**东华大学**

软件工程课程设计实验报告

——《轻易英语阅读小程序》

目录

[一、项目开发计划 2](#_Toc11603075)

[1、引言 2](#_Toc11603076)

[1.1 编写目的 2](#_Toc11603077)

[1.2 项目背景 2](#_Toc11603078)

[1.3 定义 2](#_Toc11603079)

[1.4 参考资料 3](#_Toc11603080)

[2、项目概述 3](#_Toc11603081)

[2.1 工作内容 3](#_Toc11603082)

[2.2 主要参加人员 4](#_Toc11603083)

[2.3 产品 4](#_Toc11603084)

[2.4 验收标准 5](#_Toc11603085)

[2.5 完成项目最迟日期 5](#_Toc11603086)

[2.6 本计划的批准者和批准日期 5](#_Toc11603087)

[3、实施计划 5](#_Toc11603088)

[3.1 工作任务的分解和人员分工 5](#_Toc11603089)

[3.3 进度 6](#_Toc11603090)

[3.4 关键问题 6](#_Toc11603091)

[4、专题计划要点 6](#_Toc11603092)

[二、需求分析 7](#_Toc11603093)

[1引言 7](#_Toc11603094)

[1.1编写目的 7](#_Toc11603095)

[1.2背景 7](#_Toc11603096)

[1.3项目介绍 7](#_Toc11603097)

[1.4 定义 7](#_Toc11603098)

[1.5参考资料 8](#_Toc11603099)

[2任务概述 8](#_Toc11603100)

[2.1目标 8](#_Toc11603101)

[2.2运行环境 8](#_Toc11603102)

[3需求规定 9](#_Toc11603103)

[3.1对功能的规定 9](#_Toc11603104)

[3.1.2 系统用例图 10](#_Toc11603105)

[3.1.3 各功能模块描述 10](#_Toc11603106)

[3.2对性能的规定 11](#_Toc11603107)

[3.2.1精度 11](#_Toc11603108)

[3.2.2时间特性要求 11](#_Toc11603109)

[3.2.3安全性需求 11](#_Toc11603110)

[3.2.4 软件质量属性 12](#_Toc11603111)

[3.2.5 业务规则 12](#_Toc11603112)

[3.3 数据要求 12](#_Toc11603113)

[4运行环境规定 12](#_Toc11603114)

[4.1设备 12](#_Toc11603115)

[4.2接口 12](#_Toc11603116)

[三、概要设计 13](#_Toc11603117)

[1、总体设计 13](#_Toc11603118)

[1.1 需求规定 13](#_Toc11603119)

[1.2 运行环境 13](#_Toc11603120)

[1.3 基本设计概念和处理流程 13](#_Toc11603121)

[1.4 结构 14](#_Toc11603122)

[1.5 人工处理过程 15](#_Toc11603123)

[2、接口设计 15](#_Toc11603124)

[2.1 用户接口 15](#_Toc11603125)

[2.2 外部接口 15](#_Toc11603126)

[2.3 内部接口 16](#_Toc11603127)

[3、运行设计 16](#_Toc11603128)

[3.1 运行模块组合 16](#_Toc11603129)

[3.3 运行时间 17](#_Toc11603130)

[4、系统数据结构设计 18](#_Toc11603131)

[4.1 逻辑结构设计要点 18](#_Toc11603132)

[4.2 物理结构设计要点 20](#_Toc11603133)

[4.3 数据结构与程序的关系 22](#_Toc11603134)

[5、系统出错处理设计 22](#_Toc11603135)

[5.1 出错信息 22](#_Toc11603136)

[5.2 补救措施 23](#_Toc11603137)

[5.3 系统维护设计 23](#_Toc11603138)

[四、详细设计 24](#_Toc11603139)

[1、程序系统的结构 24](#_Toc11603140)

[1.1 创建用户模块设计说明 24](#_Toc11603141)

[1.2 推荐文章模块设计说明 25](#_Toc11603142)

[1.3 词汇测试模块设计说明 28](#_Toc11603143)

[1.4 获取新闻内容模块设计说明 31](#_Toc11603144)

[1.5收藏文章模块设计说明 33](#_Toc11603145)

[1.6 最近浏览模块设计说明 35](#_Toc11603146)

[1.7 背单词模块设计说明 37](#_Toc11603147)

[1.8 生词本模块说明分析 39](#_Toc11603148)

[五、测试计划 42](#_Toc11603149)

[1引言 42](#_Toc11603150)

[1.1编写目的 42](#_Toc11603151)

[1.2背景 42](#_Toc11603152)

[1.4参考资料 42](#_Toc11603153)

[2计划 43](#_Toc11603154)

[2.1软件说明 43](#_Toc11603155)

[2.2测试内容 44](#_Toc11603156)

[2.3进度安排 44](#_Toc11603157)

[3测试设计说明 45](#_Toc11603158)

[3.1 功能测试 45](#_Toc11603159)

[3.1.1词汇量测试功能模块 45](#_Toc11603160)

[3.1.2个性化推荐文章功能模块 46](#_Toc11603161)

[3.1.3 分类文章功能模块 47](#_Toc11603162)

[3.1.4 取词功能翻译 49](#_Toc11603163)

[3.1.5 文章管理功能模块 50](#_Toc11603164)

[3.1.6 生词本功能模块 53](#_Toc11603165)

[3.2 性能测试 55](#_Toc11603166)

[3.3 界面测试 55](#_Toc11603167)

[3.4 接口测试 56](#_Toc11603168)

[4评价准则 57](#_Toc11603169)

[4.1范围 57](#_Toc11603170)

[4.2数据整理 57](#_Toc11603171)

[5 资源需求 57](#_Toc11603172)

[六、测试结果分析 58](#_Toc11603173)

[1引言 58](#_Toc11603174)

[1.1编写目的 58](#_Toc11603175)

[1.2背景 58](#_Toc11603176)

[1.3定义 58](#_Toc11603177)

[1.4参考资料 58](#_Toc11603178)

[2测试概要 59](#_Toc11603179)

[3测试结果及发现 59](#_Toc11603180)

[3.1测试用例1-文章分类分类的识别度 59](#_Toc11603181)

[3.2测试用例2-文章个性化推荐 60](#_Toc11603182)

[3.3 测试用例3-词汇量之性能测试 61](#_Toc11603183)

[3.4 测试用例4-词汇量测试之准确度测试 61](#_Toc11603184)

[3.5 测试用例5-测试文章的热点界面排版 62](#_Toc11603185)

[3.6 测试用例6-测试文章推荐界面排版 63](#_Toc11603186)

[3.7 测试用例7-测试文章详细信息排版 64](#_Toc11603187)

[3.8 测试用例8-测试文章的收藏功能 65](#_Toc11603188)

[3.9 测试用例9-测试文章分类信息排版 66](#_Toc11603189)

[3.10 测试用例-文章获取的响应时间 67](#_Toc11603190)

[3.11 测试用例—取词翻译的准确度 67](#_Toc11603191)

[3.12 测试用例-取词翻译的响应时间 68](#_Toc11603192)

[3.13 测试用例-取词翻译之具体释义的准确度 68](#_Toc11603193)

[3.14 测试用例-文章管理之收藏文章性能测试 68](#_Toc11603194)

[3.15 测试用例-文章管理之获取收藏文章性能测试 69](#_Toc11603195)

[3.16 测试用例-文章管理之收藏文章容错性 69](#_Toc11603196)

[3.17 测试用例-文章管理之删除收藏文章准确性 69](#_Toc11603197)

[3.18 测试用例-文章管理之获取最近浏览文章性能测试 69](#_Toc11603198)

[3.19 测试用例-生词本之添加单词准确性 69](#_Toc11603199)

[3.20 测试用例-生词本之删除单词准确性 70](#_Toc11603200)

[3.21 测试用例-生词本之获取单词性能 70](#_Toc11603201)

[3.22测试用例-生词本之单词复习 70](#_Toc11603202)

[3.23测试用例-小程序性能测试 70](#_Toc11603203)

[4对软件功能的结论 71](#_Toc11603204)

[4.1功能1-英文文章分类 71](#_Toc11603205)

[4.1.1能力 71](#_Toc11603206)

[4.1.2不足之处 71](#_Toc11603207)

[4.2功能2-文章个性化推荐 71](#_Toc11603208)

[4.2.1 能力 71](#_Toc11603209)

[4.2.2 不足之处 71](#_Toc11603210)

[4.3 功能3-词汇量测试 71](#_Toc11603211)

[4.3.1 能力 71](#_Toc11603212)

[4.3.2 不足之处 71](#_Toc11603213)

[4.4 功能4-文章排版和细节显示 72](#_Toc11603214)

[4.4.1 能力 72](#_Toc11603215)

[4.4.2 不足之处 72](#_Toc11603216)

[4.5 功能5-收藏功能 72](#_Toc11603217)

[4.3.1 能力 72](#_Toc11603218)

[4.3.2 不足之处 72](#_Toc11603219)

[4.6 功能6-生词本 72](#_Toc11603220)

[能力 72](#_Toc11603221)

[不足之处 72](#_Toc11603222)

[4.7 单词复习历史纪录 72](#_Toc11603223)

[能力 72](#_Toc11603224)

[5分析摘要 73](#_Toc11603225)

[5.1能力 73](#_Toc11603226)

[5.2缺陷和限制 73](#_Toc11603227)

[5.3建议 73](#_Toc11603228)

[5.4评价 73](#_Toc11603229)

[七、 项目总结 74](#_Toc11603230)

[1、 实际开发结果 74](#_Toc11603231)

[1.1 软件产品描述 74](#_Toc11603232)

[2.2 主要功能与性能 75](#_Toc11603233)

[2.3进度 76](#_Toc11603234)

[2、 开发工作评价 77](#_Toc11603235)

[3.1 对项目开发过程的评价 77](#_Toc11603236)

[3.2 对技术方法的评价 78](#_Toc11603237)

[3.3 对产品质量的评价 78](#_Toc11603238)

[3、 试运行工作评价 79](#_Toc11603239)

[4、 技术积累总结 79](#_Toc11603240)

[5、 经验与教训 79](#_Toc11603241)

[5.1 邓晏湘 79](#_Toc11603242)

[5.2 陈彩婷 81](#_Toc11603243)

[5.3 汪伦 82](#_Toc11603244)

# 一、项目开发计划

## 1、引言

### 1.1 编写目的

为了保证项目的顺利进行，更好的规划团队的任务分配。 把对于在项目生命周期内的工作任务范围、各项工作的任务分解、项目团队组织结构、各团队成员的工作责任、团队内外沟通协作方式、开发进度、经费预算、项目内外环境条件、风险对策等内容做出的安排以书面的方式，作为项目团队成员以及项目干系人之间的共识与约定，项目生命周期内的所有项目活动的行动基础，项目团队开展和检查项目工作的依据。

### 1.2 项目背景

开发软件名称：《轻易在线阅读》

项目任务提出者：邓晏湘，陈彩婷，汪伦

项目开发者：邓晏湘，陈彩婷，汪伦

用户：喜欢英语新闻阅读，希望通过阅读提高英语水平的群体

### 1.3 定义

* 微信小程序：微信小程序，简称小程序，英文名Mini Program，是一种不需要下载安装即可使用的应用、它实现了应用“触手可及”的梦想，用户扫一扫或搜一下即可打开应用。
* django：Django是一个开放源代码的Web应用框架，由Python写成。采用了MVT的框架模式，即模型M，视图V和模版T。它最初是被开发来用于管理劳伦斯出版集团旗下的一些以新闻内容为主的网站的，即是CMS（内容管理系统）软件。并于2005年7月在BSD许可证下发布。
* WXML语言：微信小程序开发所使用的语言。WXML（WeiXin Markup Language）是框架设计的一套标签语言，结合基础组件，事件系统，可以构建出页面的结构。
* GitHub：GitHub是一个面向[开源及私有软件项目的托管平台，因为只支持git 作为唯一的版本库格式进行托管，故名GitHub。

### 1.4 参考资料

《软件工程导论》，张海藩，清华大学出版社。

《实用软件工程》，郑人杰等，清华大学出版社。

《软件工程导论》，张海藩（第五版），清华大学出版社；

[微信小程序开发文档](https://developers.weixin.qq.com/miniprogram/dev/framework)

[django开发文档](https://www.djangoproject.com/)

## 2、项目概述

### 2.1 工作内容

系统规划：根据对于问题的研究和调查，进行相关可行性研究，写出项目计划书和可行性研究报告

系统需求分析：分析目标和任务，构思基本业务流程，画出数据流图，ER图等，并编写数据字典

系统总体设计：画出系统结构图，找出系统模块，并设计数据库，编写概要设计说明书。

系统详细设计：画出基本逻辑结构图，代码设计，用户界面设计，数据输入与显示，控制界面的设计，系统安全控制设计，编写详细设计文档。

系统测试：系统实施与维护。

使用说明：针对普通用户群体，编写系统使用手册，安装手册，以及编码清单。

项目报告：小组成员对整个开发过程进行分析总结，并编写项目总结报告。

### 2.2 主要参加人员

**前端/UI设计**

* 陈彩婷：擅长UI设计以及vue框架的使用和开发，综合能力较强，能够编写鲁棒性强，交互性好的前端界面，熟悉各种资源的使用。
* 汪伦：自主开发独立性强，创造性思维强，能够在前人的基础上构思新的方法，不仅熟知前端开发同时精通后端的数据处理。

