文件编号：3107-SWC2018-20180045

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



图美集

**Temage**

项目测试文档

**Version 1.0.2**

2018.11.20

**Written by 3107**



**All Rights Reserved**

目录

[1 引言 1](#_Toc502825305)

[1.1 编写目的 1](#_Toc502825306)

[1.2 项目背景 1](#_Toc502825307)

[1.3 术语和缩略语 1](#_Toc502825308)

[1.4 参考资料 1](#_Toc502825309)

[2 测试需求分析 1](#_Toc502825310)

[2.1.1 测试策略与目标 1](#_Toc502825311)

[2.1.2 测试范围 2](#_Toc502825312)

[2.1.3 测试环境 2](#_Toc502825313)

[3 单元测试 2](#_Toc502825314)

[3.1 \*\*\*\*模块 2](#_Toc502825315)

[3.1.1 测试用例与执行分析 2](#_Toc502825316)

[单元测试用例： 2](#_Toc502825317)

[分析结果： 2](#_Toc502825318)

[3.1.2 测试结果综合分析及建议 3](#_Toc502825319)

[3.1.3 测试经验总结 3](#_Toc502825320)

[4 功能测试 3](#_Toc502825321)

[4.1 \*\*\*\*功能 3](#_Toc502825322)

[4.1.1 测试用例与执行分析 3](#_Toc502825323)

[测试用例： 3](#_Toc502825324)

[分析结果： 3](#_Toc502825325)

[4.1.2 测试结果综合分析及建议 3](#_Toc502825326)

[4.1.3 测试经验总结 3](#_Toc502825327)

[5 系统测试 4](#_Toc502825328)

[5.1 模型性能测试 4](#_Toc502825329)

[5.1.1 测试用例与执行分析 4](#_Toc502825330)

[预期性能指标测试用例： 4](#_Toc502825331)

[分析结果： 4](#_Toc502825332)

[5.1.2 测试结果综合分析及建议 4](#_Toc502825333)

[5.1.3 测试经验总结 4](#_Toc502825334)

[5.2 压力测试 4](#_Toc502825335)

[5.2.1 测试用例与执行分析 4](#_Toc502825336)

[核心模块A压力测试用例： 4](#_Toc502825337)

[分析结果： 5](#_Toc502825338)

[分析结果： 5](#_Toc502825339)

[5.2.2 测试结果综合分析及建议 5](#_Toc502825340)

[5.2.3 测试经验总结 5](#_Toc502825341)

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C | 2018/11/17 | 初稿 |
| 2 | 更新 | 1.0.1 | 队员C | 2018/11/18 | 整体修改 |
| 3 | 更新 | 1.0.2 | 队员C | 2018/11/19 | 初赛终稿 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# 引言

## 编写目的

编写本测试文档的核心意义在于明确本项目的测试目标，为项目的测试建立系统的计划以及分析项目的结构进而设计测试过程以及实现方案。

## 项目背景

Temage智能图文排版项目是一个运用深度学习技术实现自动图文排版的web应用。用户给定文本和图片，即可根据文章的主题，在线智能排版文本与图片，选择合适的风格，生成一篇优美悦目的文章。能够极大地节省文章从排版到发布的时间，提高用户的效率和生产力。

项目针对当下大众资讯获取、媒体信息传播的新变化，具有很好的应用前景和广大的目标用户群体。

Temage注重用户的体验，旨在为用户提供优美友好的界面和优质流畅的服务。

## 术语和缩略语

[1] Vue.js：Vue.js是一个构建数据驱动的 web 界面的渐进式框架。Vue.js的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。它不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与单文件组件和 Vue.js生态系统支持的库结合使用时，Vue.js也完全能够为复杂的单页应用程序提供驱动。

[2] Karma: A simple tool that allows you to execute JavaScript code in multiple real browsers. 一款前端测试Javascript框架

[3] CI: CI的全称是Continuous Integration，表示持续集成。在CI环境中，开发人员将会频繁地向主干提交代码。这些新提交的代码在最终合并到主干前，需要经过编译和自动化测试流进行验证。持续集成过程中很重视自动化测试验证结果，以保障所有的提交在合并主线之后没有问题，或对可能出现的一些问题进行预警。

[4] CD: CD的全称是Continuous Deployment，表示持续部署。在CD环境中，通过自动化的构建、测试和部署循环来快速交付产品。该产品可以进行灰度测试或是直接上线运营。

[5] Mock.js: Mock.js 是一款模拟数据生成器，旨在帮助前端独立于后端进行开发，帮助编写单元测试。提供了根据数据模板生成模拟数据、模拟 Ajax 请求，生成并返回模拟数据和基于 HTML 模板生成模拟数据

