

## 摘要

随着科学技术的发展，高性能处理技术所需硬件集成度已经越来越高，手机及平板电脑等移动终端的性能已经得到飞跃式的提升。**Android** 作为近年来新兴的移动设备操作系统，以其优越的性能、可扩展性、易用性等特点迅速占领了移动终端市场。

移动互联网时代，一个高性能的稳定 **Web** 段服务器是广大移动互联网 **App** 开发者所缺乏及关心的一个问题，**Google App Engine** 正是藉此而生。它提供了免费优越的数据存储，处理以及响应服务，解决了一般 **App** 开发者开发 **Web Service** 时的服务器需求。

**关键字：**移动互联网，云服务，安卓系统，谷歌应用引擎

## ABSTRACT

With the development of scientific technology, hardware integration needed for high performance processing technology is getting higher, the performance of mobile terminals such as mobile phones and tablet computers has been dramatic improvement. Android as emerging mobile device operating system in recent years, with its superior performance, scalability, ease of use, characteristics of the mobile terminal market rapidly occupation.

In the mobile internet era, a high performance server web section is a broad problem that mobile Web App developers concern about, Google App Engine is born for this. It provides superior free data storage, processing and response service, solves the Web Service server requirements of the App developers.

**Key Words:** mobile internet, cloud service, Android, Google App Engine

## 目 录

第 1 章 引言 .....	1
1.1 选题背景 .....	1
1.2 研究目标和意义 .....	2
1.3 Android及Google App Engine发展趋势 .....	2
第 2 章 系统开发环境及相关技术 .....	4
2.1 Android开发环境介绍 .....	4
2.2 GAE开发环境介绍 .....	4
2.3 开发操作系统及硬件介绍 .....	4
2.4 本章小结 .....	5
第 3 章 系统分析 .....	6
3.1 需求分析 .....	6
3.1.1 客户端功能 .....	6
3.1.1.1 多语言翻译功能 .....	6
3.1.1.2 注册功能 .....	6
3.1.1.3 登录功能 .....	6
3.1.1.4 注销功能 .....	6
3.1.1.5 本地生词保存功能 .....	6
3.1.1.6 本地历史记录功能 .....	6
3.1.1.7 网络状态检测功能 .....	7
3.1.1.8 云端单词同步功能 .....	7
3.1.1.9 单词测试功能 .....	7
3.1.2 服务端功能 .....	7
3.1.2.1 用户信息保存 .....	7
3.1.2.2 用户名重复验证 .....	7
3.1.2.3 云端生词保存 .....	7
3.1.2.4 云端历史记录保存 .....	7
3.2 性能分析 .....	8
3.3 本章小结 .....	8

---

第 4 章 程序设计 .....	9
4.1 界面设计 .....	9
4.1.1 单词查询界面 .....	9
4.1.2 生词与历史界面 .....	9
4.1.3 注册界面 .....	10
4.1.4 登录界面 .....	11
4.1.5 单词测试界面 .....	12
4.2 数据库设计 .....	13
4.2.1 数据库ER图 .....	13
4.2.2 数据库实体模型图 .....	14
4.2.3 数据库数据字典 .....	16
4.3 程序模块设计 .....	17
4.3.1 软件总体模块设计 .....	17
4.3.2 软件功能模块流程设计 .....	18
4.3.2.1 网络检测模块流程 .....	18
4.3.2.2 单词查询模块流程 .....	18
4.3.2.3 生词功能模块流程 .....	19
4.3.2.4 历史单词功能模块流程 .....	21
4.3.2.5 云端服务模块功能流程 .....	23
4.3.2.6 用户功能模块流程 .....	24
4.3.2.7 单词测试功能模块流程 .....	26
4.4 本章小结 .....	26
第 5 章 程序实现 .....	27
5.1 文件结构与用途 .....	27
5.1.1 Translator文件目录结构说明 .....	27
5.1.2 GTranslator文件目录结构说明 .....	29
5.2 核心功能实现解析 .....	31
5.2.1 翻译核心类Translator.java .....	31
5.2.2 主Activity类MainActivity.java .....	33
5.2.3 本地数据库初始化类DbHelper.java .....	35
5.2.4 云端数据库操作类RemoteDatabase.java .....	37
5.2.5 单词测试Activity类WordTestActivity.java .....	39