**后端**

* 邓晏湘：能力一般，比较擅长面向搜索引擎的代码理解和移植。

### 2.3 产品

#### 2.3.1 程序

系统名称：轻易在线阅读

编程语言：Python，Javascript

功能

* 个性化推荐最适合用户阅读的文章
* 测试词汇量
* 阅读最新的英文时事要闻
* 背单词

#### 2.3.2 文件

应交付的文档：可行性研究文档，项目开发计划， 需求分析，系统设计，测试计划，结果分析，项目总结，可独立运行的程序，使用手册

### 2.4 验收标准

文件：说明具体详尽完整，能够体现小组的工作量。格式满足国家标准文件规范，易读性强，并且结合项目实际情况，无夸大。

程序：鲁棒性强，实现了要求的功能，用户体验感好。

### 2.5 完成项目最迟日期

2019年6月17日（周日）之前

### 2.6 本计划的批准者和批准日期

软件工程课程设计授课老师：王鹏伟

## 3、实施计划

### 3.1 工作任务的分解和人员分工

项目开发计划 负责人：邓晏湘

需求分析 负责人：陈彩婷

（系统）概要设计 负责人：邓晏湘

（系统）详细设计 负责人：邓晏湘

测试计划 负责人：陈彩婷，汪伦

测试结果分析 负责人：陈彩婷，汪伦

项目总结 负责人：邓晏湘，陈彩婷，汪伦

使用手册：汪伦

安装手册：汪伦

编码清单：邓晏湘

### 3.3 进度

开发进度规划以及每个阶段需要完成的任务由甘特图给出。

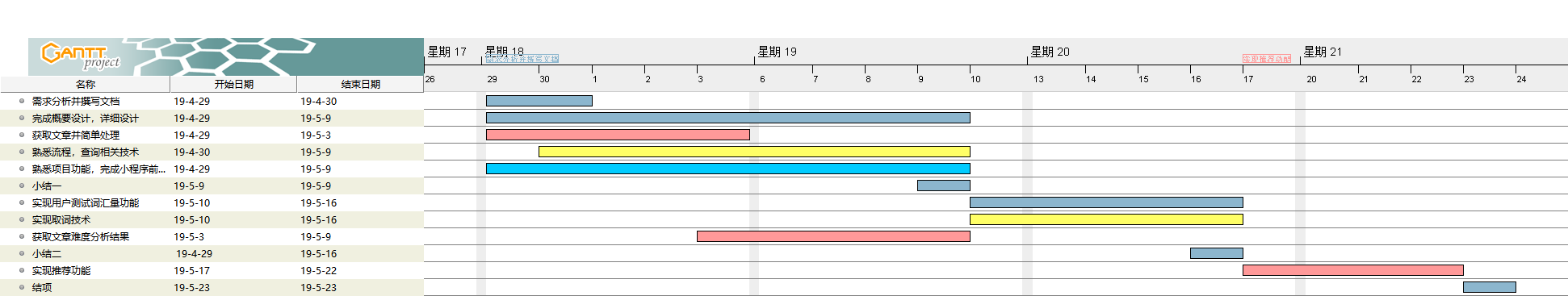


表 1开发进度规划（甘特图）

### 3.4 关键问题

* 测试词汇量：测试词汇量的可靠程度直接影响推荐的文章能否满足用户需求，提升用户的阅读体验。
* 根据词汇量个性化推荐文章：个性化的定制程度直接影响用户是否愿意一直使用该软件。
* 爬取各大外国新闻网站文章：有些国外网站不允许爬取文章以及爬取文章每天更新背后的逻辑会对系统的稳健性造成一定的影响。
* 点击取词翻译：如何识别词汇，并且显示词汇含义对于实现添加生词，背单词都起到了决定性的作用。如果不能实现直接影响了后续功能。

## 4、专题计划要点

测试计划：后端边编写边自测，前端编写完并且和后端的接口联系起来后再统一进行测试。

质量保证计划：在开发的过程中，小组成员随时保持交流，并且对彼此的代码进行了解，提出疑问，相互监督和检查。

配置管理计划：利用现有设备对整个系统进行开发。

系统安装计划：在每个小组成员的机器上进行安装调试，保证设备都能正常工作。

# 二、需求分析

# 1引言

## 1.1编写目的

确定轻易英语阅读小程序功能的有效性需求；以供本系统的开发人员分析讨论，进一步制定本软件开发的细节问题。并且为本软件的用户提供一个清晰的认识，说明了本产品的各项功能需求、性能需求和数据要求，明确各功能的实现过程，阐述可实用背景和范围。

## 1.2背景

说明：

* 开发软件名称：轻易英语阅读
* 项目提出者：邓晏湘

开发者：邓晏湘、陈彩婷、汪伦

用户：通过阅读时事新闻提高英语水平的学习者

* 同其他系统或其他机构的基本的相互来往关系：

1. 从CNN网站上获取一定数量的当天新闻文章
2. 调用有道翻译API、扇贝API获取英语单词的详细释义

## 1.3项目介绍

“轻易英语阅读”是一个根据英语学习者的实际词汇量水平，提供最切合用户的英语实时文章的小程序。英语新闻类的阅读小程序较少，并且可选择范围有限，无法个性化推荐。本软件以小程序为开发平台，分为定制化阅读、推荐阅读、词汇测试、单词本等功能模块，为广大英语学习者提供轻量型学习平台。

## 1.4 定义

Scrapy：爬虫框架，提取结构性数组，可快速抓取到数据内容；

CNN：Cable News Network，美国有线电视新闻网；

API：应用程序编程接口，是一些预先定义的函数，使应用程序与开发人员基于某软件或者硬件得以访问一组例程的能力。

## 1.5参考资料

（1）《软件工程导论》

（2）《微信小程序开发——零基础入门》

# 2任务概述

## 2.1目标

（1）开发意图：

* 英语新闻类阅读小程序较少
* 开发一块免费并且无需强制性推广的英语阅读小程序
* 做到能够为用户进行个性化推荐

（2）应用目标

* 长期有效地运行
* 界面美观，良好的用户体验、
* 文章内容的及时更新
* 有效记录用户的学习历程

## 2.2运行环境

操作系统：Microsoft Windows 10

软件设备：微信web开发者工具

数据库：sqlite

# 3需求规定

## 3.1对功能的规定

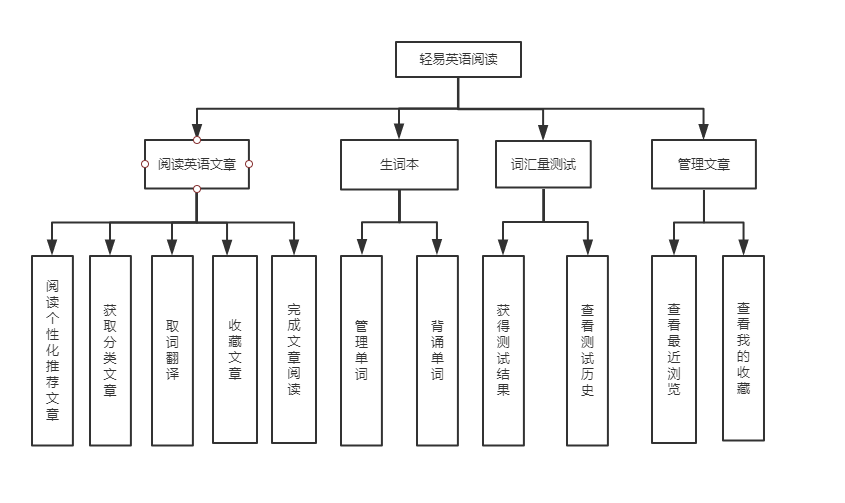


图3-1 整体软件功能模块图

### 3.1.2 系统用例图

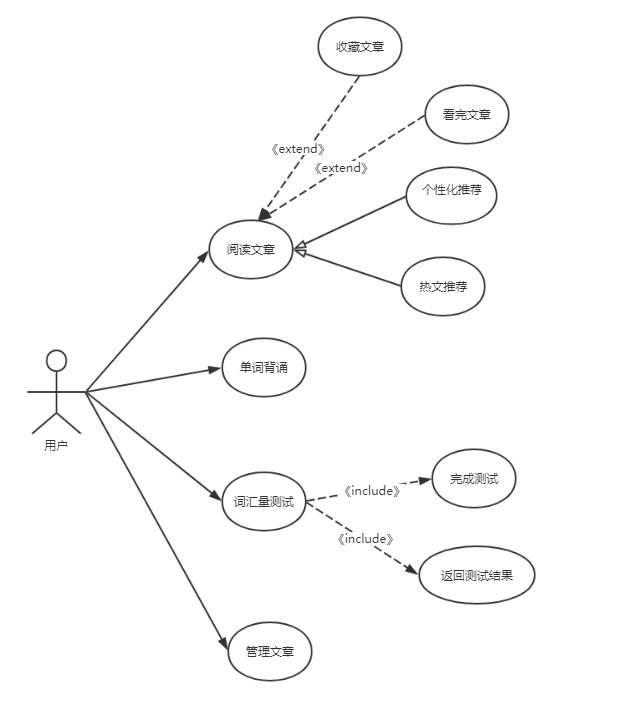


图3-2 系统用例图

### 3.1.3 各功能模块描述

#### 3.1.3.1 个性化推荐文章阅读

* 功能描述：

为用户提供小于10篇的个性化推荐的文章。这些文章从每天所有从CNN获取的文章中选取与用户匹配度最高的少量文章数据。匹配的程度根据用户的词汇量测试结果判断。当用户还未完成词汇量测试时，该功能为空，提示用户完成测试。

#### 3.1.3.2 各类文章阅读

* 功能描述：

为用户提供每日从CNN获取到的所有文章数据，并进行分类显示，包括等级分类与类别分类，为用户提供多样化选择。

#### 3.1.3.3 取词翻译

* 功能描述：

用户阅读文章时，点击不认识的英语单词，可出现对应单词释义，并且可跳转到详细释义界面、添加到生词本中。

#### 3.1.3.4 文章管理

* 功能描述：

用户可根据自己的喜好收藏感兴趣的文章，收藏成功的文章将与用户进行关联存储，可以在我的收藏列表中查看收藏数据。

当用户完成一篇文章的阅读之后，系统将该文章数据存储起来；用户可以找到自己的近期阅读。

#### 3.1.3.5 生词本

* 功能描述：

用户在阅读一篇文章的过程中，当遇到不懂的单词，可以使用添加到生词本中，记录下来。查看生词本中自己的添加历史，可对生词进行删除管理；还能背诵其中的单词，加强记忆。

#### 3.1.3.6 词汇量测试

* 功能描述：

用户可随时进行词汇量测试，这个测试结果不仅是提供个性化推荐文章的基础，也是用户对自己的学习进步水平的一个认识。系统提供一组分7个层次的单词数组，给出释义，用户，再随机获取不同的单词作为选项，用户选择其中的正确单词，或者“不认识”，以获取最为准确的测试数据。

测试结束后，为了对单词有更加深刻的印象，将测试过程中出现的每一个单词记录下来，包括对错、等级等。用户在测试后也能加强记忆。

## 3.2对性能的规定

### 3.2.1精度

* 词汇量测试数据精度：后端处理的用户词汇测试结果为各等级小数数组，要求精确到一定位数
* 最佳匹配文章推荐：在对用户进行个性化推荐文章匹配时，要求匹配度不能低于特定值
* 单词释义：单词与释义一一对应

### 3.2.2时间特性要求

* 响应时间：文章内容获取、单词释义、生词表内容获取等应在人的感觉和视觉事件范围内；
* 更新处理时间；文章数据在1-3天内有更新；
* 数据的转换和传送时间；词汇量测试结果的获取，处理后转化为图表的时间不宜过长。

### 3.2.3安全性需求

* 保证用户唯一识别标识在前后端传输中的安全性；
* 提供必要的冗余和备份措施。当系统发生故障时能够立即恢复，保证系统的可靠运行：系统备份、数据库备份、快速恢复。

### 3.2.4 软件质量属性

* 可靠性：词汇量测试准确度应尽量准确，不低于70%；当获取CNN网站失败后，有错误处理机制；
* 健壮性：选择适当方法获取CNN网站数据，保证出错概率低于一定值；
* 并发性：保证至少100个用户同时登录阅读，正常获取数据；
* 易用性：界面设计简洁明了，用适合的图像元素表示功能，使操作简便，易上手；
* 可扩展性：方便对该软件系统扩充新功能模块，便于版本的迭代更新。

### 3.2.5 业务规则

* 用户只有在进行词汇量测试之后，才可以为用户提供个性化推荐文章
* 用户只有看完一篇文章后，才会显示最近浏览
* 用户只有收藏文章后，才会出现收藏数据

### 3.3 数据要求

用户的个性化推荐文章、收藏文章、最近浏览文章、单词本都要与用户本身id具有关联性，并有一致性，数据完整。

# 4运行环境规定

## 4.1设备

a. 在微信web开发者工具中进行模拟器调试

b. 使用移动设备进行真机调试

c. 使用scrapy获取CNN在规定时间间隔内获取网站文章数据

## 4.2接口

* 查词接口

有道Restful Web API：获取单词的考试等级，发音与释义；

扇贝API：获取单词的双语例句

# 三、概要设计

## 1、总体设计

### 1.1 需求规定

功能性要求：构建一个完整的阅读小程序，实现阅读，推荐，背单词，查 询单词，收藏文章等一系列功能。

输入项目为用户的一系列操作请求

输出项目为针对不同请求的响应。

### 1.2 运行环境

* 安卓4.5及以上
* ios设备11.0及以上

### 1.3 基本设计概念和处理流程

系统整体架构如下。用户在微信小程序界面进行操作，前端处理用户各种操作请求，针对需要涉及到数据交互以及复杂操作的命令，则交由后端处理。后端对于前端的http请求，首先从映射的url列表中一一匹配，找到合适的则进行处理并返回处理结果。

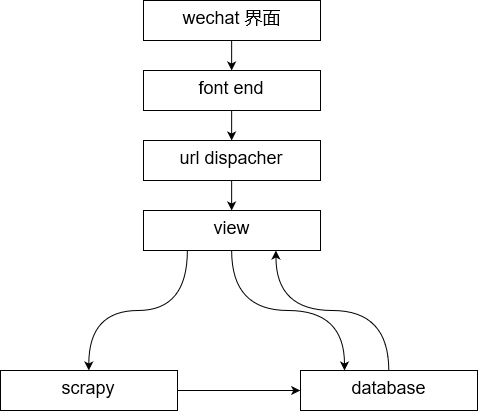


表 2系统架构

### 1.4 结构

小程序用户角色只有阅读者，无管理员等角色。功能主要分为四大模板，用户登陆小程序，创建身份，并获取唯一标识。阅读文章功能包含四个子系统，分别是浏览所有文章模板，查看最近浏览文章，收藏某文章以及获取个性化推荐的文章。测试词汇量下属两个子模板，分别是获取测试的词汇数据，以及查看历史测试结果。用户可以通过生词本背单词，并且对单词进行删除添加等操作。

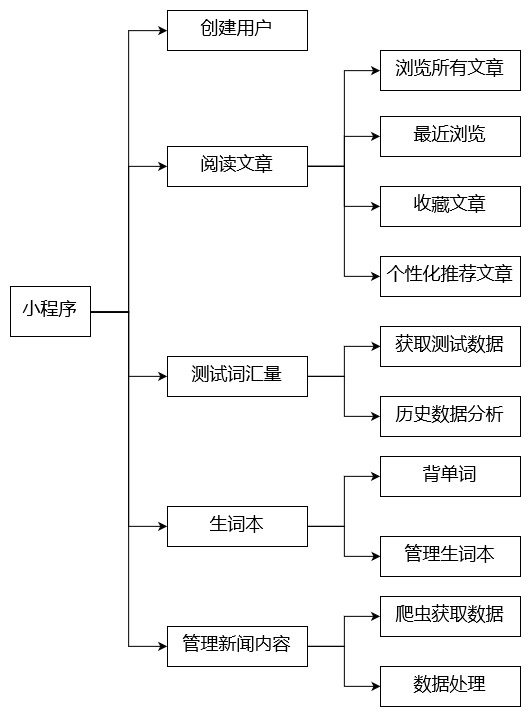


表 3系统功能点

### 1.5 人工处理过程

从新闻网站爬取数据以及将数据移植到服务器的数据库中需要手动操作。

## 2、接口设计

### 2.1 用户接口

用户接口主要功能为用户对文章，生词进行一系列相关处理，并且对个人的生词数据进行操作。

| **接口** | **功能** |
| --- | --- |
| 浏览推荐文章 | 根据用户词汇测试数据返回推荐的文章 |
| 登陆 | 获取用户登陆状态以及唯一标识 |
| 按分类浏览文章 | 获取某分类所有文章 |
| 管理生词本 | 删除/添加单词 |
| 背单词 | 获取生词本所有单词开始记忆 |
| 测试词汇量 | 获取待测试词汇，处理测试结果 |
| 收藏文章 | 收藏文章信息，可供再次查看 |
| 最近浏览 | 查看最近阅读的文章 |

