**专业方向综合项目 测试报告**

**张尧 陈东 姜其升**

**TEam 12**

**MEETING！**

基于微信小程序的协同日程安排应用

**目录**

[1.引言 2](#_Toc44672653)

[1.1编写目的 2](#_Toc44672654)

[1.2项目背景 2](#_Toc44672655)

[1.3 定义 2](#_Toc44672656)

[2. 测试概要 6](#_Toc44672657)

[2.1测试时间 6](#_Toc44672658)

[2.2 测试范围 6](#_Toc44672659)

[2.2.1功能测试 6](#_Toc44672660)

[2.2.2接口测试 7](#_Toc44672661)

[2.2.3兼容性测试 7](#_Toc44672662)

[2.2.4安全性测试 7](#_Toc44672663)

[2.3测试用例 7](#_Toc44672664)

[2.4测试策略 14](#_Toc44672665)

[3.结果分析 15](#_Toc44672666)

[3.1 缺陷趋势 15](#_Toc44672667)

[3.2 缺陷优先级分布 15](#_Toc44672668)

[3.3 缺陷按模块分布 16](#_Toc44672669)

[3.4 重开缺陷情况 17](#_Toc44672670)

[3.5 遗留缺陷情况 17](#_Toc44672671)

[3.6 上线跟踪测试结果 17](#_Toc44672672)

[4.总结 18](#_Toc44672673)

[4.1 测试结论 18](#_Toc44672674)

[4.2 呈现的问题 18](#_Toc44672675)

[4.3 测试建议 18](#_Toc44672676)

# 1.引言

## 1.1编写目的

本文档用于记录测试过程，总结各轮次的测试情况，分析测试数据，归纳测试工作进行过程中暴露的问题与遗留的风险，给出相应的测试建议以供后续项目参考。

## 1.2项目背景

在众多需要多人参与的任务中，例如会议安排、团队协作、各种约会等等，在这些事件中我们可能都亲身体验过这样的一个痛点，事件参与成员的空闲时间不统一，就难以为一件共同事件安排计划。作为大学生的我们更能体会到这一点。我们每个人在课外都有着多多少少的项目，每个人上课时间不同，每个项目的团队成员不同，大家都有着各自的计划，而且很多时候项目都是同等重要的，所以有效的安排团队事件将很大程度上提升我们的工作效率，增加团队默契。

如今市面上存在着各式各样的日程管理软件，且几乎都为独立app的形式。其中大多数日程管理软件都只是专注与个人日程的管理，而忽略了团队事件与个人日程之间的冲突。当然也中不少日程管理应用功能全面，但从另一个角度来说这也增加了软件使用的复杂程度。而多数用户都会因为功能复杂和难以上手的原因放弃使用这样的日程管理应用。此外在团队事件都存在这样的一个特点，团队在一定的时间内是相对固定，而且多数的团队都会拥有各自的交流空间，而这其中大多数的都会选择微信群。所以这里存在着几个机会，Meeting系统在保留最基础的个人日程管理功能的基础上，增加对与团队事件的管理，对功做加法和减法，尽可能保证功能的必要性和使用的简单性。同时通过微信小程平台呈现将避免用户下载独立应用，降低产品接触目标用户的难度，与微信群之间建立良好互动。

meeting需要一个拥有真实用户的社区化产品，通过真实高信任度用户关系的建立，提高用户粘性，提升活跃会员数，带来长效的增长。在此背景下，以真实用户为基础的社区应运而生。主要具有以下5点意义：

1. 提高社区活跃会员数

2. 提高用户粘度

3. 建立真实（和用户的社区身份相一致）的多维用户信息

4. 建立高信任度的用户关系

5. 达到真实可信用户关系中的用户之间的传播效应

## 1.3 定义

**Redis**

Redis是一个开源的使用 ANSI C语言编写，支持网络，可基于内存也可持久化的日志型，Key-Value数据库，并提供了多种语言的 API ,相比 Memcached 它支持存储的类型相对更多 (字符，哈希，集合，有序集合，列表等)，同时Redis是线程安全的。