## 参考资料

[1] Karma-npm Available online on https://www.npmjs.com/package/karma

[2] Mockjs online on http://mockjs.com/0.1/

# 测试计划

### 测试策略与目标

测试过程可大致分成两大部分：前端测试以及后端测试。前后端的测试又可细分为单元测试、集成测试以及系统测试。

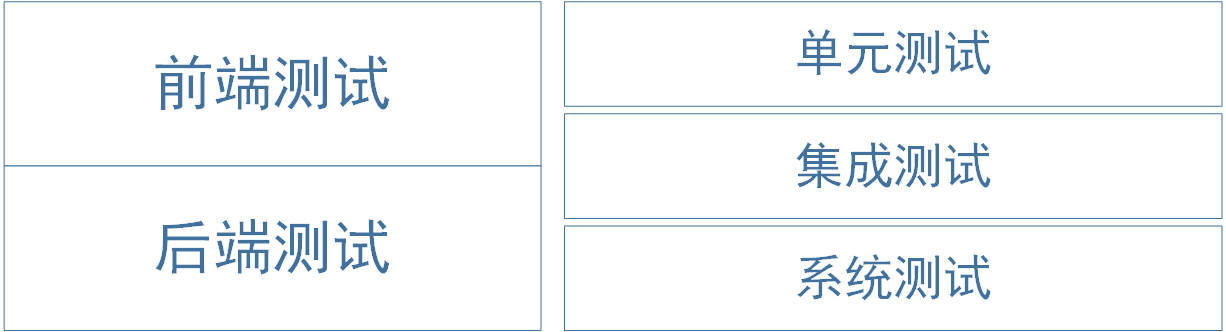


图1.1 Temage项目测试架构

前端单元测试主要为编写基本测试用例来检查各项函数的返回值是否正确，此部分测试可以使用测试框架（Karma）进行。鉴于项目的前后端是分离开发的，在前端单元测试中我们也会使用Mock.js作为随机数据的生成器，模拟后端的数据进行测试。

前端的集成测试要求将通过单元测试的前端各模块聚合起来，配合Mock.js进行前端的集成测试,确保各模块之间能够正常协同运作。除了正确性，集成测试中的代码覆盖率将是我们的另一重要关注点。我们计划经常性地进行集成测试以确保任何时刻都有一份稳定且版本较新的代码可供部署。

后端单元测试的和集成测试将会使用Django自带的单元测试工具。测试策略与前端的代码相似。

后端集成测试将会使用Django的LiveServerTestCase进行，确保前后端能够正确协同工作。

系统测试时检验项目完成度的一大指标，我们决定使用在Django的LiveServerTestCase中使用selenium模拟操作前端，进行前后端集成的系统测试。

