文件编号：从容应队-SWC2020-从容应队

受控状态：■受控 □非受控

保密级别：□公司级 □部门级 ■项目级 □普通级

采纳标准：CMMI DEV V1.2



****

素音

**Divoice**

项目开发文档

**Version 1**

2019.12.19

**Written by 从容应队**



**All Rights Reserved**

目录

[1 测试计划 1](#_Toc22846574)

[1.1 测试策略与目标 1](#_Toc22846575)

[1.2 测试范围 1](#_Toc22846576)

[1.3 测试环境 1](#_Toc22846577)

[2 单元测试 1](#_Toc22846578)

[2.1 \*\*\*\*模块 1](#_Toc22846579)

[2.1.1 测试用例与执行分析 1](#_Toc22846580)

[单元测试用例： 1](#_Toc22846581)

[分析结果： 1](#_Toc22846582)

[2.1.2 测试结果综合分析及建议 2](#_Toc22846583)

[2.1.3 测试经验总结 2](#_Toc22846584)

[3 功能测试 2](#_Toc22846585)

[3.1 \*\*\*\*功能 2](#_Toc22846586)

[3.1.1 测试用例与执行分析 2](#_Toc22846587)

[测试用例： 2](#_Toc22846588)

[分析结果： 2](#_Toc22846589)

[3.1.2 测试结果综合分析及建议 2](#_Toc22846590)

[3.1.3 测试经验总结 2](#_Toc22846591)

[4 系统测试 3](#_Toc22846592)

[4.1 模型性能测试 3](#_Toc22846593)

[4.1.1 测试用例与执行分析 3](#_Toc22846594)

[预期性能指标测试用例： 3](#_Toc22846595)

[分析结果： 3](#_Toc22846596)

[4.1.2 测试结果综合分析及建议 3](#_Toc22846597)

[4.1.3 测试经验总结 3](#_Toc22846598)

记录更改历史

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **更改原因** | **版本** | **作者** | **更改日期** | **备 注** |
| 1 | 创建 | 1.0.0 | 队员A | 2019/12/21 | 创建 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# 测试计划

## 测试策略与目标

【测试策略：测试策略在软件需求分析完成后就开始实施，根据项目需求对项目有一个整体的把握，包括：测试重点、测试难点、测试分层。】

【目标：定义项目在发布时候的质量等级】

【测试策略】

将测试过程分为两部分：前端测试以及后端测试。前后端测试又包括单元测试与继承测试，项目完成前还需进行集成前后端的系统测试，以确保系统最终集成之后能够符合预期正常运行。

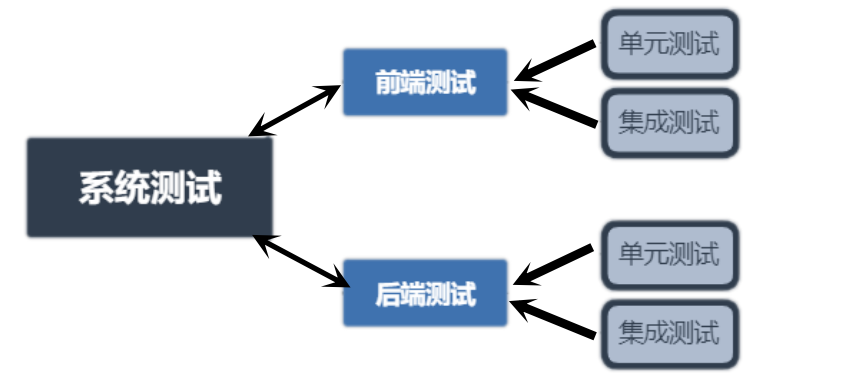
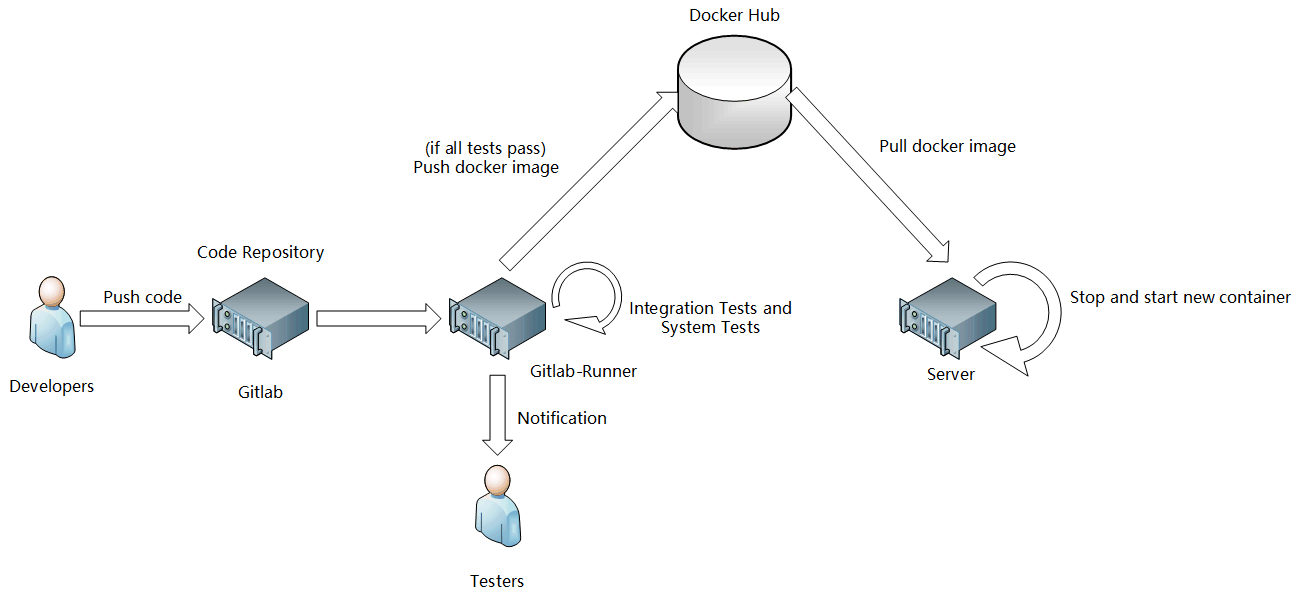