客户端连接 Redis 使用的是 TCP协议，直连的方式每次需要建立 TCP连接，而连接池的方式是可以预先初始化好客户端连接，所以每次只需要从 连接池借用即可，而借用和归还操作是在本地进行的，只有少量的并发同步开销，远远小于新建TCP连接的开销。另外，直连的方式无法限制 redis客户端对象的个数，在极端情况下可能会造成连接泄漏，而连接池的形式可以有效的保护和控制资源的使用。

Jedis 在实现上是直连 redis server，多线程环境下非线程安全，除非使用连接池，为每个 redis实例增加 物理连接。

我们的项目使用Redis来支持微信用户登录服务相关。当新用户登录时，后端会向微信发送请求以获取该用户的“openid”、“session\_key”和后端将分配一个新的“userid”，这是我们在关系数据库中对用户的主键。这三个条目都存储在Redis作为后端uuid生成的密钥的值。下次用户登录时，微信小程序会向微信发送请求，查看其会话密钥是否过期。如果不是，小程序告诉后端使用现有的条目。然后小程序通信从上一次使用uuid的后端，后端将检查相应的“userid”告诉哪个用户在线。

**MyBatis**

MyBatis 是支持普通 SQL查询，存储过程和高级映射的优秀持久层框架。MyBatis 消除了几乎所有的JDBC代码和参数的手工设置以及结果集的检索。MyBatis 使用简单的 XML或注解用于配置和原始映射，将接口和 Java 的POJOs（Plain Ordinary Java Objects，普通的 Java对象）映射成数据库中的记录。

MyBatis是半自动的ORM框架，原来叫ibatis， 10年apache软件基金组织把它托管给了goole code，就重新命名了MyBatis，功能相对以前更强大了。它相对全自动的持久层框架Hibernate，更加灵活，更轻量级，这点我还是深有体会的。MyBatis的一个强大特性之一就是动态SQL能力了，能省去我们很多串联判断拼接SQL的痛苦，根据项目而定，在一定的场合下使用，能大大减少程序的代码量和复杂程度，不过还是不是过度太过复杂的使用，以免不利于后期的维护和扩展。

**HikariCP连接池**

HikariCP是由日本程序员开源的一个数据库连接池组件，代码非常轻量，并且速度非常的快。根据官方提供的数据，在i7,开启32个线程32个连接的情况下，进行随机数据库读写操作，HikariCP的速度是现在常用的C3P0数据库连接池的数百倍。在SpringBoot2.0中，官方也是推荐使用HikariCP。优势是

1.字节码更加精简，所以可以加载更多代码到缓存。

2.实现了一个无锁的集合类型，来减少并发造成的资源竞争。

3.使用了自定义的数组类型，相对与ArrayList极大地提升了性能。

4.针对CPU的时间片算法进行优化，尽可能在一个时间片里面完成各种操作。

**Lettuce**

Lettuce是一个高性能基于Java编写的Redis驱动框架，底层集成了Project Reactor提供天然的反应式编程，是 一种可伸缩，线程安全，完全非阻塞的Redis客户端，多个线程可以共享一个RedisConnection,它利用Netty NIO 框架来高效地管理多个连接，从而提供了异步和同步数据访问方式，用于构建非阻塞的反应性应用程序。通信框架集成了Netty使用了非阻塞IO，5.x版本之后融合了JDK1.8的异步编程特性，在保证高性能的同时提供了十分丰富易用的API，

支持Redis的新增命令ZPOPMIN, ZPOPMAX, BZPOPMIN, BZPOPMAX。

支持通过Brave模块跟踪Redis命令执行。

支持Redis Streams。

支持异步的主从连接。

支持异步连接池。

新增命令最多执行一次模式（禁止自动重连）。

全局命令超时设置（对异步和反应式命令也有效）。

**Apache Shiro**

Apache Shiro 是一个功能强大、灵活的，开源的安全框架。它可以干净利落地处理身份验证、授权、企业会话管理和加密。

实际上，它实现了管理应用程序安全性的所有方面，同时尽可能避免出现问题。它建立在完善的接口驱动设计和面向对象的原则之上，可以在任何你想象得到的地方实现自定义行为。但是，对于所有事情来说，默认情况下都是合理的，这与应用程序安全性是一样的。至少这是我们所追求的。