---

5.3 本章小结 .....	43
第 6 章 软件测试 .....	44
6.1 网络检查测试 .....	44
6.2 注册测试 .....	44
6.3 登录测试 .....	45
6.4 单词查询测试 .....	45
6.5 生词测试 .....	45
6.6 历史单词测试 .....	46
6.7 单词同步测试 .....	46
6.8 单词测试功能测试 .....	47
6.9 本章小结 .....	47
第 7 章 结束语 .....	48
参考文献 .....	49
致谢 .....	50



## 第1章 引言

### 1.1 选题背景

随着宽带无线移动通信技术的进一步发展和 Web 应用技术的不断创新，移动互联网业务的发展将成为继宽带技术后互联网发展的又一个推动力，为互联网的发展提供一个新平台，使得互联网更加普及，并以移动应用固有的随身性、可鉴权、可身份识别等独特优势，为传统的互联网类业务提供了新的发展空间和可持续发展的新商业模式；同时，移动互联网业务的发展为移动网带来了无尽的应用空间，促进了移动网络宽带化的深入发展。从最初简单的文本浏览、图铃下载等业务形式发展到当前的与互联网业务深度融合的业务形式，移动互联网业务正在成长为移动运营商业发展的战略重点。

随着移动互联网的发展，智能手机迅速兴起，iPhone 的诞生给了世界一个不小的轰动，但其昂贵的价格还是让很多普通人望尘莫及，此时诺基亚依旧占有市场的很大份额，Symbian 系统已经深入人心，有利可图的市场必然会出现搅局者，谷歌正如日中天，当起了这个搅局者的角色。

安迪.鲁宾发明的 android 系统基于 linux 内核，这是从系统级与苹果 iOS 的最大不同，iOS 采用的是 unix 内核，内核不同，但这丝毫不影响 android 强大的图形化操作。

Android 操作系统是开源的，开源是一种精神，开源也让智能手机市场的格局骤变，智能手机生产商开始走向 android 阵营。开源的 android 使使用智能手机的门槛降低，如果你对手机硬件配置要求不太高，那么也许花几百元就能拿到一部智能手机，当然它的操作系统是 android。为了打造了一个开放的开发者平台，谷歌在开源 android 的同时也开放了 android API，开发的核心语言是程序员熟悉的 java，因此对于很多传统的程序员来说，开发门槛较低，能从 j2EE 或者 j2ME 迅速转型，急需人才的 android 开发市场也让程序员的薪水大涨。

依靠软件商店，好的 app 可以达到很高的下载量，而且都是免费的，玩游戏大家都喜欢 FTP，和谷歌的初衷一样，做平台，然后靠广告赚钱，app 赚钱也可以采用广告模式，因此也造就了一个繁荣的移动广告平台市场。而第三方可以修改 android 原生态的 ROM，也使得第三方 ROM 几乎覆盖了所有的 android 设备，呈

现出一个多样的 android 生态环境。

## 1.2 研究目标和意义

本软件从广义上讲, Android 作为当前最流行的移动终端操作系统, 研究其 SDK 的开发使用, 对于程序开发人员了解当前热门开发语言和形势是大有裨益的。而使用 Google API 提供的便利接口, 在减少工作量以及提高代码质量的同时, 可以学习其优秀的实现思路和方法。其次是使用 GAE 作为本软件 Web Service, 无成本地获得了高性能稳定的服务器, 且 GAE 的开发, 管理与使用也应是每个程序开发人员所需要了解的。