图1.1 Divoice项目测试构架

1. 前端单元测试使用测试框架（Jest）进行，以编写基本测试用例来检查各项函数的返回值的正确性为主（即黑盒测试）。同时，为了加快前后端开发进程，在前端单元测试中也使用Express搭建MockServer，模拟后端进行测试。
2. 前端集成测试需将通过单元测试的前端各个模块聚合，配合MockServer，在Nightwatch和Chromedriver的模拟环境下进行集成测试，用以保证个模块之间的正常协同运作。
3. 后端单元测试使用Django和Tornado自带单元测试工具，与前端单元测试相似，也是以编写基本测试用例检查各函数的返回值的正确性为主（黑盒测试）。
4. 后端集成测试使用Django的LiveServerTestCase进行，用以确保前后端能够正确协同工作。

使用CI(持续集成)/CD(持续交付和持续部署)架构开发Divoice。在本地的单元测试通过之后，向master提交代码，每次提交均会触发。Gitlab-ci.yml 定义的流水线操作。由测试服务器（Gitlab-Runners）拉取代码、自动进行各项测试。若各项测试通过则会向DockerHub推送新的镜像。服务器拉取新的镜像并部署。最后通知软件测试人员部署结果。若测试出现问题则会以邮件形式提示软件测试人员。因此，这样频繁的系统测试和集成测试能够极大地加快项目的迭代、更新速度，从而能够在问题出现早期及时发现并处理，从而大大降低debug的时间成本。

图1.2 CI/CD架构图

【测试重点】前端页面能够正确解析并显示后端的数据，同时后端应该确保业务逻辑的正确性，主要包括语音分离清晰度，保真度，音频文件编辑功能健壮性等等。

【测试难点】由于选择前后端分离开发，所以造成相对应的测试也是较为独立。建立了MockServer帮助前端独立于后端开发，编写单元测试，故采用Express模拟后端的数据进行测试，在系统各个结构都确定后再进行集成前后端的系统测试。后端的测试则利用Django的LiceServerTestCase进行单元测试和集成测试。

【测试分层】



图1.3 软件开发V模型（图片来源https://baijiahao.baidu.com/s?id=1611467961183421188&wfr=spider&for=pc）

测试由下至上：单元测试、集成测试、系统测试和用户的接受测试（上线之后）。其中前后端的测试相对独立，在进行系统测试时则会集成前后端。

【测试目标】

1. 单元测试首要保障各个单元（函数为主）的正确性，其次再是检查代码的覆盖率。在保证代码正确性的前提下，通过编写尽可能多的单元测试来覆盖单元的所有功能。
2. 集成测试目标为验证前端或者后端的各个模块业务逻辑的正确性。
3. 系统测试目标是检验前后端集成后的系统是否能够正确满足所有需求，其次是系统的各项非功能性的需求，后者包括产品的响应时间、产品的计算能力、容错性、抗压测试等等。

## 测试范围

测试广度：测试从尽可能多的角度对产品进行测试。前后端都将分别进行功能性测试以保证程序逻辑的正确性与包容性；同时还应该包括性能测试——保障用户的使用体验；安全锤屙屎——防御常见攻击（CSRF等）；稳定性测试——检验服务抗压能力等。

深度测试：测试对产品的每一个特性或功能进行尽可能多、尽可能深的测试。以编写大量样例来确保测试能够覆盖前后端的所有代码。每次测试的同时会进行代码覆盖率的分析，以保证项目的特性能够在测试中被包含。

## 测试环境

测试将使用安卓手机。前端测试环境：Node.js，测试工具：Jest，Express，nightwatch。后端测试环境：Python环境。测试工具：Django和Tornado自带的单元测试工具。CI/CD ：使用Gitlab内置的CI/CD服务。

# 单元测试

## \*\*\*\*模块

### 测试用例与执行分析

### 单元测试用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 001 | | | |
| 测试单元描述 |  | | | |
| 用例目的 |  | | | |
| 前提条件 |  | | | |
| 特殊的规程说明 |  | | | |
| 用例间的依赖关系 |  | | | |
| 步骤 | 输入 | 期望输出 | 实际输出 | 备注 |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| … | … | … | … | … |

### 分析结果：

### 测试结果综合分析及建议

### 测试经验总结

# 功能测试

## \*\*\*\*功能

### 测试用例与执行分析

### 测试用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 001 | | | |
| 功能描述 |  | | | |
| 用例目的 |  | | | |
| 前提条件 |  | | | |
| 特殊的规程说明 |  | | | |
| 用例间的依赖关系 |  | | | |
| 步骤 | 动作 | 期望结果 | 实际结果 | 备注 |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| … | … | … | … | … |

### 分析结果：

### 测试结果综合分析及建议

### 测试经验总结

# 系统测试

## 模型性能测试

### 测试用例与执行分析

### 预期性能指标测试用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 001 | | | |
| 性能描述 |  | | | |
| 用例目的 |  | | | |
| 前提条件 |  | | | |
| 特殊的规程说明 |  | | | |
| 用例间的依赖关系 |  | | | |
| 步骤 | 输入/动作 | 期望的性能  (平均值) | 实际性能  (平均值) | 备注 |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| … | … | … | … | … |

### 分析结果：

### 测试结果综合分析及建议

### 测试经验总结