表 4用户接口

### 2.2 外部接口

外部接口主要用于与数据库进行信息交互。

| **接口类型** | **接口** | **功能** |
| --- | --- | --- |
| 软件接口 | 与数据库接口 | 连接数据库并进行信息交互 |
| 服务器接口 | 微信服务器接口 | 从微信服务器解码code，获取用户openid |

表 5外部接口

### 2.3 内部接口

内部接口主要是系统内部功能，例如利用爬虫获取新闻资源，对文章内容进行一定的处理，不同模块之间的调用。

| **接口** | **功能** |
| --- | --- |
| 添加测试记录 | 获取用户测试词汇结果并存入数据库 |
| 查询测试记录 | 返回用户历史测试记录 |
| 添加单词 | 向服务器生词本添加单词 |
| 删除单词 | 删除生词本中指定单词 |
| 查询单词 | 返回所有生词 |
| 添加收藏记录 | 将收藏文章的信息录入数据库 |
| 生成测试词汇 | 随机生成用户测试词汇数据 |
| 录入文章信息 | 将服务器本地文章信息录入数据库 |
| 爬取新闻资源 | 调用scrapy框架获取新闻资源 |

表 6内部接口

## 3、运行设计

该部分主要描述针对系统施加不同外界运行控制时所引起的各种不同的运行模块组合。

### 3.1 运行模块组合

#### 3.1.1 阅读文章模块组合

* 用户浏览所有文章，点击查看文章，添加生词，添加收藏功能模块组合。
* 用户测试词汇，查看个性化推荐文章功能模块组合。
* 用户查看单词本，删除单词，背单词功能模块组合。
* 用户查看最近浏览文章，点击阅读，添加收藏功能模块组合
* 用户背单词，选择单词释义，查看测试结果功能模块组合
* 用户点击分类，阅读文章功能模块组合。

#### 3.1.2 处理新闻资源模块组合

* 后台运行爬虫程序，获取新闻资源，录入数据库功能模块组合
* 后台录入新闻资源，进行数据整理，查询数据功能模块组合
* 后台根据用户信息，获取数据库资源，进行数据比较，返回文章信息功能模块组合
* 后台根据文章图片地址，修改图片名，下载图片并修改数据库功能模块组合

#### 3.2 运行控制

#### 3.2.1 阅读文章模块组合

* 用户需要授权小程序获取用户id才能阅读文章
* 用户需要至少测试过一次词汇量才可以获取个性化推荐文章

### 3.3 运行时间

#### 3.3.1 阅读模块运行时间

系统主要的运行时间消耗在对数据库进行增删改查的操作以及前后端交互过程中大量数据传输的时间。当后台文章未及时清理，导致文章条目不断变多，会致使用户浏览全部文章时，后台一次性返回的数据量过多，加载缓慢。

#### 3.3.2 处理新闻资源运行时间

主要时间消耗在获取文章资源，由于外文网站服务器在国外，相对时延较大，并且在爬取的过程中，框架本身的特性也会造成一定的时间消耗。所以时间的开销不能忽略。

但是好在文章爬取一天仅做一次，并且并非实时获取。所以对于用户体验影响不大。

#### 3.3.3 耗时少的功能模块：

直接在前端实现的所有功能都耗时较少。例如，对用户测试词汇的数据进行分析，按照分类展示文章等等。

#### 3.3.4 耗时大的功能模块：

与后端进行大量数据传输的功能模块，以及与数据库进行交互的模块。

## 4、系统数据结构设计

### 4.1 逻辑结构设计要点

* 系统涉及到的实体主要有如下几种，文章，用户，词汇测试记录，收藏文章信息。

用户（编号，小程序唯一标识，生词本）

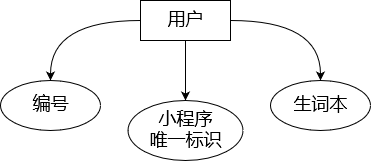
* 

表 7用户逻辑设计图

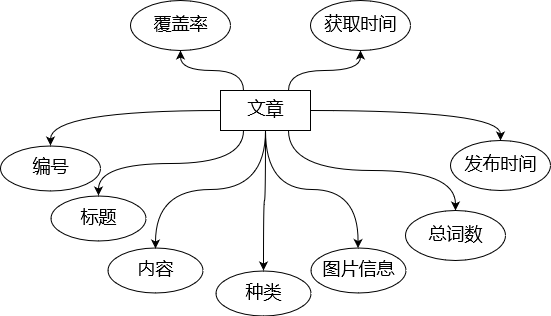
* 文章（编号、标题、内容、作者、种类、图片信息、总词数、发布时间、覆盖率、获取时间）
* 

表 8文章逻辑设计

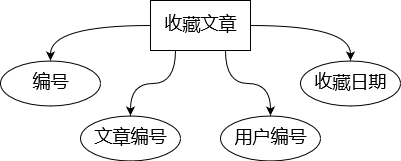
* 收藏文章（编号、文章编号、用户编号、收藏日期）
* 

表 9收藏文章逻辑设计

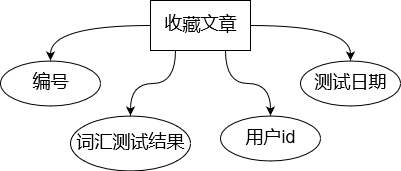
* 词汇测试记录（编号、词汇测试结果、用户id、测试日期）
* 

表 10测试结果逻辑设计

* ER图
* 

表 11 ER图

### 4.2 物理结构设计要点

用户编号是用户唯一标识加密过后的结果，为了保证传输过程的安全性，在一般传输过程中都使用用户编号进行操作。生词本记录了用户生词本在服务器中的存储位置。

| **字段名** | **类型** | **宽度** | **主键/索引** | **约束** | **是否可为null** |
| --- | --- | --- | --- | --- | --- |
| 用户编号 | varchar | 500 | 主键 |  | 否 |
| 用户唯一标识 | varchar | 500 |  |  | 否 |
| 生词本 | varchar | 500 |  |  | 否 |

表 12 用户物理结构设计

文章通过文章编号唯一标识，该属性是自增的。考虑到文章内容有长有短，内容过长在数据库会占据较大空间，因此文章内容存储的是文章存储在服务器上的位置，是一个json文件。图片信息也是如此。只有一点特别的就是图片存储在服务器文件的指定文件夹下，可以通过url直接访问。

| **字段名** | **类型** | **宽度** | **主键/索引** | **约束** | **是否可为null** |
| --- | --- | --- | --- | --- | --- |
| 文章编号 | integer |  | 主键 |  | 否 |
| 标题 | varchar | 500 |  |  | 否 |
| 文章内容 | varchar | 500 |  |  | 否 |
| 作者 | varchar | 500 |  |  | 否 |
| 类型 | varchar | 500 |  |  | 否 |
| 图片信息 | varchar | 500 |  |  | 否 |
| 总词数 | integer |  |  |  | 否 |
| 出版时间 | varchar | 500 |  |  | 否 |
| 覆盖率 | varchar | 500 |  |  | 否 |
| 获取时间 | date |  |  |  | 否 |

表 13 文章表物理设计

测试内容由编号唯一标识，该属性自增。外键为用户编号。一个用户可以有零到多条测试记录。

| **字段名** | **类型** | **宽度** | **主键/索引** | **约束** | **是否可为null** |
| --- | --- | --- | --- | --- | --- |
| 测试编号 | integer |  | 主键 |  | 否 |
| 覆盖率 | varchar | 500 |  |  | 否 |
| 用户编号 | varchar | 500 |  |  | 否 |
| 日期 | date |  |  |  | 否 |

表 14 测试结果物理设计

收藏条目由编号唯一标识，属性自增。外键为文章编号与用户编号。

| **字段名** | **类型** | **宽度** | **主键/索引** | **约束** | **是否可为null** |
| --- | --- | --- | --- | --- | --- |
| 收藏编号 | integer |  | 主键 |  | 否 |
| 文章编号 | integer |  |  |  | 否 |
| 用户编号 | varchar | 500 |  |  | 否 |
| 日期 | date |  |  |  | 否 |

表 15 收藏条目物理设计

### 4.3 数据结构与程序的关系

后端在获取用户测试结果，记录用户操作信息时需要对数据库中的表进行修改，即增删改查操作。

物理数据结构主要用于各模块之间函数的信息传递。接口传递的信息是数据结构封装了的数据，以参数或者返回值的形式在各模块之间传输。

## 5、系统出错处理设计

### 5.1 出错信息

| **出错类型** | **出错形式** | **处理办法** |
| --- | --- | --- |
| 404 | 从后端获取数据，显示内容不存在 | 检查发送请求的url地址以及http请求的格式，逐项排查。 |
| 500 | 从后端获取数据，显示服务器内部错误 | 检查服务器部分代码，重点查看负责该部分功能模块代码的业务逻辑以及返回数据的格式。 |
| 登陆 | 无法获取小程序用户的唯一标识 | 排查后端处理code部分关于小程序的开发者信息是否正确以及检查用户是否过长时间未登陆导致信息失效。 |
| 无法获取文章信息 | 用户查看收藏文章，无法正确显示 | 查看后端是否对数据库进行了误操作，删除了文章内容和记录。 |
| 无法爬取文章资源 | 启动爬虫程序，无法获取任何文章信息 | 可能是访问过于频繁被网站禁止访问了，可以通过更换ip池的ip来访问。 |

表 16 出错信息

### 5.2 补救措施

通过对以上的错误信息进行分析，我们可以针对系统进行一些优化：

1、数据库数据交互错误：这类错误主要是因为开发人员对数据库进行了误操作或者是功能模块内部对数据进行了操作但是开发人员没有意识到，对此， 我们应该定期对数据库进行备份，对进行了数据库相关操作的模块进行严谨的测试。

2、请求错误：主要是前后端的开发人员之间对于接口的定义以及数据形式没有协商好，需要细化接口设计。

3、其他操作错误：针对用户因为不熟悉系统而造成的错误，我们可以通过撰写详尽的使用手册来规避。

4、其他不可知的程序错误：比如，原本网站突然不服务了，或者被禁掉了。这种不可预知的错误可以通过顶起维护后端来规避，或者在程序中设置提示，在发生异常的时候，设置自动消息提示。

### 5.3 系统维护设计

随着文章爬取数量的不断增加，数据库势必会越来越庞大，在爬取的过程中，可能会产生一些无效文章信息，以及脏数据。人工检查成本比较高，因此采取后端维护人员定期对数据进行维护备份的方案。

前端由开发人员不断根据技术更新进行升级完善。

# 四、详细设计

## 1、程序系统的结构

针对概要设计部分的说明，针对每个模块进一步细化，明确内部结构和实现细节。方便后期测试时进行调试和复查。

### 1.1 创建用户模块设计说明

#### 1.1.1 程序描述

用户初次登陆小程序需要获取唯一标识以记录登陆状态，以及在数据库中生成条目用以记录用户后续在小程序中一系列操作。

#### 1.1.2 功能

由于微信对小程序开发的安全性有较高要求，因此用户信息的解码无法在前端直接进行，只能在前端获取code，然后在后端，附带开发者的基本信息调用微信的api接口来获取解码信息。

用户的基本标识不能直接明文传输，因此加密以后作为数据库用户表的主键并传回前端。此后涉及用户的数据操作，前端都需要附带用户的唯一标识。

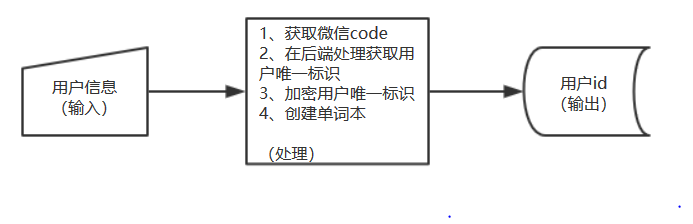


表 17创建用户IPO图

#### 1.1.3 性能

时间：由于用户登陆授权后，需要及时得到唯一标识并允许用户进行后续操作，因此对于服务器获取解码信息的响应要求较高。

#### 1.1.4 输入项

| **名称** | **标识** | **类型** | **输入方式** |
| --- | --- | --- | --- |
| 用户编码 | code | json | 附在http请求中 |

#### 1.1.5 输出项

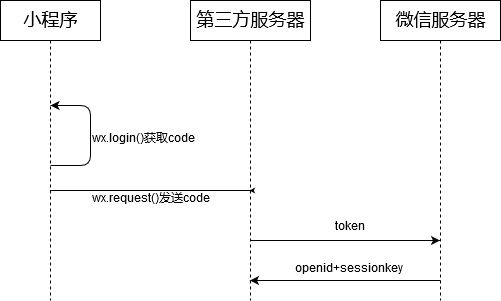
| **名称** | **标识** | **类型** | **输出方式** |
| --- | --- | --- | --- |
| **用户id** | **uid** | **string** | **附在httpResponse中** |

#### 1.1.6 算法

使用了python自带的md5加密算法对所有用户的唯一标识进行加密。

#### 1.1.7 流程逻辑

整个获取流程如下图所示。



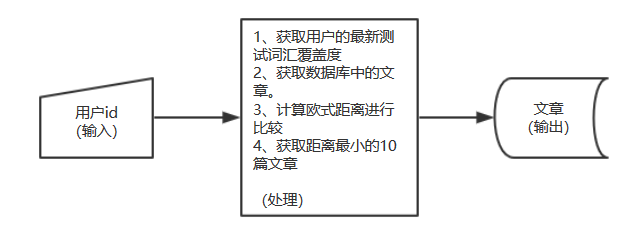
### 1.2 推荐文章模块设计说明

#### 1.2.1 程序描述

该模块主要实现了针对用户的词汇量以及对不同词库（四六级等）词汇掌握情况推荐最适合阅读的文章的功能。

#### 1.2.2 功能

用户获取推荐文章的前提是已经做过词汇测试，如果做过多轮词汇测试则取最新的结果。而文章根据后台管理者的需求，可以设置只取近3天（或者任意）的文章进行推荐。



#### 1.2.3 性能

时间特性：用户做完词汇测试后应当立即就能看到推荐文章，等待时间过长会影响用户体验，因此要求后端进行数据处理时，不能消耗过长时间，如果数据库中的文章条目过多，后台管理员应该设计一个取数据范围。