从狭义上讲, 此基于 GAE 的翻译软件基于 Google APIs, 能让用户便利的查询各种语言的译文, 且耗费流量低, 速度快, 翻译精准。除了本地端可以储存历史以及生词, 用户也可以通过注册账号, 通过 GAE 上的 Web Service 实现需要单词的云端同步等功能。本软件还提供单词测试功能, 除了为用户提供查询服务以外, 也能达到巩固单词学习的效果。

## 1.3 Android 及 Google App Engine 发展趋势

Android 系统的开源性能为用户提供了多种便利, 系统的扩展性好, 可以独立开发多种功能软件, 还可以提供市场盈利的有效途径, 其震撼人心之处就在于 Android 系统的开放性和免费性服务。Android 作为一个对第三方软件完全开放的平台, 开发者在为其开发程序时拥有更大的自由度, 突破了 iOS 只能添加为数不多的固定软件的枷锁。同时, 与 Windows、Symbian 不同, Android 操作系统免费向开发人员开放, 开发人员可以基于 Android 开发出多种第三方 Rom。Google 与开放手机联盟合作开发了 Android, 这个联盟由包括中国移动、摩托罗拉、高通、宏达电和 T-Mobile 在内的多家技术和无线应用的领军企业组成。Google 通过与运营商、设备制造商、开发商和其他有关各方结成深层次的合作伙伴关系, 希望借助建立标准化、开放式的移动电话软件平台, 在移动产业内形成一个开放式的生态系统。未来越来越多的 Android 的软件会进入手机市场, 乐观的前景也让人期待。

GAE 提供开放大量 API, 供广大部署在 GAE 上的 App 使用, 其接口涉及身份验证, 云存储, 本地存储, 日志, 邮件等方面, 并且 GAE 提供支持 Java, Python



两种语言的 SDK 供开发者使用。GAE 的免费空间有 500M，每月 500 万次 PV，能够满足大部分一般应用，对于个人应用的部署是一个很好的选择。

云计算是未来的发展趋势，GAE 作为谷歌旗下的免费空间，其性能和稳定性得以保证，未来 GAE 还会将其技术优势转化为企业业务模型，为企业提供低成本高性能的云计算服务，使其应用面更加广，市场和前景更加开阔。

## 第2章 系统开发环境及相关技术

### 2.1 Android 开发环境介绍

Android 的上层应用程序是用 Java 语言开发，此处使用 Google 公司推荐使用主流的 Java 集成开发环境 Eclipse。只有 Eclipse 还不够，因为使用 Java 语言进行开发，还应该有由 SUN 公司提供的 Java SDK(其中包括 JRE: Java Runtime Environment)。此外，Android 的应用程序开发和 Java 开发有较大区别的，所以还需要有 Google 提供的 Android SDK。同时，还需要在 Eclipse 安装 ADT，为 Android 开发提供开发工具的升级或者变更，是 Eclipse 下开发工具的升级下载的工具。

其具体版本如下：

Java 7 SDK

Eclipse Indigo

Android SDK

ADT

### 2.2 GAE 开发环境介绍

GAE 开发除了 Java SDK 以外，还需要 Google App Engine SDK for Java，这个 SDK 是 Google Inc 专门提供给 Java 开发者的 GAE 操作接口，里面包含了身份验证，云存储，本地存储，日志，邮件等方面的接口。同时，由于是使用 Eclipse 开发，还需要使用 GAE for Eclipse 插件。需要的软件如下：

Google App Engine SDK for Java

Eclipse Google

### 2.3 开发操作系统及硬件介绍

本次开发的计算机操作系统使用 Ubuntu，其优点是运行稳定，硬件需求低，内存及 CPU 资源占用相对较少。测试机器使用的是 Samsung Galaxy Nexus，避免使用 Dalvik 虚拟机，可以更快速地作测试与调试。具体版本型号如下：