我们 POST 用户名与密码到 /login 进行登入，如果成功返回一个sessionKey，失败的话直接返回 401 错误。

之后用户访问每一个需要权限的网址请求必须在 header 中添加 Authorization 字段，例如 Authorization: token ，token 为密钥。

后台会进行 token 的校验，如果有误会直接返回 401。

**微信小程序**

微信小程序是一种不用下载就能使用的应用，也是一项创新，经过将近两年的发展，已经构造了新的微信小程序开发环境和开发者生态。微信小程序也是这么多年来中国IT行业里一个真正能够影响到普通程序员的创新成果，已经有超过150万的开发者加入到了微信小程序的开发，与我们一起共同发力推动微信小程序的发展，微信小程序应用数量超过了一百万，覆盖200多个细分的行业，日活用户达到两个亿，微信小程序还在许多城市实现了支持地铁、公交服务。微信小程序发展带来更多的就业机会，2017年小程序带动就业104万人，社会效应不断提升

本项目采用微信小程序的客户端形式，充分利用跨平台一次开发的遍历，也是让我们在同类产品中具有更多可得性的地方。

**HyperText Transfer Protocol Secure (HTTPS)**

HTTPS （全称：Hyper Text Transfer Protocol over SecureSocket Layer），是以安全为目标的 HTTP 通道，在HTTP的基础上通过传输加密和身份认证保证了传输过程的安全性。HTTPS 在HTTP 的基础下加入SSL 层，HTTPS 的安全基础是 SSL，因此加密的详细内容就需要 SSL。 HTTPS 存在不同于 HTTP 的默认端口及一个加密/身份验证层（在 HTTP与 TCP 之间）。这个系统提供了身份验证与加密通讯方法。它被广泛用于万维网上安全敏感的通讯，例如交易支付等方面。

HTTP 协议虽然使用极为广泛, 但是却存在不小的安全缺陷, 主要是其数据的明文传送和消息完整性检测的缺乏, 而这两点恰好是网络支付, 网络交易等新兴应用中安全方面最需要关注的。

关于 HTTP协议的明文数据传输, 攻击者最常用的攻击手法就是网络嗅探, 试图从传输过程当中分析出敏感的数据, 例如管理员对 Web 程序后台的登录过程等等, 从而获取网站管理权限, 进而渗透到整个服务器的权限。即使无法获取到后台登录信息, 攻击者也可以从网络中获取普通用户的隐秘信息, 包括手机号码, 身份证号码, 信用卡号等重要资料, 导致严重的安全事故。进行网络嗅探攻击非常简单, 对攻击者的要求很低。使用网络发布的任意一款抓包工具, 一个新手就有可能获取到大型网站的用户信息。

另外,HTTP协议在传输客户端请求和服务端响应时, 唯一的数据完整性检验就是在报文头部包含了本次传输数据的长度, 而对内容是否被篡改不作确认。 因此攻击者可以轻易的发动中间人攻击, 修改客户端和服务端传输的数据, 甚至在传输数据中插入恶意代码, 导致客户端被引导至恶意网站被植入木马。

#### 1.4参考资料

各轮测试阶段总结

# 2. 测试概要

整个项目的测试经历了meeting-1.0一个阶段，共经历了1轮集成测试、1轮系统测试和1轮上线跟踪测试。整个测试过程中累计执行用例11条，发现缺陷1个。截至1.0系统测试结束，所发现的高权重问题已得到修复和验证。

## 2.1测试时间

整个meeting项目的测试时间从2020年6月26日开始，到2020年6月30日上线止，期间各阶段工作情况如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 工作阶段 | 开始时间 | 结束时间 | 工作量(人日) |
| 1.0 | 需求确认、评审、测试用例编写&评审 | 2020年6月27日 | 6 |
| 1.0第一轮系统测试 | 2020年6月28日 | 2020年6月30日 | 9 |
| 1.0上线跟踪测试 | 2020年7月1日 | 2020年7月1日 | 3 |
| 合计 |  |  | 18 |

## 2.2 测试范围

本次测试覆盖的范围包括：功能测试、兼容性测试、接口测试、数据迁移测试、性能测试、安全性测试和品质监控。以下分别对功能测试、兼容性测试、接口测试、数据迁移测试、性能测试和安全性测试进行说明。