#### 1.2.4 输入项

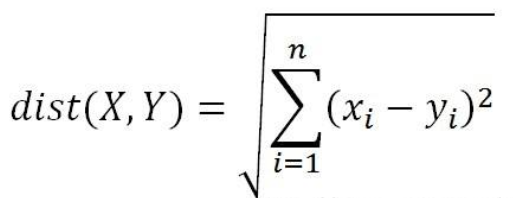
| **名称** | **标识** | **类型** | **输入方式** |
| --- | --- | --- | --- |
| 用户id | uid | string | 附在http请求中 |

#### 1.2.5 输出项

| **名称** | **标识** | **类型** | **输出方式** |
| --- | --- | --- | --- |
| 推荐文章 | articles | dict | httpresponse中 |

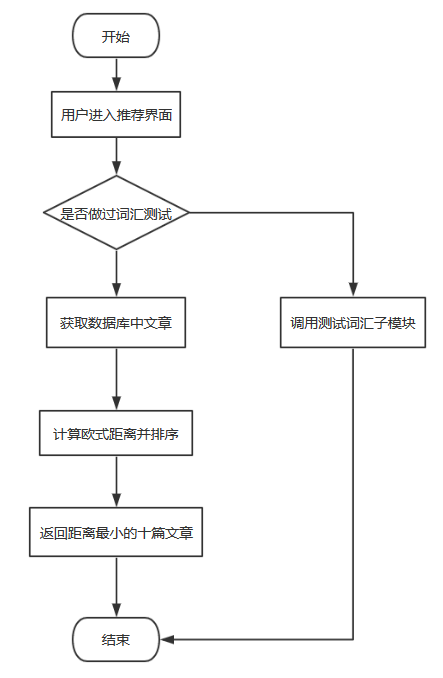
#### 1.2.6 算法

主要用到了欧式距离来计算用户词汇覆盖度和文章词汇覆盖度之间近似程度。



#### 1.2.7 流程逻辑

当用户进入推荐模块才会判断是否做过词汇测试，若用户刚进入小程序，并且没有用到推荐功能，则不会要求进行词汇测试。



#### 1.2.8 接口

若用户未做过词汇测试并想使用推荐功能，则调用词汇测试模块。

### 1.3 词汇测试模块设计说明

#### 1.3.1 程序描述

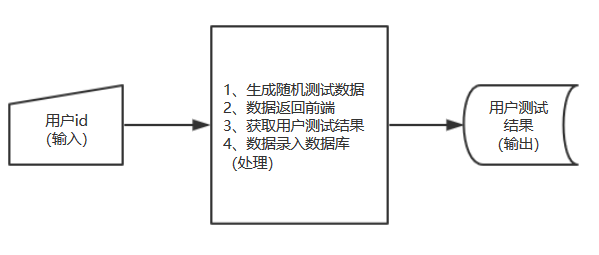
不同的用户掌握的词汇不尽相同，有的用户可能掌握的四级词汇较多，有的可能掌握的六级词汇较多， 该模块主要用于对用户的词汇覆盖程度做一定的度量（度量的标准有四级，六级，托福，雅思等），得出的结果是系统个性化推荐文章的重要依据。

#### 1.3.2 功能

首先获取用户id，因为测试的结果需要和用户绑定。用户使用该功能后，后端在每个词库中分别随机化抽取词汇，返回前端。

用户对每个单词进行判断，如果认识则选择“认识”，不认识则选择“不认识”，最后将认识的词汇数/该词库抽取的总词数，该比例可以指代用户对于该词库的掌握程度。该数据由前端统计后返回后端并存入数据库。

同时，用户在测试完以后可以查看不认识的词汇进行复习巩固。



#### 1.3.3 性能

精度：用户词汇掌握程度的结果取浮点数，小数点后两位即可。

灵活性：由于随机生成的测试词汇比较多，因此，应当允许用户随时停止测试，并对未测试的数 据做出随机化处理。

#### 1.3.4 输入项

| **名称** | **标识** | **数据类型** | **输入方式** |
| --- | --- | --- | --- |
| 用户id | uid | string | http请求 |

#### 1.3.5 输出项

| **名称** | **标识** | **数据类型** | **输出方式** |
| --- | --- | --- | --- |
| 词汇覆盖度 | cover\_rate\_list | list | http请求 |
| 测试词汇 | test\_words | dict | httpResponse |

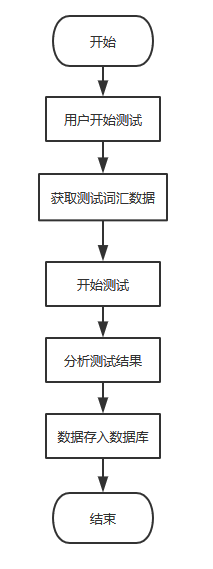
#### 1.3.6 算法

获取的词库以及词库词汇数量如下：

| **词库** | **cet4** | **cet6** | **高中** | **GRE** | **IELTS** | **考研** | **托福** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 词数 | 4616 | 2088 | 3933 | 6495 | 6555 | 5325 | 5395 |

按照一定的比例从每个词库中随机抽取单词（该比例可有后台开发者自定），若从六级词库中抽取10个词，用户进行测试后，认识6个词，那么可以估计用户对于四级词汇的掌握程度为0.6。其他的词库掌握情况以此类推。

#### 1.3.7 流程逻辑



#### 1.3.8尚未解决的问题

有的词库中抽取的词汇可能会和其他词库重合（但是概览较低），因此最保险的做法是在随机抽取词汇时进行判定，若抽到了重复的数据则重新抽取。

### 1.4 获取新闻内容模块设计说明

#### 1.4.1程序描述

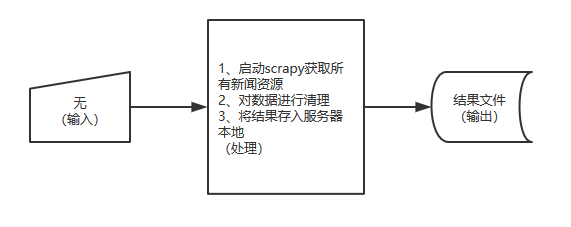
系统最重要的部分是对于数据的获取，我们使用scrapy框架来获取新闻内容以及其他相关信息。scrapy框架功能较为完整，只需要明确需要获取的页面元素即可自动化的爬取内容。

调用该程序后，模块会自动爬取当日所有条目下的新闻文章，并对数据做一定的处理，最后形成json文件存入本地。

工作人员手工将获取的新闻数据再存入数据库。

#### 1.4.2功能

获取了所有新闻数据后，需要清洗数据，将文本内容从string格式变成以单独词为单位的list列表，并且获得去重的文章单词信息，再将去重单词列表与词库相对比，查看文章内容针对每个词库的覆盖情况。



#### 1.4.3性能

时间特性：由于数据每日获取一次，对于获取的时间没有要求

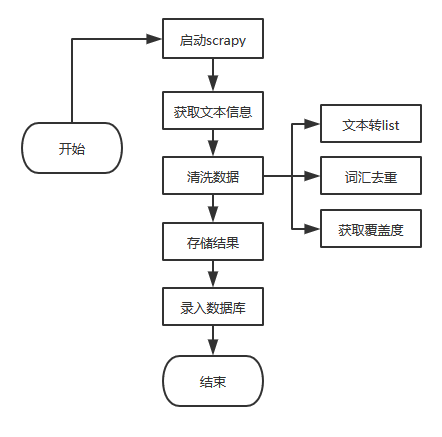
灵活性：模块应当能够处理一定的错误，例如在请求过于频繁，ip地址被禁止以后能够切换ip。

精度：精度要求不高，但是要确保文章内容是准确的。

#### 1.4.4 输出项

| **名称** | **标识** | **数据类型** | **输出形式** |
| --- | --- | --- | --- |
| 文章信息汇总 | news | dict | 本地文件 |
| 文章内容 | content | list | 本地文件 |

#### 1.4.5 流程逻辑

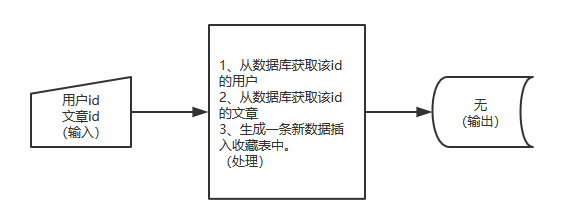


### 1.5收藏文章模块设计说明

#### 1.5.1程序描述

该模块主要实现了用户收藏文章的功能，用户与文章是一对多的关系，一个用户可以收藏多篇文章，并且可以在”我的收藏“模块查看历史收藏。

#### 1.5.2功能



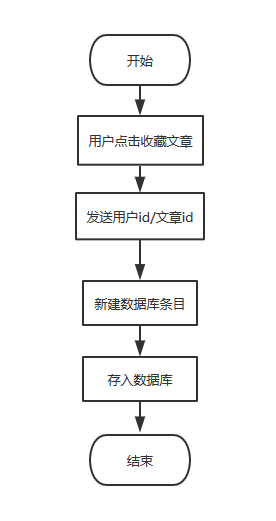
#### 1.5.3性能

精度：需要准确的记录用户收藏的文章信息以及收藏的日期。

#### 1.5.4输入项

| **名称** | **标识** | **类型** | **输入方式** |
| --- | --- | --- | --- |
| 文章id | article\_id | integer | http请求 |
| 用户id | user\_id | string | http请求 |

#### 1.5.5流程逻辑



### 1.6 最近浏览模块设计说明

#### 1.6.1 程序描述

该模块主要实现存储用户最近阅读的10篇文章记录的功能。

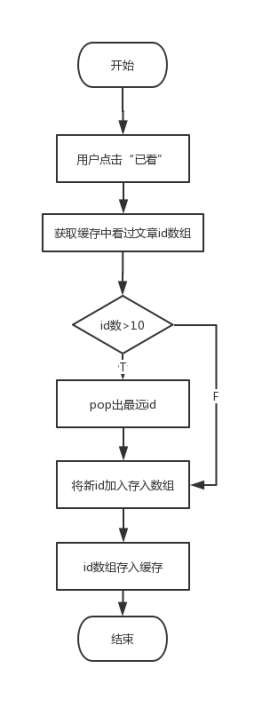
#### 1.6.2 功能

由于最近浏览不存储过多数据，将用户每篇阅读完成的文章主键存储在数组中，再将该数组同步保存到小程序缓存中：wx.setStorageSync('seenList', temp)；查看浏览记录，则直接从缓存中同步获取wx.getStorageSync('seenList')

#### 1.6.3 输出项

| **名称** | **标识** | **数据类型** | **输出方式** |
| --- | --- | --- | --- |
| 最近阅读文章列表 | article\_list | list | 从缓存中取 |

#### 1.6.4 流程逻辑

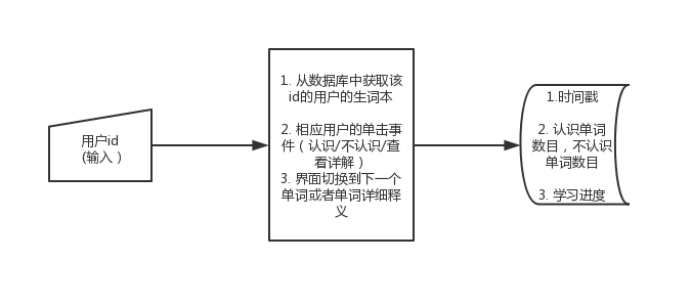


### 1.7 背单词模块设计说明

#### 1.7.1 程序描述

该模块主要实现了用户背单词的功能，每一个用户拥有的生词本存放在后台的服务器中。背单词的过程中，程序会随机以卡片的形式展示生词本中的莫一个单词，用户对当前卡片所展示的单词有两个选项，认识和不认识。如果用户不认识该单词可以单击单词卡片上的“查看详细释义”，之后便可以看到单词的详细解释，包括双语例句，发音，以及词性。不认识的单词会多出现几遍。

#### 1.7.2 功能



#### 1.7.3 输入项

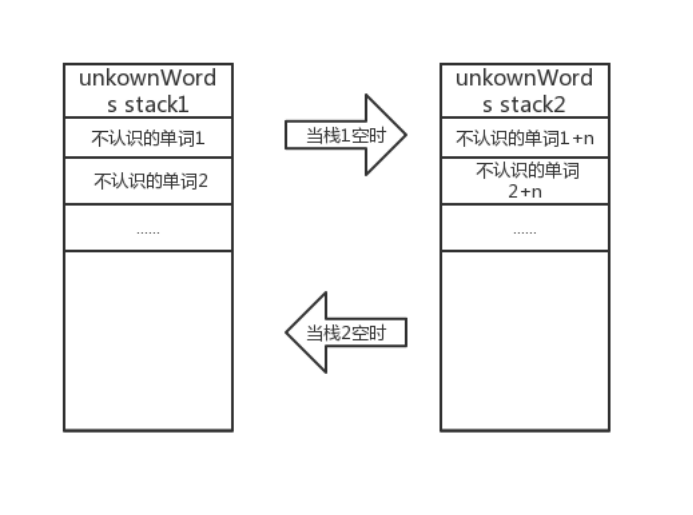
| **名称** | **标识** | **数据类型** | **输入方式** |
| --- | --- | --- | --- |
| 用户id | uid | string | http请求 |
| 认识 | Known | event | 用户触发认识按钮 |
| 不认识 | Unkown | event | 用户触发不认识按钮 |