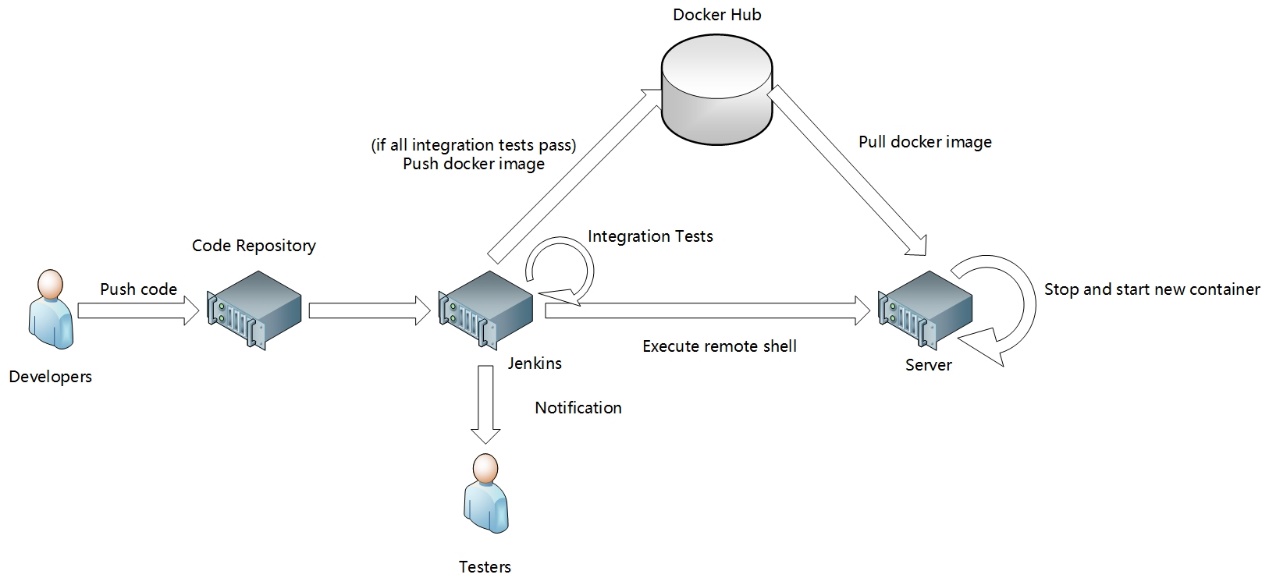


图1.2 CI/CD架构图

/\* 前端集成测试则是将每份经过单元测试的代码集中提交到主干，之后使用CI环境自动进行编译并运行编写的集成测试样例，保障所用样例均能通过，同时生成一份代码覆盖率报告，以加速代码的聚合。在前端集成测试中应确保代码覆盖率高于70%（除前端推断部分）。前端系统测试会使用CD框架进行，确保任何时刻都有一份相对稳定可靠的发布版本供一小部分用户进行内测。我们只进行前端的单元测试和集成测试，系统测试将在后期进行。后端单元测试将会使用Django自带的单元测试工具。 \*/

测试目标：

【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：测试重点前端页面正确显示后端业务逻辑、测试难点前后端分离mock、测试分层。】

【目标：定义项目在发布时候的质量等级】

我们的测试重点为：前端页面应能正确解析并显示后端的数据，同时后端应确保业务逻辑的正确性，主要包括用户登录、编辑、推荐、分享等功能。

我们的测试难点为：我们选择前后端分离进行开发，则相对应的测试也是较为独立的。我们建立的mock server帮助前端独立于后端进行开发，帮助编写单元测试。对此，我们选择采用Mock.js模拟后端的数据进行测试。在系统各接口均确定之后再进行集成前后端的系统测试；后端的系统测试将会使用Django的LiveServerTestCase进行单元测试和集成测试。

我们的测试将分为如下几个层次：



图1.3 测试分层图（图片来自http://www.anexinet.com/wp-content/uploads/Acceptance.png）

总体上的测试由下至上为单元测试、集成测试、系统测试和用户的接受测试（上线之后）。其中前后端的测试又相对独立。在进行系统测试时则会进行前后端的集成。

项目的测试目标有如下几点：

1. 能够实现用户的登陆注册功能
2. 接收用户输入的未排版的文本与图片，在前端进行初步推断之后交给后端确定可能的风格给用户进行选择
3. 在利用深度学习技术为用户排版之后，用户可进行二次编辑，得到最终稿。
4. 将最终稿以长图的形式反馈给用户
5. 用户可发布、分享自己的作品

### 测试范围

【从测试广度和测试深度两方面了解整个测试项目的测试规模】

从测试广度的角度来说，我们将对前后端的代码进行功能性测试，确保使用逻辑的完整与正确；性能测试，保障用户的使用体验；安全测试，防御常见攻击（csrf等）；易用性测试，确保用户能快速上手；稳定性测试，保障服务抗压能力；兼容性测试，保障web前端在不同主流浏览器上都能正常运行。

从测试深度的角度而言，我们的测试将会覆盖前后端的所有代码。每次测试都将会进行代码覆盖的分析，确保项目的所有特性都能在测试中被覆盖到。

### 测试环境

【包括软硬件环境、网络环境、测试工具】

所有测试将会运行在有英特尔芯片的机器上。其中前端测试运行在Node.js环境中，测试工具为Karma, Mock。后端测试运行在python环境中。测试工具为Django自带的单元测试工具。

# 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

## 压力测试

### 测试用例与执行分析

### 核心模块A压力测试用例：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 功能 |  | | | | | |
| 目的 |  | | | | | |
| 方法 |  | | | | | |
| 并发用户数与事务执行情况 | | | | | | |
| 并发用户数 | 事务平均响应时间 | 事务最大响应时间 | 平均每秒处理事务数 | 事务成功率 | 每秒点击率 | 平均流量(字节/秒) |
| 20 |  |  |  |  |  |  |
| 25 |  |  |  |  |  |  |
| … | … | … | … | … | … | … |

### 分析结果：

[模块A, 模块B, 模块C]组合压力测试用例：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 功能 |  | | | | | | | | | | | | | |
| 目的 |  | | | | | | | | | | | | | |
| 方法 |  | | | | | | | | | | | | | |
| 并发用户数与事务执行情况 | | | | | | | | | | | | | | |
| 并发用户数 | 事务平均响应时间 | | | 事务最大响应时间 | | | 平均每秒处理事务数 | | | 事务成功率 | | | 每秒点击率 | 平均流量(字节/秒) |
| A | B | C | A | B | C | A | B | C | A | B | C |
| 20 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 25 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| … | … | … | … | … | … | … | … | … | … | … | … | … | … | … |

### 分析结果：

### 测试结果综合分析及建议

### 测试经验总结