_SS\_003\_001**

**测试项描述**：成功登录  
**优先级**：高  
**前置条件**：用户已注册且密码已存储

| 用例序号 | 输入 | 期望结果 |
| --- | --- | --- |
| 1 | username="test", password="password" | 返回{user: {id:1,name:"test"}, token: "xxx"}，状态码200 |

#### **TC\_SS\_003\_002**

**测试项描述**：用户不存在  
**优先级**：中  
**前置条件**：数据库无用户nonexistent

| 用例序号 | 输入 | 期望结果 |
| --- | --- | --- |
| 1 | username="nonexistent", password="xxx" | 返回状态码401，提示"用户不存在" |

#### **TC\_SS\_003\_003**

**测试项描述**：密码错误  
**优先级**：高  
**前置条件**：用户test已存在

| 用例序号 | 输入 | 期望结果 |
| --- | --- | --- |
| 1 | username="test", password="wrong" | 返回状态码401，提示"密码错误" |

### 8.1.1.4 / getUserInfo

**设计标识符**：TC\_SS\_004  
**被测特性**：

1. **成功获取用户信息**：当用户ID存在时，返回完整的用户信息（含敏感信息脱敏）。
2. **用户不存在**：当用户ID未注册时，提示"用户不存在"。
3. **参数校验**：当用户ID为空或非法格式时，抛出参数异常。
4. **权限控制**：非公开用户信息需验证访问权限（可选扩展测试项）。

**测试方法**：  
采用 **等价类划分** 和 **边界值分析** 方法：

* **有效等价类**：
  + userId：数据库中存在的有效ID（1≤id≤10000）
* **无效等价类**：
  + userId：空值、非数字字符、负数、超出数据库最大ID

### **测试项标识**

| 测试项标识符 | 测试项描述 | 优先级 |
| --- | --- | --- |
| TC\_SS\_004\_001 | 用户ID存在且有效 | 高 |
| TC\_SS\_004\_002 | 用户ID不存在 | 中 |
|  |  |  |

### **测试用例**

#### **TC\_SS\_004\_001**

**测试项描述**：成功获取用户信息  
**优先级**：高  
**前置条件**：数据库中存在ID=1的用户

| 用例序号 | 输入 | 期望结果 |
| --- | --- | --- |
| 1 | userId=1 | 返回{id:1, name:"用户1", email:"u\*\*\*@example.com", (其他脱敏字段)}，状态码200 |

#### **TC\_SS\_004\_002**

**测试项描述**：用户不存在  
**优先级**：中  
**前置条件**：数据库最大用户ID<999

| 用例序号 | 输入 | 期望结果 |
| --- | --- | --- |
| 1 | userId=999 | 返回状态码404，提示"用户不存在" |

### 8.1.1.5 / sendMessage

**设计标识符**：TC\_SS\_005  
**被测特性**：

1. **成功发送消息**：当用户是群聊成员且参数有效时，消息持久化并返回完整Message对象
2. **权限验证失败**：当用户非群聊成员时，拒绝发送并提示
3. **参数校验**：对userId/chatId/content的格式和必填校验
4. **内容安全**：敏感词过滤和消息长度限制（可选扩展测试）

**测试方法**：  
采用 **场景分析法** 和 **边界值测试**：

* **有效场景**：
  + 用户是群成员 + 消息内容合规（1≤长度≤1000字符）
* **无效场景**：
  + 用户非群成员/聊天不存在/内容为空/长度超限

### **测试项标识**

| 测试项标识符 | 测试项描述 | 优先级 |
| --- | --- | --- |
| TC\_SS\_005\_001 | 合法用户发送合规消息 | 高 |
| TC\_SS\_005\_002 | 非群成员发送消息 | 中 |
|  |  |  |

### **测试用例**

#### **TC\_SS\_005\_001**

**测试项描述**：成功发送消息  
**优先级**：高  
**前置条件**：

* 用户1是群聊1的成员
* 群聊1状态为活跃

| 用例序号 | 输入 | 期望结果 |
| --- | --- | --- |
| 1 | userId=1, chatId=1, content="Hello" | 返回{id:123, senderId:1, chatId:1, content:"Hello", sendTime:"2024-03-20T10:00:00"}，状态码200 |

#### **TC\_SS\_005\_002**

**测试项描述**：非群成员发送消息  
**优先级**：中  
**前置条件**：用户1不在群聊1的成员列表中

| 用例序号 | 输入 | 期望结果 |
| --- | --- | --- |
| 1 | userId=1, chatId=1, content="Test" | 返回状态码403，提示"该用户并非该群聊的成员" |

### 8.1.1.6 / quitGroup

**设计标识符**：TC\_SS\_006  
**被测特性**：

1. **成功退群**：当用户是群成员时，正确移除成员关系并返回成功提示
2. **非成员退群**：当用户未加入群组时，返回明确错误提示
3. **参数校验**：对groupId/memberId的格式和必填校验
4. **群组状态检查**：已解散的群组不允许退群（可选扩展测试）