#### 1.7.4 输出项

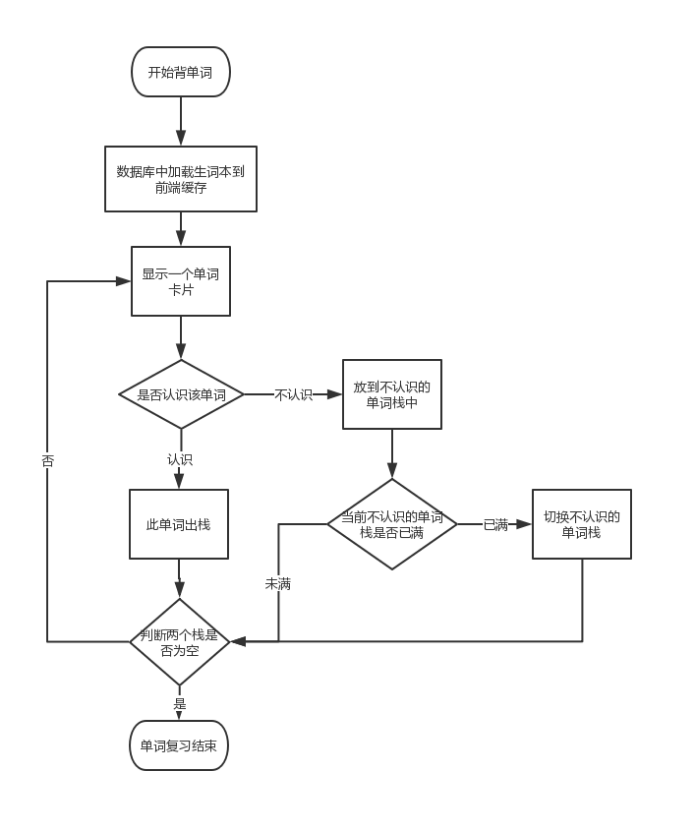
| **名称** | **标识** | **数据类型** | **输出方式** |
| --- | --- | --- | --- |
| 认识单词数 | familiarWords | int | 小程序缓存 |
| 不认识单词数 | UnfamiliarWords | int | 小程序缓存 |
| 背单词日期 | currentTime | Date | 小程序缓存 |

#### 1.7.5 算法

背单词的过程中，每次用户单击不认识此单词的时候就将此不认识的单词放到一个栈中，当当前的栈空的时候开始背另一个栈，两个栈来回倒来实现背单词的功能，直到用户将单词背完。



#### 1.7.6 流程逻辑

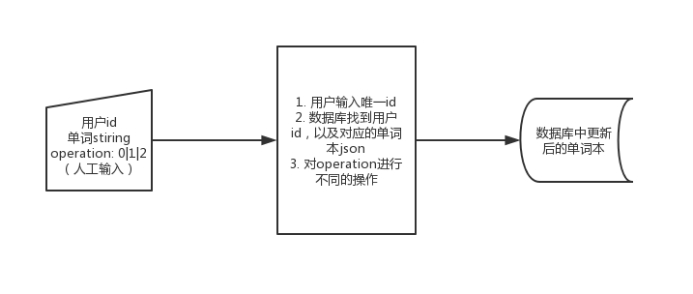


### 1.8 生词本模块说明分析

#### 1.8.1 程序描述

用户在阅读文章的过程中，可以添加单词到单词本中，单词本中的每一个单词在数据库中的存放形式是一个json的列表，列表里的每一个元素都是一个字典。字典数据的健是单词，值是被添加的次数。用户可以对单词本进行，增和删两种操作。

#### 1.8.2 功能



#### 1.8.3 输入项

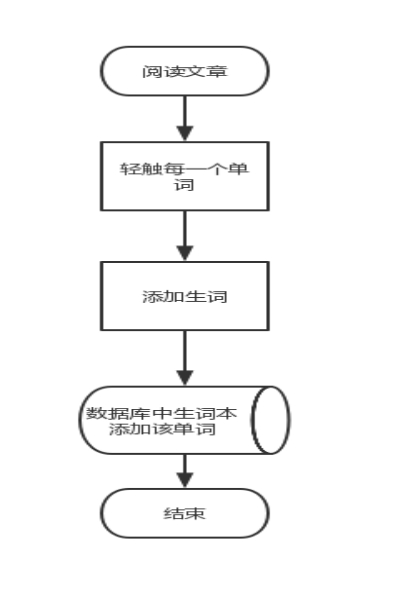
| **名称** | **标识符** | **数据类型** | **输入方式** |
| --- | --- | --- | --- |
| 0 | operation | int | http |
| 1 | operation | int | http |
| 2 | operation | int | http |

#### 1.8.4 输出项

| **名称** | **标识符** | **数据类型** | **输出方式** |
| --- | --- | --- | --- |
| 0 | operation | list | http |

#### 1.8.5 逻辑流程

添加生词业务逻辑



# 五、测试计划

# 1引言

## 1.1编写目的

轻易阅读小程序大体功能已经实现，但是系统本身还存在一些问题，包括功能上的缺失以及信息提供的不足。希望通过本系统的测试，除了在发现更多的系统缺陷外，同时建立起一套较完整的测试过程规范和一套较完整的测试用例库。

## 1.2背景

轻易英语阅读小程序是一个轻型英语阅读平台。市场已经有一些英语阅读APP，例如“扇贝阅读”，但基于小程序平台的却很匮乏。本系统加入亮点：根据用户的词汇量测试结果，进行个性化推荐文章，并加入学习英语的功能，例如生词本，取词翻译等。

|  |  |  |
| --- | --- | --- |
|  | 手工测试 | 职责 |
| 测试人员 | 陈彩婷 | 单词本、取词翻译、文章管理 |
| 测试人员 | 汪伦 | 个性化推荐文章、分类文章、词汇量测试、 |

|  |  |  |
| --- | --- | --- |
| **术语或缩写词** | **英文解释** | **中文解释** |
| GRE |  | 美国研究生入学考试 |
| API |  | 软件应用程序编程接口 |

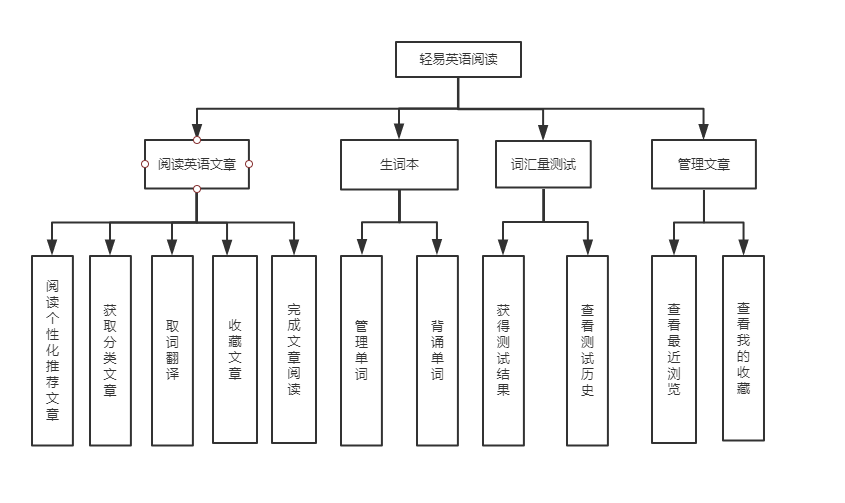
## 1.4参考资料

|  |  |  |
| --- | --- | --- |
| **资料名称** | **作者** | **说明** |
| 《轻易英语阅读项目开发计划书》 | 邓晏湘 | 项目开发计划 |
| 《《轻易英语阅读软件需求规格说明书》 | 陈彩婷 | 项目需求说明书 |
| 《软件工程导论》 |  | 软件工程课程教材 |
| 《软件测试技术》 | 清华大学出版社 | 软件测试课程教材 |

# 2计划

## 2.1软件说明

* 功能层次图



* **质量指标**
* 词汇量测试：随机获取各层次（CET-4、CET-6、考研等）测试词汇列表，用户进行测试；测试结果判断应在2s内；获得具体测试结果（认识单词列表、遇到的各层次单词分类）；
* 个性化推荐文章：后台根据词汇量测试结果对具体用户进行个性化推荐，推荐文章数量为10篇以内；推荐文章类别与用户水平不存在太大出入；对为进行测试的用户进行引导；
* 分类文章：显示所有文章，分类显示，大致分为（Healthy、World、Home、Sport几大类），用户可根据分类选取阅读文章；
* 取词翻译：对用户阅读文章过程中的单词进行随时点击，随时得到翻译结果的功能；单词翻译准确无误，大部分单词可得到释义；
* 文章管理：“看过”文章加入到“最近浏览”列表中，“收藏”文章添加到“我的收藏”列表中，可正确获得列表内容；定期清理“最近浏览”内容；对收藏文章可进行删除操作；
* 生词本：用户在遇到生词时，将该单词添加到生词本中，在生词本中查看单词列表并进行背诵，删除已熟悉的单词；

## 2.2测试内容

|  |  |  |  |
| --- | --- | --- | --- |
| **测试类型** | **是否进行测试** | **测试优先级** | **说明** |
| 单元测试 | 是 | P01 | 模块功能点、接口、数据结构 |
| 功能测试 | 是 | P01 | 分模块进行 |
| 集成测试 | 是 | P01 | 将各个模块整合 |
| 界面测试 | 是 | P02 | 简洁美观大方 |
| 易用性测试 | 是 | P02 | 操作方便、易学 |
| 兼容性测试 | 否 | P03 |  |
| 性能测试 | 是 | P01 | 各项响应时间最多5s |
| 配置测试 | 否 | P03 |  |
| 安装和卸载测试 | 否 | P04 |  |
| 回归测试 | 是 | P02 | 在开发过程中反复测试功能 |

## 2.3进度安排

|  |  |  |  |
| --- | --- | --- | --- |
| **项目里程碑** | **开始时间** | **结束时间** | **输出要求/备注** |
| 测试规划 | 2019年6月2日 | 2019年6月2日 | 初步测试计划书 |
| 测试设计 | 2019年6月3日 | 2019年6月4日 | 测试计划书 |
| 测试设计实施 | 2019年6月5日 | 2019年6月6日 | 测试用例 |
| 测试执行 | 2019年6月7日 | 2019年6月10日 | 测试结果 |
| 测试总结 | 2019年6月10日 | 2019年6月11日 | 测试分析报告 |

# 3测试设计说明

## 3.1 功能测试

### 3.1.1词汇量测试功能模块

词汇量测试要求响应速度在2s以内，可以准确判断用户的词汇量水平。

#### 3.1.1.1控制

用户选择答案，系统对其选择的答案进行判断对错，并且存储起来。

#### 3.1.1.2输入

以不同的词汇量标准来对该模块进行测试，例如高中水平，则高难度单词选错，其他大致正确；或者GRE水平，可以达到全对。

#### 3.1.1.3输出

* + 对应一个单词的选择，可以提示用户答案是否正确；
  + 结束测试后，输出准确率分布
  + 结束测试后，输出词汇量覆盖，以及各个水平（CET-4、CET-6……）在测试中出现的单词分组

#### 3.1.1.4测试用例设计

|  |
| --- |
| **测试用例 1: 词汇量测试之性能测试** |
| 综述:   * 目的：测试词汇量测试中的相应速度 * 条件：用户登陆小程序，并且能够联网 |
| 步骤:  1. 进入小程序词汇量测试模块  2. 单击每一个单词选项，查看出现下一个单词的时间  3. 重复步骤二10次，记录下每次出现下一个单词的时间  4. 计算平均值 |
| 期望输出:  1. 单词相应的时间越短越好 |
| 实际输出:  1. |

|  |
| --- |
| **测试用例2: 词汇量测试之准确度测试** |
| 综述:   * 目的：判断词汇量是否能正确的对用户选正确的单词做出纪录，对用户选错的单词做出纪录 * 条件：用户登陆小程序，并且能够联网 |
| 步骤:  1. 进入小程序词汇量测试模块  2. 对出现的单词的中文意思选择对应的英文解释  3. 记录下每次英文解释选择是否正确  4. 重复步骤3  5. 最后查看程序所纪录的准确度 |
| 期望输出:  1. 词汇量测试拥有较高的准确度，不会误判单词 |
| 实际输出:  1. |

### 3.1.2个性化推荐文章功能模块

根据用户的词汇量测试水平，推荐最适合用户的10篇以内的不同的英语文章。

#### 3.1.2.1 控制

用户进行词汇量测试以后，后台将结果保存到数据库中，并且根据该结果为判断依据，从获取的所有文章中选取最适合用户的少量文章，将这些文章信息也存到对应数据库中。用户可以在界面中查看这些文章。

#### 3.1.2.2 输入

模拟不同的词汇测试量水平，例如高中水平，则高难度单词选错，其他大致正确；或者GRE水平，可以达到全对。

#### 3.1.2.3 输出

10篇以内的个性化推荐阅读文章。

#### 3.1.2.4 测试用例设计

|  |
| --- |
| **测试用例3: 文章个性化推荐** |
| 综述:   * 目的：查看是否能给用户推荐合适难度的文章 * 条件：用户能正确登陆小程序，并且连接到互联网 |
| 步骤:  1. 先进行用户的词汇量测试  2. 完成词汇量测试之后会得到一个词汇量的分布表  3. 之后返回文章阅读界面  4. 刷新  5. 查看推荐的文章的难度和用户词汇量难度的相符程度  6. 重复2-5步骤多次 |
| 期望输出:  1. 推荐的文章的难度和用户词汇量测试结果一致的程度较高 |
| 实际输出:  1. |

### 3.1.3 分类文章功能模块

该模块显示所有的英语文章内容，并根据类别分类显示（Healthy、Sports、Home……）

#### 3.1.3.1 控制

按照分类从CNN网站上爬取近期时事文章，对文章进行错误判断，筛选出可以正常阅读的所有文章列表；并进行等级评估，得出适宜的难度水平。将文章列表及相关信息存储在数据库，在界面上可以查看所有文章。

#### 3.1.3.2 输入

无

#### 3.1.3.3 输出

分类处理后的文章列表。

#### 3.1.3.4 测试用例设计

|  |
| --- |
| **测试用例4: 文章内容的准确度** |
| 综述:   * 目的：测试获取的文章列表是否正确、完整 * 条件：用户登陆小程序，在后端验证成功 |
| 步骤:  1. 用户登陆小程序  2. 翻阅获取文章列表  3. 在各个类别中随机抽取5篇以上的文章  4. 查看文章内容是否正确显示  5. 记录准确度 |
| 期望输出:  1. 一个较高的准确度 |
| 实际输出:  1. |

|  |
| --- |
| **测试用例5: 文章显示的正确性** |
| 综述:   * 目的：测试小程序能否正确显示文章内容 * 条件：用户登陆小程序，在后端验证成功 |
| 步骤:  1. 用户登陆小程序  2. 在每个分类中随机抽取5篇以上的文章  3. 阅读文章根据判断界面是否清晰  4. 判断句子是否通顺  5. 判断内容是否缺失  6. 判断内容与标题是否匹配  7. 计算每篇文章的正确率 |
| 期望输出:  1. 一个较高的正确率 |
| 实际输出:  1. |

|  |
| --- |
| **测试用例6: 文章难度划分的准确度** |
| 综述:   * 目的：测试小程序能否根据文章的词汇覆盖来对文章进行难度划分 * 条件：用户登陆小程序，在后端验证成功 |
| 步骤:  1. 用户登陆小程序  2. 在各个难度等级中随机抽取至少5篇以上的文章  3. 对选择的文章进行单独处理，判断出现的词在不同难度词库中的覆盖率  4. 以覆盖率为依据判断难度是否与标签不相同  5. 记录难度划分匹配的准确度 |
| 期望输出:  1. 一个较高的准确度 |
| 实际输出:  1. |

|  |
| --- |
| **测试用例7: 文章分类的识别度** |
| 综述:   * 目的：测试小程序能否根据文章的主旨内容来对文章进行分类 * 条件：用户登陆小程序，在后端验证成功 |
| 步骤:  1. 用户登陆小程序  2. 随机抽取其中一篇文章，查看文章的分类  3. 阅读文章根据经验判断文章的主旨内容是否和文章在后端的分类是否准确  4. 重复步骤三十次，记录下每次的验证结果  5. 计算分类准确的百分数，以此百分数作为识别度 |
| 期望输出:  1. 一个较高的识别度 |
| 实际输出:  1. |

|  |
| --- |
| **测试用例8: 文章获取的响应时间** |
| 综述:   * 目的：测试小程序能否在较短时间内获取大量文章 * 条件：用户登陆小程序，在后端验证成功，可获取文章内容 |
| 步骤:  1. 用户登陆小程序  2. 记录界面加载出所有文章列表的时间  3. 重复步骤1至少5次  4. 计算平均时间 |
| 期望输出:  1. 小程序能否在3s内获取文章 |
| 实际输出:  1. |

