<Vmeet>

测试报告

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 11/9/2022 | <1.0> | 进行了初步的测试 | 魏靖霖 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 范围 4

1.3 定义、首字母缩写词和缩略语 4

1.4 参考资料 4

1.5 概述 4

2. 测试概要 4

3. 测试环境 4

4. 测试结果及分析 4

4.1 需求覆盖率及缺陷分布 4

4.2 缺陷严重程度 5

5. 缺陷清单 5

5.1 功能性缺陷 5

5.2 非功能性缺陷 6

6. 测试结论与建议 6

测试报告

# 简介

## 目的

用于测试与用户交互的前端页面的性能以及后端的压力测试，对于后端的正确性进行单元测试。

## 范围

1. 前端性能的测试
2. 端到端的测试
3. 后端的压力测试
4. 后端的单元测试

## 定义、首字母缩写词和缩略语

单元测试:又称为模块测试,是针对程序模块（软件设计的最小单位）来进行正确性检验的测试工作

Junit: 一个 Java 编程语言的单元测试框架

E2ETest： 端到端测试

Puppeteer: 一个 Node 库，它提供了高级API 并通过 DevTools协议来控制 Chrome

JMeter: JAVA桌面应用程序，被设计为用于测试客户端/服务端结构的软件(例如web应用程序)

LightHouse: Google Chrome 推出的一款开源自动化工具，它可以搜集多个现代网页性能指标，分析 Web 应用的性能并生成报告

## 参考资料

无

## 概述

1， 功能测试用例包括对于业务逻辑的后端的接口进行的单元测试和手动进行和自动进行的端到端的测试。

2， 非功能测试包括对于前端的性能测试以及对于业务后端的压力测试，以及对前端页面的性能测试以及对于前端的首屏测试。

3， 组织方式：从功能测试到非功能测试注意陈述测试的结果。

# 测试概要

1. 单元测试，远端，王劭，Junit,对于后端的接口进行测试
2. E2eTest,远端，魏靖霖，puppeteer,对于从访问网页到普通的视频对话的测试。
3. 手动的端到端测试，远端，所有人，对于各种功能进行测试
4. 压力测试，远端，魏靖霖，JMeter，对于后端的一些接口进行测试
5. 前端首屏时间测试，远端，魏靖霖，puppeteer，多次访问网址，统计性能
6. 前端性能测试，远端，王奕文，lightouse

# 测试环境

硬件：16G内存的Windows主机，2核8G的Ubuntu云主机，4核16G的Ubuntu主机

# 测试结果及分析

## 需求覆盖率及缺陷分布

**表1 需求覆盖率及缺陷分布**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试内容** | | **测试**  **用例数** | **需求**  **覆盖率** | **缺陷数** | **缺陷率** | **备注** |
| **功能项** | **视频通话** | 1 | 33.33% | 0 | 0 | 全自动的端到端测测试开发效率低，剩下的手动进行测试 |
| **视频通话** | 3 | 66.67% | 0 | 0 | 剩下的一个使用自动化测试工具完成 |
| **文字聊天** | 1 | 100% |  |  |  |
| **功能项小计** | 5 | 100% | 4 | 0 |  |
| **非功**  **能项** | **性能** | 4 | 80% | 2 | 50% | 因为不是每个后端的接口都能够重复调用以及前端的性能测试 |
| **可靠性** | 8 | 100% | 2 | 12.5% | 手动测试，文字聊天会导致一些功能不可重入，以及文件传输的连接不稳定 |
|  |  |  |  |  |  |
|  | **非功能项小计** | 12 | 92.3% |  |  |  |
|  | **总计** | 17 | 94.1% |  |  |  |

## 缺陷严重程度

**表2 缺陷严重程度**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **严重程度** | **致命** | **严重** | **一般** | **细微** | **总数** |
| **缺陷个数** | 0 | 2 | 1 | 2 | 0 |
| **占缺陷百分比** | 0 | 50% | 25% | 25% | 0 |

# 缺陷清单

## 功能性缺陷

**表3 功能性缺陷列表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序**  **号** | **缺陷**  **编号** | **严重程度** | **功能模块** | **缺陷标题** | **缺陷描述** | **测试用例编号** |
| 1 | 1 | 严重 | 文字聊天 | 空白对象的非法访问 | 出现对空白对象的非法访问导致页面崩溃 | 9 |
| 2 | 2 | 一般 | 文件传输 | 连接不稳定 | P2P连接不稳定 | 10 |
| 3 |  |  |  |  |  |  |

表中相关项说明：

1. 缺陷编号：为每个缺陷赋予一个唯一的编号，可以通过此编号对缺陷进行跟踪。例如：Bug001。
2. 缺陷严重程度：缺陷可以根据严重程度分为以下几种情况。
3. 致命（fatal）：致命的错误，测试执行直接导致系统死机、蓝屏、挂起、或是程序非法退出；系统的主要功能或需求没有实现。
4. 严重（critical）：严重错误，系统的次要功能点或需求点没有实现；数据丢失或损坏。执行软件主要功能的测试用例导致系统出错，程序无法正常继续执行；程序执行过于缓慢或是占用过大的系统资源。
5. 一般（major）：不太严重的错误，这样的缺陷虽然不影响系统的基本使用，但没有很好地实现功能，没有达到预期的效果。如次要功能丧失，界面错误，打印内容、格式错误，提示信息不太正确，或用户界面太差，简单的输入限制未放在前台进行控制，删除操作未给出提示，操作时间长等。
6. 细微（minor）：一些小问题，对功能几乎没有影响，产品及属性仍可使用。如软件的实际执行过程与需求有较小的差异；程序的提示信息描述容易使用户产生混淆；有个别错别字、 文字排列不整齐；辅助说明描述不清楚，显示格式不规范，长时间操作未给用户进度提示，提示窗口文字未采用行业术语，可输入区域和只读区域没有明显的区分标志，系统处理未优化等。
7. 功能模块：所测试并出现该缺陷的功能模块名称。
8. 缺陷标题：描述缺陷的标题。
9. 缺陷描述：对缺陷的详细描述。描述缺陷的重现步骤和问题所在，描述需清晰明了，使研发人员可以按照描述将缺陷重现出来。必要时可以放上相应的截图，以便于研发人员理解问题所在。

## 非功能性缺陷

**表4 非功能性缺陷列表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序**  **号** | **缺陷**  **编号** | **严重程度** | **测试类型** | **缺陷标题** | **缺陷描述** | **测试用例编号** |
| 1 | 1 | 严重 | 性能 | 首屏时间过久 | 由于项目的过大导致首屏时间过长 | 11 |
| 2 | 2 | 细微 | 性能 | 好友邀请刷新时间过久 | 采用轮询的方式降低了不必要的算力支出 | 12 |
| 3 | 3 | 严重 | 可靠性 | 由于对空白对象的访问导致白屏 | 由于对空白对象的访问导致白屏 | 9 |
| 4 | 4 | 一般 | 可靠性 | P2P连接不稳定 | 文件传输功能的连接不稳定 | 10 |

# 测试结论与建议

问题：对于性能的测试要尽可能保证消除服务器中其他进程的影响，防止影响性能测试的结果。

结论：这次测试很好地帮我们找到了问题的存在，发现性能较差的位置，让我们能够开始去针对的优化对应的功能。

建议：建议测试应该伴随着开发进行，而不是在开发中后期才开始。