**测试方法**：  
采用 **状态转换测试** 和 **异常流测试**：

* **有效场景**：
  + 用户是活跃群组成员 → 退群后成员列表更新
* **无效场景**：
  + 用户非群成员/群组不存在/参数非法

### **测试项标识**

| 测试项标识符 | 测试项描述 | 优先级 |
| --- | --- | --- |
| TC\_SS\_006\_001 | 群成员正常退群 | 高 |
| TC\_SS\_006\_002 | 非群成员尝试退群 | 中 |
|  |  |  |

### **测试用例**

#### **TC\_SS\_006\_001**

**测试项描述**：群成员成功退群  
**优先级**：高  
**前置条件**：

* 群组1状态为活跃
* 用户1是群组1的成员

| 用例序号 | 输入 | 期望结果 |
| --- | --- | --- |
| 1 | groupId=1, memberId=1 | 返回{code:200, msg:"退出团体成功"}，且数据库members表移除对应记录 |

#### **TC\_SS\_006\_002**

**测试项描述**：非群成员退群  
**优先级**：中  
**前置条件**：用户999未加入任何群组

| 用例序号 | 输入 | 期望结果 |
| --- | --- | --- |
| 1 | groupId=1, memberId=999 | 返回{code:403, msg:"该用户没有加入团体"} |

### 8.1.1.7 / getAllFriendApplication

**设计标识符**：TC\_SS\_007  
**被测特性**：

1. **正确获取好友申请列表**：返回当前用户的所有好友申请记录
2. **空列表处理**：当无申请时返回空数组而非null
3. **字段映射完整性**：确保DTO包含所有必需字段且数据正确
4. **服务调用验证**：正确调用依赖服务（如UserService）

**测试方法**：  
采用 **数据驱动测试** 和 **接口契约测试**：

* **数据场景**：
  + 0条/1条/多条申请记录
  + 不同申请状态（待处理/已通过/已拒绝）
* **契约验证**：
  + 字段类型和结构符合API文档
  + 服务调用次数和参数正确

### **测试项标识**

| 测试项标识符 | 测试项描述 | 优先级 |
| --- | --- | --- |
| TC\_SS\_007\_001 | 获取含多条申请记录的列表 | 高 |
| TC\_SS\_007\_002 | 无申请记录时返回空列表 | 中 |
| TC\_SS\_007\_003 | DTO字段完整性和映射验证 | 高 |
| TC\_SS\_007\_004 | 依赖服务调用验证 | 低 |

### **测试用例**

#### **TC\_SS\_007\_001**

**测试项描述**：获取多条好友申请记录  
**优先级**：高  
**前置条件**：

* 用户1存在2条待处理的好友申请

| 用例序号 | 输入 | 期望结果 |
| --- | --- | --- |
| 1 | userId=1 | 返回[{id:101, fromUserId:2, toUserId:1, status:"PENDING", createTime:"2024-01-01T00:00:00"}, {...}]，数组长度=2 |

#### **TC\_SS\_007\_002**

**测试项描述**：无申请记录时返回空列表  
**优先级**：中  
**前置条件**：用户1无任何好友申请

| 用例序号 | 输入 | 期望结果 |
| --- | --- | --- |
| 1 | userId=1 | 返回[]（空数组） |

#### **TC\_SS\_007\_003**

**测试项描述**：DTO字段映射验证  
**优先级**：高  
**测试数据**：

{  
 "id": 101,  
 "fromUserId": 2,  
 "toUserId": 1,  
 "status": "PENDING",  
 "createTime": "2024-01-01T00:00:00",  
 "remark": "求通过"  
}

| 验证字段 | 预期值 | 实际值 | 通过 |
| --- | --- | --- | --- |
| id | 101 | - | □ |
| fromUserId | 2 | - | □ |
| status | "PENDING" | - | □ |
| createTime | ISO8601格式 | - | □ |

#### **TC\_SS\_007\_004**

**测试项描述**：UserService调用验证  
**优先级**：低  
**前置条件**：存在2条来自不同用户的好友申请

| 用例序号 | 输入 | 期望验证点 |
| --- | --- | --- |
| 1 | userId=1 | 验证调用UserService.getUserProfile(2)和UserService.getUserProfile(3)各1次 |

**附件：测试报告**

**结果分析**

**覆盖率分析表**

|  |  |  |
| --- | --- | --- |
| 标识符 | 名称 | 覆盖率 |
| TC\_MS\_001 | public List<MessageUserDTO> getChatHistory(Integer chatId, Integer userId) | 100% |
| TC\_MS\_002 | public ResultMsg deleteMsg(Integer userId,Integer messageId) | 100% |
| TC\_US\_001 | public LoginResponseDTO login(String userName, String password) | 100% |
| TC\_US\_002 | public UserInfoDetailDTO getUserInfo(int userId) | 100% |
| TC\_MC\_001 | public Message sendMessage(MessageDTO messageDto, Integer userId) | 100% |
| TC\_GC\_002 | public ResultMsg quitGroup(Integer groupId,Integer memberId) | 100% |
| TC\_FU\_001 | public List<ApplicationResponseDTO> getAllFriendApplication(Integer userId) | 100% |