### 3.1.4 取词功能翻译

用户在阅读文章的过程中，对遇到的所有单词可以对其进行即时翻译，可选择查看具体释义。

#### 3.1.4.1 控制

在用户选择完具体单词后，调用有道翻译API得到该英语单词的释义，包括中文释义、音标、出现词库，呈现在用户面前；具体释义中包括例句

#### 3.1.4.2 输入

用户选择指定单词。

#### 3.1.4.3 输出

单词的中文释义，覆盖词库，音标。

#### 3.1.4.4 测试用例设计

|  |
| --- |
| **测试用例9: 取词翻译的准确度** |
| 综述:   * 目的：测试能否对阅读过程中出现单词获取正确释义 * 条件：用户登陆小程序，在后端验证成功，获取文章内容 |
| 步骤:  1. 用户登陆小程序  2. 随机抽取其中一篇文章  3. 随机选择至少十个以上单词，包括常见单词、较偏单词、词组  4. 判断释义是否与选择的单词一一对应  5. 重复步骤二十次，记录下每次的验证结果  6. 计算准确度 |
| 期望输出:  1. 一个较高的识别度 |
| 实际输出:  1. |

|  |
| --- |
| **测试用例10: 取词翻译的响应时间** |
| 综述:   * 目的：测试取词翻译的响应时间 * 条件：用户能正确登陆小程序，并且连接到互联网，获取文章内容 |
| 步骤:  1. 用户登陆小程序  2. 随机抽取其中一篇文章  3. 随机选择至少十个以上单词，包括常见单词、较偏单词、词组  4. 记录每次选择后释义出现的时间  5. 重复步骤二5次  6. 计算取词翻译的平均响应时间 |
| 期望输出:  1. 取词翻译可以做到即时获得释义的效果，平均响应时间不大于1s |
| 实际输出:  1. |

|  |
| --- |
| **测试用例 11: 取词翻译之具体释义的准确度** |
| 综述:   * 目的：测试是否能获取对应单词的具体释义 * 条件：用户登陆小程序，并且能够联网，获取文章内容，进行取词翻译 |
| 步骤:  1. 用户登陆小程序  2. 随机抽取其中一篇文章  3. 随机选择至少十个以上单词，包括常见单词、较偏单词、词组  4. 点击选择“查看详解”  5. 查看例句是否对应  6. 查看中文释义是否对应  7. 重复步骤二5次  8. 计算具体释义的准确度 |
| 期望输出:  1. 一个较高的准确度 |
| 实际输出:  1. |

### 3.1.5 文章管理功能模块

#### 3.1.5.1 控制

用户可对文章进行收藏，将文章id存储在列表中，并将列表与用户关联起来，存储在数据库中，显示收藏文章则从数据库中获取该列表。

用户可获取最近浏览文章，将文章id存储在列表中，并将列表存储在缓存中，显示最近浏览文章则从缓存中获取该列表。

#### 3.1.5.2 输入

用户点击“收藏”按钮；“看过”按钮，依次获取文章的id值。

#### 3.1.5.3 输出

* 操作成功提示
* 在对应列表中出现文章

#### 3.1.5.4 测试用例设计

|  |
| --- |
| **测试用例12: 文章管理之收藏文章准确度** |
| 综述:   * 目的：测试用户能否正确进行文章收藏 * 条件：用户登陆小程序，并且能够联网，获得文章内容 |
| 步骤:  1. 随机选取至少10篇以上文章  2. 收藏这些文章  3. 进入查看收藏文章界面，查看是否出现所有刚刚收藏过的文章  4. 计算准确度 |
| 期望输出:  1. 所有收藏过的文章都正确保存 |
| 实际输出:  1. |

|  |
| --- |
| **测试用例13: 文章管理之收藏文章性能测试** |
| 综述:   * 目的：测试收藏的反应时间 * 条件：用户登陆小程序，并且能够联网 |
| 步骤:  1. 随机选取一篇文章  2. 收藏该文章  3. 记录显示收藏成功的提示出现的时间  4. 立即进入查看收藏文章界面  5. 对照是否出现刚刚收藏的文章  6. 计算平均时间 |
| 期望输出:  1. 成功收藏文章响应时间小于1s |
| 实际输出:  1. |

|  |
| --- |
| **测试用例14: 文章管理之获取收藏文章性能测试** |
| 综述:   * 目的：测试获取所有收藏文章的响应时间 * 条件：成功获取文章内容，用户存在收藏记录 |
| 步骤:  1. 查看收藏文章列表界面  2. 记录加载出所有文章的响应时间  3. 梯度增加收藏文章数量  4. 从步骤1开始重复至少5次以上  5. 记录平均响应时间 |
| 期望输出:  1. 成功获取收藏文章列表响应时间小于1s |
| 实际输出:  1. |

|  |
| --- |
| **测试用例15: 文章管理之收藏文章容错性** |
| 综述:   * 目的：测试不可收藏重复文章 * 条件：成功获取文章内容，用户存在收藏记录 |
| 步骤:  1. 查看收藏文章列表界面  2. 随机选取一篇文章  3. 在文章尾部点击“收藏”  4. 查看是否出现提示 |
| 期望输出:  1. 提示“你已收藏过该文章” |
| 实际输出:  1. |

|  |
| --- |
| **测试用例16: 文章管理之删除收藏文章准确性** |
| 综述:   * 目的：测试能否正确将文章移出收藏列表 * 条件：成功获取文章内容，用户存在收藏记录 |
| 步骤:  1. 查看收藏文章列表界面  2. 随机选取一篇文章  3. 点击“取消收藏”  4. 查看是否出现成功提示  5. 再次进入收藏文章列表界面，对照查看以上文章是否消失  6. 重复步骤2至少5次 |
| 期望输出:  1. 能够正确将文章移出收藏列表 |
| 实际输出:  1. |

|  |
| --- |
| **测试用例17: 文章管理之最近浏览文章准确度** |
| 综述:   * 目的：测试能否正确获取用户的浏览记录 * 条件：用户登陆小程序，并且能够联网，获得文章内容 |
| 步骤:  1. 随机选取至少10篇以上文章  2. 在文章尾部点击“看过”  3. 进入查看最近浏览文章界面，查看是否出现所有刚刚看过的文章  4. 计算准确度 |
| 期望输出:  1. 所有看过的文章都正确保存 |
| 实际输出:  1. |

|  |
| --- |
| **测试用例18: 文章管理之获取最近浏览文章性能测试** |
| 综述:   * 目的：测试获取最近浏览文章的响应时间 * 条件：成功获取文章内容，用户存在浏览记录 |
| 步骤:  1. 查看最近浏览文章列表界面  2. 记录加载出所有最近浏览文章的响应时间  3. 梯度增加最近浏览文章数量  4. 从步骤1开始重复至少5次以上  5. 记录平均响应时间 |
| 期望输出:  1. 成功获取最近浏览文章列表响应时间小于1s |
| 实际输出:  1. |

### 3.1.6 生词本功能模块

#### 3.1.6.1 控制

在用户选择完具体单词后，调用有道翻译API得到该英语单词的释义，包括中文释义、音标、出现词库，呈现在用户面前；具体释义中包括例句

#### 3.1.6.2 输入

用户选择的单词。

#### 3.1.6.3 输出

* 对单词成功操作的提示；
* 生词本出现相应更改

#### 3.1.6.4 测试用例设计

|  |
| --- |
| **测试用例19: 生词本之添加单词准确性** |
| 综述:   * 目的：测试能否正确添加单词到用户的生词本 * 条件：成功获取文章内容 |
| 步骤:  1. 随机选取一篇文章  2. 从文章中随机10个以上的单词  3. 对单词进行添加  4. 进入生词本界面  5. 查看是否将上述单词添加到单词本中 |
| 期望输出:  1. 成功添加所有单词 |
| 实际输出:  1. |

|  |
| --- |
| **测试用例20: 生词本之删除单词准确性** |
| 综述:   * 目的：测试能否正确删除生词本中的单词 * 条件：成功获取文章内容，生词本中存在生词记录 |
| 步骤:  1. 进入生词本界面  2. 随机选择10个以上的单词  3. 将这些单词一一删除  4. 判断是否从生词列表中消失 |
| 期望输出:  1. 成功删除所有选择单词 |
| 实际输出:  1. |

|  |
| --- |
| **测试用例21: 生词本之获取单词性能** |
| 综述:   * 目的：测试能否在一定时间内快速获取生词本中所有内容 * 条件：登陆小程序，并且能够联网，用户存在100个以上的生词 |
| 步骤:  1. 进入生词本界面  2. 记录加载出生词列表的时间 |
| 期望输出:  1. 可以在1s内加载出所有单词内容 |
| 实际输出:  1. |

|  |
| --- |
| **测试用例22: 生词本之单词复习** |
| 综述:   * 目的：测试单词复习功能的准确性，与生词本中一一对应 * 条件：登陆小程序，并且能够联网，用户存在生词记录 |
| 步骤:  1. 进入生词本界面  2. 点击“单词复习”  3. 进行单词复习的过程查看是否为生词表中的记录 |
| 期望输出:  1. 所有复习单词都是生词表中的 |
| 实际输出:  1. |

## 3.2 性能测试

|  |
| --- |
| **测试用例23: 性能测试** |
| 综述:   * 目的：测试小程序能否在多个并发用户数的情况下正常运行 * 条件：登陆小程序，并且能够联网 |
| 步骤:  1. 让至少5个以上的终端用户使用小程序  2．做出与服务器交互的操作，例如添加生词本、获取文章  3．记录服务器负载以及数据获取的响应速度 |
| 期望输出:  1. 小程序可以在一定并发用户数内正常运行功能 |
| 实际输出:  1. |

## 3.3 界面测试

|  |
| --- |
| **测试用例23: 获取文章内容** |
| 综述:   * 目的：测试小程序能否在加载文章内容过程中有所提示 * 条件：登陆小程序，并且能够联网，有文章列表 |
| 步骤:  1. 获得文章列表  2. 点击进入一篇具体文章  3. 观察是否出现“loading”提示符 |
| 期望输出:  1. 小程序有载入提示 |
| 实际输出:  1. |

|  |
| --- |
| **测试用例24: 操作提示** |
| 综述:   * 目的：测试小程序能否在操作成功或失败后有所提示 * 条件：登陆小程序，并且能够联网 |
| 步骤:  1. 获得文章列表  2. 点击进入一篇具体文章  3. 点击“收藏”，观察是否有“成功”标识  4. 点击“看过”，观察是否有“成功”标识  5. 点击一个具体单词，点击“添加到生词本”，观察是否有“成功”标识  6. 进入生词本界面，删除一个生词，观察是否有“成功”标识  7．进行词汇测试，观察是否有对错提示 |
| 期望输出:  1. 小程序有相应提示 |
| 实际输出:  1. |

|  |
| --- |
| **测试用例25: 文章内容渲染** |
| 综述:   * 目的：测试小程序显示文章内容时是否简洁，美观 * 条件：登陆小程序，并且能够联网，有文章列表 |
| 步骤:  1. 获得文章列表  2. 点击进入一篇具体文章  3. 观察每个文章词间距，段落间距，整体风格，是否存在错位显示等 |
| 期望输出:  1. 文章内容显示正确 |
| 实际输出:  1. |

## 3.4 接口测试

|  |
| --- |
| **测试用例25: 扇贝API** |
| 综述:   * 目的：测试小程序调用的扇贝API的性能 * 条件：登陆小程序，并且能够联网，生词本中存在100个以上单词 |
| 步骤:  1. 进入生词本  2. 在后台查看扇贝接口的调用结果  3. 梯度增加单词数量  4. 重复步骤1  5. 记录出错率 |
| 期望输出:  1. 扇贝API性能较好，能支持大数据量访问 |
| 实际输出:  1. |

# 4评价准则

## 4.1范围

1、响应时间

这里把“响应时间”的概念确定为“对请求作出响应所需要的时间”，把响应时间是做用户视角的软件性能的主要体现。

2、并发用户数

这里把“并发用户数”与“同时在线数”进行区别对待，“并发用户数”的标准是：并发用户取决于测试对象的目标业务场景。

## 4.2数据整理

功能测试使用手工生成测试清单；

性能测试使用XTest生成测试数据。

# 5 资源需求

1、硬件需求

本次测试使用电脑Window 10系统，内存2G，硬盘320G不等；

安卓手机一台（OPPO A59m）；

2、软件需求

微信Web开发者工具（模拟器）；

腾讯优测XTest Server；

3、空间需求

一张办公桌；

4、相关信息保存位置

随时产生测试文档，以word文档形式保存。

# 六、测试结果分析

# 1引言

## 1.1编写目的

此份测试分析报告指明了软件使用过程中，所需要用到的网络条件，设备要求，以及能够流畅使用本软件所需要的一些微信版本要求。

## 1.2背景

被测软件的测试硬件环境： intel® Core™ i7-6500U CPU @ 2.5GHz, 内存：8G，无线网卡：QCA9377

软件条件：win10系统，微信开发者工具模拟器，模拟测试环境为iphone 6

后端条件：Django开启本地服务当作后台的服务器，服务ip地址为：127.0.0.1:8000

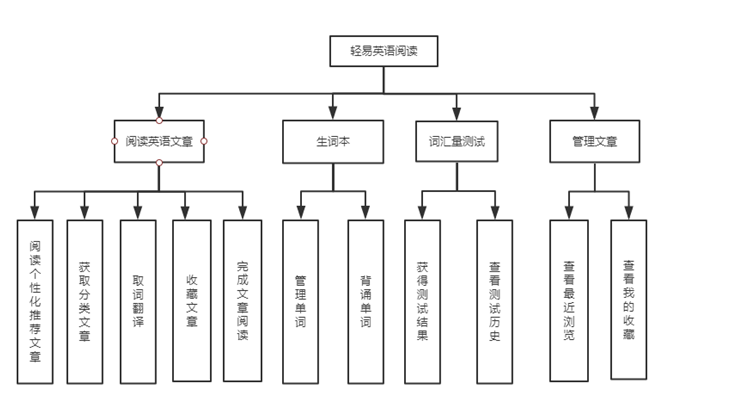
## 1.3定义

## 1.4参考资料

本项目所需要用到的参考资料：

1. 《测试分析报告》（GB8567——88）模板
2. Django测试：<https://docs.djangoproject.com/en/2.2/intro/tutorial02/>
3. 小程序Debug（如何使用模拟器进行Debug）：<https://developers.weixin.qq.com/miniprogram/dev/devtools/debug.html#Wxml-panel>
4. 《测试计划-轻易英语阅读小程序》