### 2.2.1功能测试

1.0 包括的主要功能如下：

|  |  |  |
| --- | --- | --- |
| No. | 模块 | 权重 |
| 1 | 显示用户日历 | A |
| 2 | 在日历中添加事件 | A |
| 3 | 修改和删除个人日历中的事件 | A |
| 4 | 事件提醒 | C |
| 5 | 创建群组和群组日历 | A |
| 6 | 参加一个小组 | A |
| 7 | 退出群组（仅限群组中的成员） | A |
| 8 | 解散一个组并删除组日历（只有组日历的所有者） | B |
| 9 | 通过微信小程序链接分享群，邀请他人加入群 | B |
| 10 | 添加组事件（仅限组日历的所有者） | A |
| 11 | 修改和删除组事件（仅限于组日历的所有者） | A |
| 12 | 查看团队成员的个人日历 | C |
| 13 | 增加团体活动时的空闲时间推荐 | A |
| 14 | 更改组 | B |
| 15 | 请勿打扰模式 | C |
| 16 | 隐私模式 | C |
| 17 | 修改用户名 | C |

### 2.2.2接口测试

|  |  |  |
| --- | --- | --- |
| No. | 关注项 | 权重 |
| 1 | 前后端接口 | A |

### 2.2.3兼容性测试

|  |  |  |
| --- | --- | --- |
| No. | 关注项 | 权重 |
| 1 | Android手机 | A |
| 2 | IOS手机 | A |

### 2.2.4安全性测试

整个测试过程中先后进行了两轮安全性测试，发现了2个影响较严重的安全性问题，且都已得到修复和验证。

## 2.3测试用例

根据需求文档，测试人员编写和内审了测试用例。

|  |  |
| --- | --- |
| 用例名称 | 创建个人事件 |
| 标识符 | UC01 |
| 主要参与者 | 小组成员或组长，两者在创建个人事件的时候没有差别 |
| 说明 | 用户在不产生时间冲突的前提下在日程表上创建一个个人事件 |
| 前置条件 | 1. 用户已经注册 2. 用户处在良好的网络环境中 |
| 后置条件 | 1、用户输入的所有信息被接收，该事件被添加到个人日程中 |
| 被包含的用例 | 设置事件详细信息 |
| 触发器 | 用户点击添加个人事件 |
| 基本操作流程 | 1. 设置事件详细信息   1.1、输入事件的日期、开始时间和结束时间  1.2、输入事件的优先级  1.3、选择事件是否重复及是否添加事件详细描述  2、提交事件信息 |
| 可选操作流程 | 1. 事件名称为空：弹出提示，用户补充事件名称 2. 事件时间段冲突：弹出提示，用户重新输入事件的开始时间和结束时间 3. 未指定时间优先级：弹出提示，用户补充事件优先级 |
| 优先级 | 必须实现 |
| 使用频率 | 每天多次 |
| 使用方式 | 通过日程表界面接口 |

|  |  |
| --- | --- |
| 用例名称 | 修改个人事件 |
| 标识符 | UC02 |
| 主要参与者 | 小组成员或组长，两者在修改个人事件的时候没有差别 |
| 说明 | 用户选择一个事件并对其基本信息进行修改或删除 |
| 前置条件 | 1. 用户已经注册 2. 用户处在良好的网络环境中 3. 用户的日程表上至少存在一个个人事件 |
| 后置条件 | 1、用户输入的修改要求被同步更新到日程表中 |
| 被扩展的用例 | 删除个人事件 |
| 触发器 | 用户长按某个个人事件 |
| 基本操作流程 | 1. 输入更新后事件的名称 2. 输入更新后事件的优先级   3、输入更新后事件的重复信息和详细描述  4、提交修改信息 |
| 可选操作流程 | 1. 用户选择删除事件：弹出提示，将该事件从用户日程表中移除   2、修改后事件名为空：弹出提示，用户重新输入事件的名称 |
| 优先级 | 必须实现 |
| 使用频率 | 每天多次 |
| 使用方式 | 通过日程表界面接口 |