Ubuntu Linux 12.04

Samsung Galaxy Nexus with Android 4.1.2

## 2.4 本章小结

本章研究了 Android 及 Google App Engine 的开发需求，搭建了 Java+Android SDK+GAE SDK+Eclipse 的集成开发环境；分析了软件测试需求，选定了软件测试及开发硬件设备。

## 第3章 系统分析

### 3.1 需求分析

本软件是基于 GAE 的在线翻译软件，其设计目的是提供文本翻译及帮助用户学习记忆单词功能，软件使用前提条件是具备网络连通环境。功能开发分为客户端开发和服务端开发两部分。

#### 3.1.1 客户端功能

##### 3.1.1.1 多语言翻译功能

用户在软件翻译功能界面输入待查询文本，并选取中译英或英译中选项，点击查询后返回中文或英文翻译文本，并显示在查询界面。

##### 3.1.1.2 注册功能

用户从注册界面输入希望注册的用户名、密码以及重复密码，点击用户名检测，软件会返回用户名和密码检测结果，检测通过后点击注册即可成功注册。若检测不通过，软件输出出错提示。注册成功后用户可登录后使用单词云端同步功能。

##### 3.1.1.3 登录功能

用户从登录界面输入用户名密码，若用户名密码正确，将成功登录并能使用登陆后的功能。若用户名密码不正确，软件返回出错信息。

##### 3.1.1.4 注销功能

用户点击注销登录，即可退出登录状态。

##### 3.1.1.5 本地生词保存功能

用户在查询单词界面，在查询单词后点击保“加生词”按钮，即可将当前词语保存为生词。用户在以后使用软件过程中可以翻阅生词本查阅和删除已经保存的单词。

##### 3.1.1.6 本地历史记录功能

软件会自动为用户保存已查询的单词。用户在以后使用软件时可以通过打开历史单词列表查阅和删除历史查询单词。

#### 3.1.1.7 网络状态检测功能

系统提供手机网络联通状态监测，若手机网络不可用，系统会终止当前服务，并提示用户检查网络状况后重新使用软件功能。

#### 3.1.1.8 云端单词同步功能

用户登录后可以使用云端单词同步功能，将本地的生词和历史查询单词存储到云端服务器上。云端生词和历史查询单词与用户相关，可以在用户下次登录软件时同步到手机客户端供用户查阅。

#### 3.1.1.9 单词测试功能

软件从当前手机的本地错词库、生词库、历史词库中提取 10 个用户在测试中错误率最高的单词供用户做单词测试，首次使用会提示用户先保存 10 个生词或查询 10 个单词。测试时用户填入中文（英文）单词的英文（中文）翻译，10 个单词测试完毕后软件显示测试结果，并将错误的单词存入本地错词库更新错词库内容。

### 3.1.2 服务端功能

#### 3.1.2.1 用户信息保存

服务端从手机端接收用户注册时键入的用户名及密码，并保存于云端服务器，供下次用户登录时软件检验登录状态使用。

#### 3.1.2.2 用户名重复验证

服务端在用户注册时将用户键入的用户名与云端数据库的用户名比对，检查当前用户注册用户名是否已存在。

#### 3.1.2.3 云端生词保存

用户在客户端软件登录后使用单词同步功能，同步的单词上传到云端后云端服务器将生词保存到云端生词数据库表。

#### 3.1.2.4 云端历史记录保存

用户在客户端软件登录后使用单词同步功能，同步的单词上传到云端后云端服务器将历史单词保存到云端历史单词数据库表。

## 3.2 性能分析

- 1) 程序 UI 响应速度快，网络联通问题和程序运行状态不会阻塞 UI 进程
- 2) 安全性能良好，不会因用户恶意使用破坏程序进程或服务端进程
- 3) 数据处理速度快，网络连通情况数据处理响应时间不大于 3s
- 4) 网络流量消耗小，在 2G/3G 网络下均可运行