# 2测试概要



本次测试按照如上所示的树状图进行功能，性能的测试。测试的结果如下一节所示

# 3测试结果及发现

## 3.1测试用例1-文章分类分类的识别度

|  |
| --- |
| **标题：文章分类分类的识别度** |
| **实际结果**：有部分文章分类和文章的内容不相互符合，有些文章属于Life类，但是却被分到了Health类，测试中还发现有相当大一部分的文章分类属于World和Wordl这两个类别，在下面的截图中可以看到有大量的文章的分类属于world和Health两类。 |
| **结果截图**： |

## 3.2测试用例2-文章个性化推荐

|  |
| --- |
| **标题：文章个性化推荐** |
| **实际结果**：小程序能够基于测试的结果推荐难度适中的文章，对不同等级的用户（包括四六级，考研，托福，GRE等常规考试难度，都可以胜任） |
| **结果截图**：    （此图为词汇量测试成功之后的图片） （此图为用户推荐完之后的文章） |

## 3.3 测试用例3-词汇量之性能测试

|  |
| --- |
| **标题：词汇量之性能测试** |
| **实际结果**：测试实际中可以看到用户单击之后，小程序获取下一个单词的时间一般小于1s，程序相应速度尚可 |
| **结果截图**： |

## 3.4 测试用例4-词汇量测试之准确度测试

|  |
| --- |
| **标题：词汇量测试之准确度测试** |
| **实际结果**：测试过程中发现有些中文意思可以对应多个单词，但是大部分情况下都准确没有误导性。 |
| **结果截图**： |

## 3.5 测试用例5-测试文章的热点界面排版

|  |
| --- |
| **标题：测试文章的热点界面排版** |
| **实际结果**：界面排版美观大方，方便用户使用 |
| **结果截图**： |

## 3.6 测试用例6-测试文章推荐界面排版

|  |
| --- |
| **标题：测试文章推荐界面排版** |
| **实际结果**：热点文章信息排版美观大方 |
| **结果截图**： |

## 3.7 测试用例7-测试文章详细信息排版

|  |
| --- |
| **标题：**测试文章详细信息排版 |
| **实际结果**： 文章具体信息能过够显示成功 |
| **结果截图**： |

## 3.8 测试用例8-测试文章的收藏功能

|  |
| --- |
| **标题：测试文章的收藏功能** |
| **实际结果**：小程序能够正确的对收藏功能进行使用，可以在收藏 |
| **结果截图**： |

## 3.9 测试用例9-测试文章分类信息排版

|  |
| --- |
| **标题：测试文章分类信息排版** |
| **实际结果**：对不同类别的文章单击最上面的bar能够跳转到不同类别的文章 |
| **结果截图**： |

## 3.10 测试用例-文章获取的响应时间

|  |
| --- |
| **标题: 文章获取的响应时间** |
| 结果：文章相应时间在3s以内 |
| 截图： 无 |

## 3.11 测试用例—取词翻译的准确度

|  |
| --- |
| **标题： 取词翻译的准确度** |
| 结果：能够对文章的某些单词进行取词翻译，但是遇到一些带有标点符号的单词有些不能正确的显示单词 |
| 截图： |

## 3.12 测试用例-取词翻译的响应时间

|  |
| --- |
| **标题: 取词翻译的相应时间** |
| 结果：相应时间在1s以内 |
| 截图： 无 |

## 3.13 测试用例-取词翻译之具体释义的准确度

|  |
| --- |
| **标题：取词翻译具体释义的准确度** |
| 结果：对于能够正确识别的单词准确度较高，不会出现解释错误的情况 |
| 截图： 无 |

## 3.14 测试用例-文章管理之收藏文章性能测试

|  |
| --- |
| **标题：文章管理之收藏文章性能测试** |
| 结果：后端能够即使的对用户收藏的文章做出相应 |
| 截图： 无 |
|  |

## 3.15 测试用例-文章管理之获取收藏文章性能测试

|  |
| --- |
| **标题： 文章管理之获取收藏文章性能测试** |
| 结果：后端能够即使给出所收藏的文章的信息 |
| 截图： 无 |

## 3.16 测试用例-文章管理之收藏文章容错性

|  |
| --- |
| **标题：: 文章管理之获取收藏文章性能测试** |
| 结果：文章管理之收藏文章容错性较高，即使文章的标题大同小异也能够正确的被收录 |
| 截图： 无 |

## 3.17 测试用例-文章管理之删除收藏文章准确性

|  |
| --- |
| **标题： 文章管理之删除收藏文章准确性** |
| 结果：删除指定的文章，再次刷新后可以看到收藏夹中对应的文章已经没有了。 |
| 截图： 无 |

## 3.18 测试用例-文章管理之获取最近浏览文章性能测试

|  |
| --- |
| **标题： 文章管理之获取最近浏览文章性能测试** |
| 结果：反应时间为2-3s |
| 截图： 无 |

## 3.19 测试用例-生词本之添加单词准确性

|  |
| --- |
| **标题：生词本之添加单词准确性** |
| 结果：能够正确的添加到数据库中，但是需要用户重新推出小程序，然后再进去才能刷新 |
| 截图： 无 |

## 3.20 测试用例-生词本之删除单词准确性

|  |
| --- |
| **标题：生词本之删除单词准确性** |
| 结果：能够正确的删除到数据库中，但是需要用户重新推出小程序，然后再进去才能刷新 |
| 截图： 无 |

## 3.21 测试用例-生词本之获取单词性能

|  |
| --- |
| **标题：生词本之获取单词性能** |
| 结果：加载所需要的时间为4-5s |
| 截图： 无 |

## 3.22测试用例-生词本之单词复习

|  |
| --- |
| **标题： 生词本之单词复习** |
| 结果：能够在单词本复习完成之后显示对应的复习频率和签到次数 |
| 截图： 无 |

## 3.23测试用例-小程序性能测试

|  |
| --- |
| **标题：小程序性能测试** |
| 结果：能够在多并发用户下正常运行 |
| 截图： |

# 4对软件功能的结论

## 4.1功能1-英文文章分类

### 4.1.1能力

能够实现对不同题材的英文文章的分类

### 4.1.2不足之处

但是当文章中出现一篇文章可能与多题材内容的相符合时就会有一些误判和歧义。

## 4.2功能2-文章个性化推荐

### 4.2.1 能力

经过测试，能够实现根据用户的词汇量测试结果对不同的用户进行数据解析，在后端给用户推荐出合理的难度文章。

### 4.2.2 不足之处

不足之处在于，在用户测试完词汇量之后没有给出一个用户适合看哪一个类型的题材的文章，只是指出了用户在本次词汇量测试的过程中，用户所认识的单词的分布范围。

## 4.3 功能3-词汇量测试

### 4.3.1 能力

经过测试，小程序能够实现对用户进行词汇量测试，可以将测试的结果以柱状图的形式显示出来，柱状图上显示的是用户的

### 4.3.2 不足之处

不足之处在于小程序不能显示用户应该属于哪一类用户，是四级，六级，考研，托福，GRE没有给出明确的指导。

## 4.4 功能4-文章排版和细节显示

### 4.4.1 能力

通过测试用例5，6，7可以看到小程序能够正确的显示文章的各种详细信息，包括文章的标题，阅读人数，以及文章的难度信息，文章的词数都能够正确的显示出来。

### 4.4.2 不足之处

排版有些美中不足的地方在于，由于整个小程序的主调颜色是绿色加白色，而在测试的过程中发现了几处颜色并不一致的地方，也就是说虽然颜色都是绿色，但是颜色的鲜艳程度却不一样，有些组件的绿色和本小程序的绿色并不相符合一致。

## 4.5 功能5-收藏功能

### 4.3.1 能力

经过测试，本收藏功能能够正确的使用，当用户完成阅读文章的时候，如果对此文章有所喜欢，可以单击收藏功能，将此文章的pk值发给后端进行存储。

### 4.3.2 不足之处

此功能点的不足之处在于，不能够对用户已经收藏过的文章进行一些提示。例如不能将已经收藏过的文章的图标加深。

## 4.6 功能6-生词本

### 能力

对于用户的生词本，用户可以正确的添加生词，删除生词，以及获取整个生词本的信息

### 不足之处

不能再添加生词之后及时的显示出用户的信息，需要用户重新退出小程序后再刷新

## 4.7 单词复习历史纪录

### 能力

可以查看对单词的复习历史纪录

# 5分析摘要

## 5.1能力

经过测试本轻易英语阅读小程序完成了词汇量测试，英文文章分类，文章个性化推荐，文章界面排版，文章收藏，……等功能。用户能够比较流畅，方便的使用本软件。

## 5.2缺陷和限制

陈述经测试证实的软件缺陷和限制，说明每项缺陷和限制对软件性能的影响，并说明全部测得的性能缺陷的累积影响和总影响。

总体来说本小程序功能达到了预期，能够有不错的性能表现，功能界面清晰明了。但是仍然有许多不足的地方。我将整合这些不足之处，详细说明这些缺陷对小程序的使用将会带来什么影响。

1. 文章收藏功能，由于没有对收藏功能进行一个重复检测，所以用户没有办法识别出本篇文章是否已经收藏过了，这样用户可能回重复收藏，将给小程序的使用带来一定的苦难。

2. 词汇量测试结果不能明确的指明用户所在的难度等级，所以导致用户可能会对自己的阅读文章的水平有一定的误解。

## 5.3建议

对每项缺陷提出改进建议：

1. 在每次词汇量测试结束之后可以知名用户所在的阅读水平等级，算法可以由后端来完成
2. 将文章的排版细节和配色修改为一致的深绿色，这样不会出现有有些地方的颜色比较深，有些地方的颜色比较浅的问题。
3. 对不同的文章分类更加精确，或者采用比较先进的算法比如多标签分类算法

## 5.4评价

总的来说本软件完成度较高，完成了目标预期的功能需求，对以后开发小程序有了更多的经验可以学习

# 项目总结

## 实际开发结果

### 1.1 软件产品描述

软件模块清单：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 子系统名称 | 模块名称 | 模块标识 |
| 1 | 创建用户 | 创建用户 | CreateUser |
| 2 | 阅读文章 | 个性化推荐文章 | getArticle |
| 按分类查看文章 | getArticlebyType |
| 收藏文章 | collectArticle |
| 查看最近浏览 | history |
| 查词 | getWord |
| 3 | 生词本 | 添加生词 | AddWord |
| 删除生词 | DeleteWord |
| 背单词 | Recite |
| 4 | 新闻内容 | 获取新闻数据 | GetNews |
| 数据清洗 | clearData |
| 5 | 词汇测试 | 获取测试词汇 | getTestWords |
| 查看历史测试 | showTestHistory |

文档清单：

|  |  |  |
| --- | --- | --- |
| 序号 | 文档名称 | 需提交用户 |
| 1 | 项目开发计划 |  |
| 2 | 需求分析 |  |
| 3 | 概要设计 |  |
| 4 | 详细设计 |  |
| 5 | 测试分析 |  |
| 6 | 测试结果 |  |
| 7 | 项目开发总结报告 |  |
| 8 | 编码清单 |  |
| 9 | 安装手册 | 是 |
| 10 | 使用手册 | 是 |

### 2.2 主要功能与性能

|  |  |  |  |
| --- | --- | --- | --- |
| 功能点 | 功能 | 性能 | 开发目标 |
| 个性化推荐文章 | 根据用户的词汇量推荐适合阅读的文章 | 即时、准确 | 达到了 |
| 测试词汇 | 给予用户词汇测试并得出结果 | 具有一定的可靠性 | 达到了 |
| 阅读最新英文新闻 | 基于用户每日最新英语新闻，并提供查词，按分类查看，收藏文章，浏览最近阅读文章的功能 | 内容准确，查词翻译响应准确即时 | 超过了 |
| 背单词 | 针对添加的生词提供背诵功能，已经背熟的单词可以删除 | 内容无误 | 达到了 |
| 获取文章 | 获取外文网站每日最新文章，进行数据清洗和统计 | 即时，准确，广泛 | 未完全达到 |

### 2.3进度

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 阶段 | 计划工作日 | 实际工作日 | 对比 | 原因 |
| 立项 | 2-3 | 7 | 延迟了4天 | 团队成员意见不一致 |
| 开发计划 | 2 | 2 | 按时完成 |  |
| 需求分析 | 2 | 3 | 延迟一天 | 分析之后进行了多次调整 |
| 概要设计 | 1 | 1 | 按时完成 |  |
| 详细设计 | 5 | 5 | 按时完成 |  |
| 编码 | 30 | 35 | 延迟5天 | 前后端工作人员进行接口对接多花了时间 |
| 测试计划 | 2 | 2 | 按时完成 |  |
| 测试结果分析 | 5 | 5 | 按时完成 |  |
| 项目开发总结报告 | 2 | 5 | 按时完成 | 成员共同完成，进度不一致 |
| 其他文档 | 2 | 2 | 按时完成 |  |

## 开发工作评价

### 3.1 对项目开发过程的评价

整个开发过程按照最开始的甘特图，分成了三个阶段，因此，针对每个阶段工作的完成情况以及文档编写情况和最初计划做对比，得出如下的总结。

阶段一：

* 活动：立项，进行可行性和需求分析，撰写需求分析报告。明确项目的功能点。并进行相关技术和知识的学习和搜集。
* 完成情况：良好
* 文档编写情况：初期文档编写比较简略，有诸多没有敲定的细节，文档成型比较简陋
* 出现的问题：功能点不明确，在确定项目所需技术方面不明确，以及需求分析涵盖不全
* 原因：小组成员经验不足，交流不够，导致想法和问题没有及时反馈。