|  |  |
| --- | --- |
| 用例名称 | 创建团队事件 |
| 标识符 | UC03 |
| 主要参与者 | 组长，只有组长拥有创建团队事件的权力 |
| 说明 | 组长在不产生时间冲突的情况下创建一个需要整个团队参与的事件 |
| 前置条件 | 1. 用户已经注册 2. 用户处在良好的网络环境中   3、用户在当前组中的身份是组长 |
| 后置条件 | 1、用户输入团队事件被添加到所有组员的日程表当中 |
| 被包含的用例 | 设置事件详细信息 |
| 被扩展的用例 | 时间段推荐 |
| 触发器 | 用户点击添加组事件按钮 |
| 基本操作流程 | 1、设置事件详细信息  1.1、输入事件的名称、日期、开始时间和结束时间  1.2、输入事件的优先级  1.3、选择事件是否重复及是否添加事件详细描述  2、提交事件信息 |
| 可选操作流程 | 1. 用户选择时间段推荐    1. 用户输入想要添加的团队事件长度    2. 系统返回三个不冲突的事件段    3. 用户选择其中之一，系统跳转到添加界面并自动填充好时间 2. 事件名称为空：弹出提示，用户补充事件名称 3. 事件时间段冲突：弹出提示，用户重新输入事件的开始时间和结束时间   4、未指定时间优先级：弹出提示，用户补充事件优先级 |
|  | 必须实现 |
| 使用频率 | 取决于团队事件密度 |
| 使用方式 | 通过团队日程表界面接口 |

|  |  |
| --- | --- |
| 用例名称 | 修改团队事件 |
| 标识符 | UC04 |
| 主要参与者 | 组长，只有组长拥有修改团队事件的权力 |
| 说明 | 组长在不产生时间冲突的情况下修改一个需要整个团队参与的事件 |
| 前置条件 | 1. 用户已经注册 2. 用户处在良好的网络环境中 3. 用户在当前组中的身份是组长 4. 该组至少存在一个团队事件 |
| 后置条件 | 1、用户的修改要求被同步到所有组员的日程表当中 |
| 被扩展的用例 | 删除团队事件 |
| 触发器 | 用户长按某个团队事件 |
| 基本操作流程 | 1. 输入更新后事件的名称 2. 输入更新后事件的优先级   3、输入更新后事件的重复信息和详细描述  4、提交修改信息 |
| 可选操作流程 | 1. 用户选择删除事件：弹出提示，将该事件从用户日程表中移除   2、修改后事件名为空：弹出提示，用户重新输入事件的名称 |
| 优先级 | 必须实现 |
| 使用频率 | 取决于团队事件修改密度 |
| 使用方式 | 通过团队日程表界面接口 |

|  |  |
| --- | --- |
| 用例名称 | 创建团队日历 |
| 标识符 | UC05 |
| 主要参与者 | 组长或成员，两者在创建新团队时是平等的 |
| 说明 | 用户创建一个新团队 |
| 前置条件 | 1. 用户已经注册 2. 用户处在良好的网络环境中 |
| 后置条件 | 1、用户新建的日历被同步到组列表中，该用户成为该组的组长 |
| 触发器 | 用户点击新建日历按钮 |
| 基本操作流程 | 1. 用户输入新建的组名称 2. 点击提交按钮 |
| 可选操作流程 | 1、输入的组名为空：给出提示，用户补充输入组名称 |
| 优先级 | 必须实现 |
| 使用频率 | 每月多次 |
| 使用方式 | 通过团队管理界面接口 |

|  |  |
| --- | --- |
| 用例名称 | 解散团队日历 |
| 标识符 | UC06 |
| 主要参与者 | 组长，只有组长拥有解散团队的权力 |
| 说明 | 组长解散一个现有的团队并清除组内所有相关信息 |
| 前置条件 | 1. 用户已经注册 2. 用户处在良好的网络环境中 3. 用户在当前组中的身份是组长 |
| 后置条件 | 1. 该组从组长的管理列表和组员的参加列表中被同时移除 2. 该组内现存和曾有的所有团队事件被同步删除 |
| 触发器 | 组长点击组对应的删除按钮 |
| 基本操作流程 | 1. 给出删除警告提示 2. 用户确认删除 |
| 可选操作流程 | 1、用户取消删除操作：退出用例 |
| 优先级 | 必须实现 |
| 使用频率 | 使用频率较低，每月1-2次 |
| 使用方式 | 通过团队管理界面接口 |