## 3.3 本章小结

本章从用户的角度分析了软件的功能需求，并从用户角度对功能需求的流程做了建议分析。此外本章还对软件的运行性能定下分析标准。

## 第4章 程序设计

### 4.1 界面设计

#### 4.1.1 单词查询界面

单词查询界面用于用户做单词查询，用户可从左侧下拉框选择中译英和英译中功能。查询输入框下方为查询按钮和添加生词按钮，按钮下方为翻译后文本展示区。查询过程中会显示正在查询提示框，如网络未连通，则直接显示网络异常提示信息。单词查询界面原始设计如图 4-1 所示：



图 4-1 单词查询界面

#### 4.1.2 生词与历史界面

生词和历史界面用于展示用户储存的生词以及用户的查询历史，长按某一单

词会出现删除单词对话框，确认后可删除该单词，单击该单词会自动进入单词查询界面，并展示查询结果。默认情况下显示的是本地生词及历史，如果用户登录后，则自动同步云端生词和历史并显示在界面上。生词和历史单词界面原始设计如 4-2 所示：



图 4-2 生词及历史界面

### 4.1.3 注册界面

注册界面是用户注册账号密码的界面，注册后用户可以通过云端服务远程同步生词本和查询历史。该界面还提供用户名格式及重复性检查以及密码格式检查功能。用户注册界面原始设计如图 4-3 所示：



谷歌词典

检测用户名

用户名

test1

用户名必须在4-16位以字母开头，字母或数字结尾仅包含[a-zA-Z0-9\_]的字符串

密码

\*\*\*\*\*

密码合法

重复密码

\*\*\*\*\*

两次密码输入不一致

确定

取消

图 4-3 用户注册界面

#### 4.1.4 登录界面

用户登录界面用于用户登录已注册账号，输入正确的用户名和密码登录后软件将自动同步远程单词到本地，同时用户也可以选择将本地单词上传到云端服务器以便在其他机器上使用。登陆界面原始设计图如图 4-4 所示：



The image shows a user login window titled "谷歌词典" (Google Dictionary). It contains two input fields: "用户名" (Username) with the value "test1" and "密码" (Password) with masked characters "\*\*\*\*\*". Below the fields are two buttons: "确定" (Confirm) and "取消" (Cancel).

图 4-4 用户登录界面

#### 4.1.5 单词测试界面

单词测试界面用于用户的单词测试，此功能通过选取用户生词本和历史记录中错误最多的 10 个单词让用户做翻译，最后给出测试结果。单词测试界面原始设计如图 4-5 所示：



图 4-5 单词测试及测试结果界面

## 4.2 数据库设计

软件数据库分为两部分，一部分位于本地客户端，用于存储本地生词以、历史记录以及用户单词测试单词错误记录，包括 `wrong_word` 表，`new_word` 表，`history_word` 表，其实现使用的是 SQLite 数据库；另一部分位于 GAE 服务端，用于存储用户账户资料以及用户云端单词，包括 `user` 表，`new_word` 表，`history_word` 表，其实现使用的是 Google 的 NoSQL BigTable 技术。