阶段二：

* 活动：进行概要设计，详细设计以及编码。
* 完成情况：良好
* 文档编写情况：文档中的接口设计不明确，导致前后端编码实现出现偏差以及数据库设计有问题，导致多次返工。
* 出现的问题：闭门造车，导致协同工作的成员实现的功能有部分地方重复，文档由成员完成后未经过其他成员的检查和讨论细节，导致后期出现问题，数据库全部推翻重来。
* 原因：功能分配前后端过于明确，导致成员之间无联系和交流，后期需要将整个系统的前后端联系起来时，出现了较大问题，浪费不少时间。无论各自负责的工作是什么，都要及时了解其他成员的工作以及进度，给彼此提出意见。有问题。勿要耽搁，及时反馈。

阶段三：

* 活动：进行测试调试和项目总结工作。
* 完成情况：良好
* 文档编写情况：进行了测试计划和分析，项目总结报告以及其他附属文档的编写。条例比较清晰，文档编写较为规范。
* 出现的问题：测试上针对测试用例的编写出现了一点分歧，以及测试具体执行上有困惑。其他的问题有的没有了解过，导致动手困难。
* 原因：过去都是对cs结构的系统或者java web端的系统进行测试，这是第一次对小程序进行测试，对于方法和工具都不熟悉。

### 3.2 对技术方法的评价

该系统的开发主要使用到了python django框架用于搭建后端服务器，scrapy框架用于获取网络新闻资源，以及JavaScript语言和微信开发的wxml语言进行前端开发，同时使用github进行协同办公。同时运用了多线程，缓存，欧拉距离，栈等设计方式来实现基础功能。

工具用到了vscode文本编辑器，微信开发组件等。采取敏捷开发流程，不断在开发的过程中进行小结，回顾和迭代。

总的来说，使用的都是比较新的技术，结合最适合的方法用简单的设计实现了多数功能。

在技术方法的选择上比较符合实际要求，虽然没有使用难度较大最时兴的技术，但是精简有用。

### 3.3 对产品质量的评价

开发技术和框架的选择较新，开发人员的素质良好，代码实现和沟通都非常流畅和默契。时间进度虽然中间偶有延迟，但是总体上是保持正常的节奏，并且按时完成了。软件的质量中上，不可避免还会有小的缺陷，但是整体功能是完备的。

## 试运行工作评价

在产品试运行的过程中暴露出了一定的接口设计对接，业务流程逻辑的问题，但是最后都调试修正了。基于该系统自身的性质，产品的维护工作量较小，后期只需要将人工辅助实现的数据录入部分变成自动化流程即可达到阶段性维护的要求。

## 技术积累总结

所使用的新技术和框架知识都是基于官方文档以及开源的项目。大部分技术说明都有编写完备的文档，在这里不再赘述。

## 经验与教训

经验教训因人而异，这里分成三部分，分别是邓晏湘，陈彩婷，汪伦独自撰写的心得体会。

### 5.1 邓晏湘

写在前面，首先感谢老师这一学期的指导，感谢助教的协助，也感谢我的组员汪伦，陈彩婷的付出。虽然最后做出来的小程序还有很多不足的地方，但是基本实现了所有的功能点，也能够在大多数情况下正常运行。这一切都离不开我的组员的努力。

老师在课上说，一个组无论任务如何分配，最后肯定会有人做的多有人做的少，这是事实。但是，回顾整个敏捷开发过程，我觉得，如果组内的成员都有责任心，并且配合默契，自觉承担责任较少的人会相应的去帮助任务重的同学，这也是事实。

这是我们小组第二次在课程设计中合作（前一次是软测课设），我作为组长负责后端而他们二人负责前端，一开始我们对任务分配都没有异议，但是开发到后期，我自己认为后端的任务相较于前端会轻松一些（因为他们是第一次使用微信的wxml语言开发小程序，我原来就会一些python，所以学起django和scrapy来没那么难），所以选择多写一些文档，或者是努力完善后端接口设计，尽可能多做一些事，而他们二人也会对彼此的模块进行相应了解，在其中彼此调试艰难的时候，施以援手。

作为组长，我体会到了很多原来独立完成大作业无法了解的东西。分门别类的梳理下来，有如下几点：

学会倾听和讲道理。一开始我们在选题的问题上有不小的争执，汪伦执意想做java web项目，他的理由非常有道理，现在找工作java岗位比较火。但是我认为应该在课设中去学习新东西，应该做一些真正可以用得上的东西。我们第一节课一直在试图说服对方，最后失败。我脾气火爆，一直认为作为组长就应该掌握主动权，定题应该以我为准，听不进别人的意见。实际上这是非常错误的，作为小组领导人，不能专横，要听得进意见，要虚心，也要在组员的想法和能力更加出众的情况下，去肯定他们，去学习。首先承认自己的不足，然后真诚的去了解别人这是很重要的。

不要总是push别人，每个人开发有自己的习惯，节奏慢点就多体谅，不要过分催促。一开始我担心小组进度缓慢，任务堆积，到最后可能会东拼西凑应付了事。所以一直在催促他们快马加鞭的开发，但是事实证明，这样做除了徒增压力，无任何其他效果。一起合作应该给予对方充分的信任，尊重他们的学习开发速度，只需要在项目初期约定的阶段性汇报点一起验收成果即可。

学习是一个循序渐进的过程，想要掌握一门技术，没有捷径，只能不断地尝试，犯错，然后积累经验再改正。如果害怕莫名其妙的bug，害怕困难，那就永远不会进步。首先肯定客观规律，再心平气和地努力学习。无论是在开发还是在学习理论知识的过程中都是这样，不能指望一口吃成大胖子。我一开始阅读官网文档没有耐心，总是跳字，自以为差不多了就动手开始做，这样固然启动快，但是一旦遇到稍微复杂一点的bug就束手无策了。有的bug实际很简单，但是因为我根本不懂稍微基础一些的知识，以及看不懂报错，就浪费大量时间，这些都是需要改正的。

轮子很多，不必自己造，但是不了解别人造轮子依据的原理，永远也没办法把力气用在刀刃上。一开始想的个性化推荐功能可以用聚类/人工智能啥啥算法实现，一想到网上那么多现成的库可以用，并且一旦用进去就会让整个项目都显得“高大上”，就激动的不得了。但是等到梦想照进现实才发现，确实很多现成的东西可以用，还有极其详尽的教程，可是我不懂怎么用，就算数据摆在那里也套不进去。最后还发现，我自己想实现的功能若是搬进人工智能里简直就是杀鸡焉用宰牛刀，多此一举。

说了这么多还是有困惑，我一开始秉承的目的是做一些真正有用的东西并且投入实际，可是热情不足以支撑，做到后来，就没有持续的动力了，还有很多方面可以优化，也可以做的更好，只是一旦到了课设结项时，就没有动力再去继续了。我觉得这样不好，希望以后可以找到自己真正热爱的并且愿意为之奋斗终生的事业吧。

生命不息，奋斗不止，希望今后在每一次的尝试，犯错中我都能获得进步，跌倒后勇敢的爬起来，为成为一个为祖国做贡献，有价值的人。

### 5.2 陈彩婷

本次课程设计已经步入了尾声，一个系统从无到有的过程无疑是非常艰辛的，但是在这个过程中，我也收获了很多，总结了许多经验，不仅包括敏捷开发的流程，还有小程序开发技术，以及团队合作的重要性等。

在整个敏捷开发的过程中，我们首先将首先完成了这个小程序的雏形，然后进行不断的迭代与测试，增加一些缺失的功能，最终完成可以使用的系统。需求也在开发过程中有所变化，例如项目初期的需求是有推荐近期热文，后面由于项目没有初始用户群体，无法对一篇文章的浏览人数进行排序，推荐热文，于是将该功能去掉。

本次课程设计使我熟悉了敏捷开发流程，体会到了敏捷的优势，但是在实际开发中，还是需要多注重实际问题，不能一味强调模式。

需求分析阶段是在项目的开发过程中意义非凡，是开发的依据与基线，一定不能在该阶段马虎。

开发的过程我使用的微信Web开发者工具，使用的是WXML、WXSS、JS文件对前端进行设计编写。接触一个陌生的语言一开始无疑是非常痛苦的，对该语言的使用方法以及允许机制一头雾水。在这个时候一定要沉下心来，从基础开始了解，并且可以类比曾经学过的语言特性。JS对于我来说还算是相对熟悉的，但是只有不断地实践，才能熟练使用。

前端样式的设计也是比较磨人的一个点，预期样式与呈现效果往往大相径庭，这时候更不能心浮气躁，需要耐心地找到问题所在，可以将div设为不同的背景颜色，以更直观地体现增加样式的作用。总之，样式设计还是需要大量的经验积累，初学者必然需要经过摸爬滚打才会知道这两个样式组合会导致的效果。在这部分上我做的不是很好，因为不是平地筑高楼，一味地去掉、增加或者修改一句样式，然后查看结果真的效率奇低。希望之后可以先好好做好基础，不要将大量的时间投入到产出率低的事情上，得不偿失。但是，看到自己的最终成果时，不得不说，还是充满成就感的。

本次项目主要使用组件化开发，我认识到了组件化开发的便利性。使用前辈做好的组件，大大地提高了开发的效率和质量，并且在使用的过程中学习技术。但是在自己的项目中加入组件无疑会时不时出现问题。这时，就要对组件进行修改，以满足本项目的需求。

然后是词汇测试功能模块的实现，虽然功能正确完善，用户可以对词汇量进行测试，并获取相应结果。但是由于微信开发语言的特性，存在大量的代码冗余。主要的问题还是对这一语言不够熟悉吧，当逻辑一旦复杂起来，就只能是使用最简单的语法来做这些比较困难的事，因此不断地重复。

在本次开发中还是遇到过很多坑的，例如JS中this代表当前对象，随着执行的上下文随时变化，回调函数相对于xx点击事件函数对象发生变化，不是data属性，解决方法就是用var that=this 复制对象。

循环for(let i=0;i<10;i++)需要将循环变量用let显示，才会在进行正确的循环流程。

让我印象最深刻的，就是小程序的异步处理promise。使用promise方法，当成功执行该函数功能，再将结果回调给使用该方法的函数，可以有效做到控制两个函数的执行顺序。当循环和promise异步处理结合时，需要保持清醒，对小程序的页面生命周期流程有明确地概念，知道运行的流程，才可以找到解决问题的方法。

同时，我认识到了团队协作的重要性。要做好协作，第一步就是要高效沟通交流，分配好工作任务，这样才能达到最高效率，而不是一味地重复；在沟通中，可以分享学习的知识，使同伴免走弯路；现代项目协作的方法真是先进，使用git版本控制工具，使负责前端的两人可以协作完成开发任务。

能够从一个入门级菜鸟到做出一个功能完善的小程序，还是很有成就感的。开发的小程序还存在这一些不足，开发的过程中也发现了自己在学习上的一些坏习惯，希望我可以取长补短，改掉坏习惯，继续努力，做得更好。

### 5.3 汪伦

本次课程设计学到了小程序开发的基本流程，对小程序开发有了一个深入的了解，也对js语法有了比较熟练的掌握。整个课程设计已经步入尾声，在这个过程中我们从构思一个小程序应该有哪些功能到需要自己动手写css，wxml开始可以说非常有成就感最后做出了一个美观的英语阅读小程序。通过此次开发团队合作过程，我对括敏捷开发的流程，还有小程序开发技术，以及如何使用GitHub进行协同开发有了一个基础的了解，这对我以后迈上工作岗位开发更大的程序打下了坚实的基础。

在整个敏捷开发的过程中，我负责的内容是完成小程序中的取词翻译，刚开始我并没有很明显的思路来实现这个功能，不知道该如何从前端再用户轻触但此后，获取到一篇文本的单个单词信息。后来我联想到可以再后端对整篇文章进行分词，得到一个分词的列表之后对分词的列表中的每一个单词进行一个查询函数的绑定。此功能就能够轻易的完成。

本次课程设计使我熟悉了敏捷开发流程，体会到了敏捷的优势，但是在实际开发中，还是需要多注重实际问题，不能一味强调模式，而且再开发的过程中或者说是最开始一定妖精以后整个程序中要用的那些api接口，以及对应的参数进行协商好，这样在后期就不会有很大的修改，从而提高开发效率，实现比较好的前后端分离。

并且我了解到需求分析阶段是在项目的开发过程中意义非凡，是开发的依据与基线，一定不能在该阶段马虎。

微信自己自身的开发者工具我只用到了其中的界面显示和console输出错误信息这个功能，对代码的编辑由于微信官方的程序对代码编辑的支持过程并不是十分的友好所以我没有选择官方的微信开发者工具进行编辑。小程序并没有像传统的web开发一样，有共用一套web开发语言，相反我需要重新开始学习新的语言，wxml，js.wcss等新鲜的知识。学习路上总会遇到一些痛苦，新的东西总是需要一点点时间来及时消化啊。

比较浪费时间的部分在于前端样式的设计，由于初期只是对css有一个最基本的了解，忽略了里面的很多特性知识，比如继承原则，以及各种绝对宽高和相对宽高。由于css样式非常容易崩溃，或者对设备的兼容性不红。这时候更不能心浮气躁，需要耐心地找到问题所在，可以将div设为不同的背景颜色，以更直观地体现增加样式的作用。总之，样式设计还是需要大量的经验积累，初学者必然需要经过摸爬滚打才会知道这两个样式组合会导致的效果。

由于小程序页面是一套单独的逻辑单元，因此前端的开发非常适合使用组件化开发的技术。每一个人负责一个单独的组件即可，开发效率大大的提高了。但是并使用的过程中学习技术。但是在自己的项目中加入组件无疑会时不时出现问题。这时，就要对组件进行修改，以满足本项目的需求。这个过程需要不断的学习，因为每一个组件都是一个功能独立的单位，暴露出不同的接口。

值得注意的地方就是由于小程序的所有官方web请求都是基于一个异步的请求机制，在这个过程中，我们需要对这些异步请求进行一个封装来保证必须再上一个请求结束完之后才能进行下一步的执行。所以需要使用promise方法，当成功执行该函数功能，再将结果回调给使用该方法的函数，可以有效做到控制两个函数的执行顺序。当循环和promise异步处理结合时，需要尽量不要再循环语句中出现函数的回调，因为一个回调就是相当于fork了一个新的线程，就会导致数据不同步。

由于此项目有点庞大，因为再github上进行协同开发的时候，有时候会遇到一些比较奇怪的冲突，冲突的产生是因为我和另一个组员再对同一个地方做了不同的修改。这个时候就需要我们协商好，以及懂得一些冲突的处理机制。同时，我认识到了团队协作的重要性。要做好协作，第一步就是要高效沟通交流，分配好工作任务，这样才能达到最高效率，而不是一味地重复