|  |  |
| --- | --- |
| 用例名称 | 分享团队日历 |
| 标识符 | UC07 |
| 主要参与者 | 组长或组员 |
| 说明 | 用户将某个自己加入/创建组的链接分享到微信聊天中邀请他人加入 |
| 前置条件 | 1. 用户已经注册 2. 用户处在良好的网络环境中 |
| 后置条件 | 1、分享链接被成功转发到指定微信聊天中 |
| 触发器 | 点击组对应的分享按钮 |
| 基本操作流程 | 1. 给出分享提示 2. 用户确认分享 |
| 可选操作流程 | 1、用户取消分享操作：退出用例 |
| 优先级 | 必须实现 |
| 使用频率 | 每周多次 |
| 使用方式 | 通过团队管理界面接口 |

|  |  |
| --- | --- |
| 用例名称 | 加入团队日历 |
| 标识符 | UC08 |
| 主要参与者 | 组长或组员 |
| 说明 | 用户点击他人分享的链接加入一个组 |
| 前置条件 | 1. 用户处在良好的网络环境中 |
| 后置条件 | 1、该用户被成功加入链接指定的团队日历 |
| 被包含的用例 | 设置隐私模式 |
| 被扩展的用例 | 用户注册 |
| 触发器 | 点击聊天中其他用户的分享链接 |
| 基本操作流程 | 1. 给出加入确认信息 2. 用户选择是否对该组开启隐私模式 3. 点击确认加入按钮 |
| 可选操作流程 | 1、用户未注册：弹出访问微信个人信息请求，用户同意注册后继续给出确认信息，否则退出用例。 |
| 优先级 | 必须实现 |
| 使用频率 | 每月多次 |
| 使用方式 | 通过微信聊天界面的分享接口 |

|  |  |
| --- | --- |
| 用例名称 | 退出团队日历 |
| 标识符 | UC09 |
| 主要参与者 | 组员 |
| 说明 | 组员退出某个已加入的团队日历 |
| 前置条件 | 1. 用户已经注册 2. 用户处在良好的网络环境中 3. 用户指定的团队存在且身份为组员 |
| 后置条件 | 1. 指定团队日历从用户的加入列表中移除 2. 指定团队日历的团队事件从用户的日程表中移除 |
| 触发器 | 点击组对应的退出按钮 |
| 基本操作流程 | 1. 给出警告信息 2. 用户确认退出该组 |
| 可选操作流程 | 1、用户取消退出操作：退出用例 |
| 优先级 | 必须实现 |
| 使用频率 | 每月1-2次 |
| 使用方式 | 通过组管理界面接口 |

|  |  |
| --- | --- |
| 用例名称 | 设置勿扰模式 |
| 标识符 | UC10 |
| 主要参与者 | 组员 |
| 说明 | 组员对某个组设置勿扰模式，勿扰模式开启后，团队事件将不会被添加到该成员的个人日程中 |
| 前置条件 | 1. 用户已经注册 2. 用户处在良好的网络环境中 3. 用户指定的团队存在且身份为组员 |
| 后置条件 | 1、用户在该组的勿扰模式信息被更新 |
| 触发器 | 点击组对应的勿扰按钮 |
| 基本操作流程 | 1. 给出警告信息 2. 点击确认按钮 |
| 可选操作流程 | 无 |
| 优先级 | 必须实现 |
| 使用频率 | 每月多次 |
| 使用方式 | 通过团队日程管理界面接口 |

|  |  |
| --- | --- |
| 用例名称 | 查看日程安排 |
| 标识符 | UC11 |
| 主要参与者 | 组员或组长 |
| 说明 | 用户对所有的事件（包括私人事件和团队事件）进行查看 |
| 前置条件 | 1. 用户已经注册 2. 用户处在良好的网络环境中 |
| 后置条件 | 无 |
| 被包含的用例 | 查看个人日程、查看他人日程 |
| 触发器 | 用户点击个人日程安排按钮 |
| 基本操作流程 | 1、用户查看个人日程安排  2、用户切换组  3、用户切换组内成员，查看他人日程安排 |
| 可选操作流程 | 无 |
| 优先级 | 必须实现 |
| 使用频率 | 每天多次 |
| 使用方式 | 通过日程管理界面接口 |

|  |  |
| --- | --- |
| 用例名称 | 注册 |
| 标识符 | UC12 |
| 主要参与者 | 组员或组长 |
| 说明 | 初次进入系统的用户申请获得使用系统的权限 |
| 前置条件 | 1. 用户处在良好的网络环境中 |
| 后置条件 | 无 |
| 被包含的用例 | 无 |
| 触发器 | 用户初次进入本系统 |
| 基本操作流程 | 1、用户在微信小程序平台搜索或点击他人分享的链接初次进入本系统  2、系统提示用户注册  3、用户点击同意注册  4、系统将获取用户微信账户昵称、头像信息，作为本系统的账户信息，并创建新账户 |
| 可选操作流程 | 1、用户取消注册操作：退出用例 |
| 优先级 | 必须实现 |
| 使用频率 | 只有1次 |
| 使用方式 | 通过系统外部接口 |

## 2.4测试策略

1. 测试类型：按阶段划分定义为集成测试和系统测试。

2. 集成测试阶段进行了一轮集成测试，主要以需求挖掘、分析、确认和寻找实现与需求不一致为主要目标

3. 系统测试阶段基本策略如下：

第一轮为覆盖性测试，测试范围覆盖以上描述的所有范围，关注所有级别的bug；

第二轮对权重为A、B的模块进行功能测试、兼容性测试，权重为C的模块进行冒烟测试，回归测试所有已修复的bug；

第三轮对权重为A的模块进行功能测试、兼容性测试，对权重为B、C的模块进行冒烟测试，回归测试所有待解决的bug，及已关闭的高优先级bug。

接口测试只在第一轮进行。

4. 缺陷评估：每轮测试结束后都组织开发工程师、测试工程师、产品工程师等共同评估产品缺陷，评估内容包括缺陷解决方案、是否涉及需求变更、下一轮开始时间及是否可以结束测试等。

# 3.结果分析

整个测试过程中累计发现有效缺陷1个，其中A级缺陷0个，B级0个，C级1个。经项目组成员评估，到1.0发布止遗留缺陷0个，总共1个缺陷均已修复且全部验证通过。下面从不同角度对缺陷进行分析。

## 3.1 缺陷趋势

整个测试过程中累计发现缺陷1个，各轮次缺陷分布情况如下表。

从缺陷趋势图中可以看出，测试阶段，缺陷均随着测试过程的推进呈现收敛趋势，这符合测试缺陷的发展规律，证明测试计划和策略是可靠有效的。

## 3.2 缺陷优先级分布

每轮次各级别缺陷分布情况如下：

整个项目测试过程种中发现的C级以上（包括C级）缺陷1个，占总缺陷数的100%，这说明系统在测试过程中处于不稳定状态，存在较为严重的问题，但随着测试过程的推进，高优先级问题又逐渐减少，整个系统趋于稳定。

## 3.3 缺陷按模块分布

下表显示了整个测试过程中发现缺陷在各模块中的分布情况

|  |  |  |
| --- | --- | --- |
| 模块 | 缺陷数 | 百分比 |
| 显示用户日历 | 0 | 0% |
| 在日历中添加事件 | 1 | 100% |
| 修改和删除个人日历中的事件 | 0 | 0% |
| 事件提醒 | 0 | 0% |
| 创建群组和群组日历 | 0 | 0% |
| 参加一个小组 | 0 | 0% |
| 退出群组（仅限群组中的成员） | 0 | 0% |
| 解散一个组并删除组日历（只有组日历的所有者） | 0 | 0% |
| 通过微信小程序链接分享群，邀请他人加入群 | 0 | 0% |
| 添加组事件（仅限组日历的所有者） | 0 | 0% |
| 修改和删除组事件（仅限于组日历的所有者） | 0 | 0% |
| 查看团队成员的个人日历 | 0 | 0% |
| 增加团体活动时的空闲时间推荐 | 0 | 0% |
| 更改组 | 0 | 0% |
| 请勿打扰模式 | 0 | 0% |
| 隐私模式 | 0 | 0% |
| 修改用户名 | 0 | 0% |
| 显示用户日历 | 0 | 0% |
| 在日历中添加事件 | 0 | 0% |
| 修改和删除个人日历中的事件 | 0 | 0% |
| 事件提醒 | 0 | 0% |
| 创建群组和群组日历 | 0 | 0% |
| 参加一个小组 | 0 | 0% |
| 退出群组（仅限群组中的成员） | 0 | 0% |
| 合计 | 1 |  |

从下图中可以看出各模块缺陷的分布趋势：

从以上缺陷分布情况看，所有缺陷中和添加事件相关的，诸如需求定义欠明确、需求描述有歧义、需求没有定义、实现和需求不一致等。

## 3.4 重开缺陷情况

从上表可以看出整个测试过程中，各轮验证缺陷的重开比率都偏高，这是我们后续项目中需要关注和提高的地方。

## 3.5 遗留缺陷情况

到1.0发布止，整个项目遗留缺陷0个。

## 3.6 上线跟踪测试结果

1.0于7月2日上线后，我们在当日的7:00-12:00进行了集中跟踪测试，且在此之后安排有2名测试工程师，每天用一些时间跟踪上线情况最新动态。

# 4.总结

## 4.1 测试结论

1. 经过前后两个阶段的多轮测试，虽遗留了一些缺陷没有解决，但系统功能已趋于稳定，且项目确定的范围、策略和计划均已实现，项目测试可以结束、1.0可以上线。

2. 通过测试觉得产品在用户体验方面有待后续版本进一步改进，不排除用户在使用该产品时有“晕”的感觉。

## 4.2 呈现的问题

1. **需求问题。**项目需求有了很大改观，本身需求经过和收集、分析、确认和评审的过程，但对各接口产品的需求仍然没有进行统一的分析、确认和评审，这部分需求的歧义性较大且变更较多，整个需求文档的可读性、可测试性、完整性和清晰性仍然较差。

2. **变更控制问题。**1.0测试过程中这一问题得到了较好的改进，但变更控制规约的实施欠佳。所有的需求变更均没有及时很好的更新至需求文档。

3. **版本控制问题。**1.0测试过程中对本身的代码进行了版本管理，各接口产品的代码均由各技术负责人进行管理，在这期间出现过代码覆盖的情况、代码忘记上传或遗漏部署的情况。难以保证每轮测试版本的清晰、和发布版本与测试版本的一致性。

4. **测试环境问题。**1.0测试期间测试环境和开发环境没能很好的分离，导致测试和开发修复缺陷不能并行；测试期间有开发工程师直接在测试环境上修复缺陷和修改测试环境的情况；测试环境不稳定，如hosts设置不正确等。

## 4.3 测试建议

1. **遗留缺陷。**建议在1.0上线后，在后续版本中解决遗留缺陷，以提升产品的稳定性和用户体验。

2. **需求建议。**不论是本身还是各接口产品，建议进一步加强需求收集、分析、确认和评审过程，进一步提升需求文档的质量：减少需求的歧义性，提升需求的完整性、描述的清晰性、一致性、可读性、可实现性和可测试性。同时建议在后续项目中能对设计文档（如UI/UE等）进行评审，以增强产品的使用性、提升用户体验。

3. **变更控制。**建议在后续项目中进一步加强变更控制策略和规约制定，并强化变更控制规约的执行。不怕变更，关键要控制好变更的时机和策略。

4. **版本控制。**加强本身，特别是各接口产品的版本控制策略，以保证测试版本的清晰性、发布/上线版本和最终测试版本的一致性。

5. **测试环境。**期望在后续项目中及各接口产品的测试环境和开发环境完全分开，或阶段性完全独立，且各部分环境有专门的接口人负责，在测试期间严格禁止在测试环境上修复缺陷或更改环境配置（如确实需要更改配置，请提前通知测试及其它相关负责人）。以减少因此带来的沟通、反复侦测的成本。

6. **项目管理。**主要是建议加强项目的计划性，诸如：进度计划、人力资源计划、风险预防机制等，这也将更利于项目成员间高效的配合：大家能更适时的、更合理的制定各自工作计划，也更清楚到什么时候我会输出什么、我将配合他人做些什么。减少项目进行过程中的紧张和慌乱、项目也变得更加易控和可控。