### 4.2.1 数据库 ER 图

云端数据库 ER 图如图 4-6 所示：

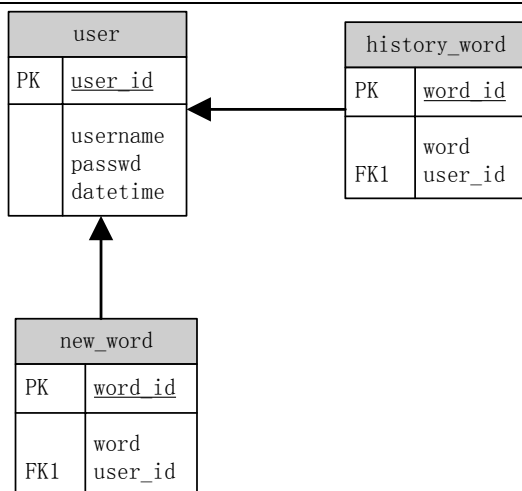


图 4-6 云端服务器数据库 ER 图

## 4.2.2 数据库实体模型图

云端数据库用户信息表 **user** 表实体模型如图 4-7 所示：

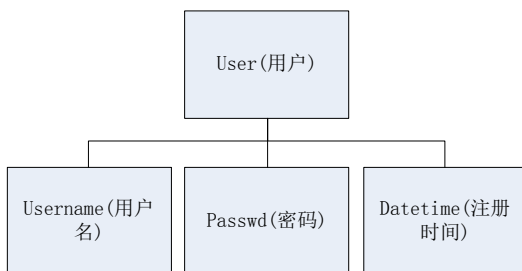


图 4-7 云端数据库 user 表实体模型图

云端数据库生词表 **new\_word** 表实体模型图如图 4-8 所示：

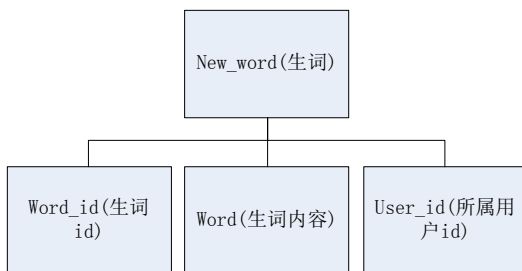


图 4-8 云端数据库 new\_word 表实体模型图

云端数据库历史单词表 **history\_word** 表实体模型图突入 4-9 所示：

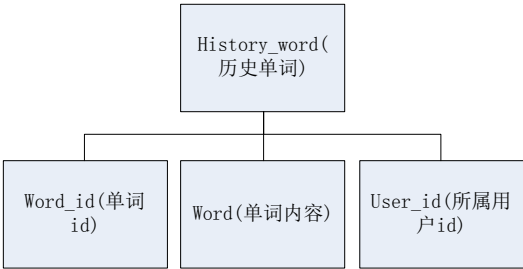


图 4-9 云端数据库 hitory\_word 表实体模型图

本地数据库错词表 wrong\_word 表实体模型图如图 4-10 所示：

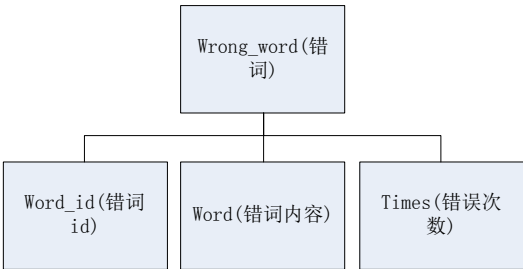


图 4-10 本地数据库 wrong\_word 表实体模型图

本地数据库生词表 new\_word 表实体模型图如图 4-11 所示：

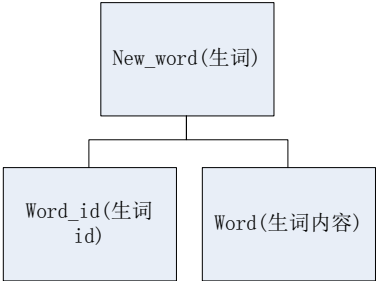


图 4-11 本地数据库 new\_word 表实体模型图

本地数据库历史单词表 hitory\_word 表实体模型图如图 4-12 所示：

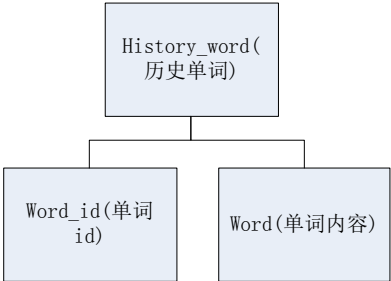


图 4-12 本地数据库 history\_word 表实体模型图

### 4.2.3 数据库数据字典

数据库表数据字典如下各表所示：

表 4-1 本地生词表 history\_word 的结构

属性	数据类型	说明
word_id	integer	自动增加的单词主键
word	varchar	历史词语内容

表 4-2 本地生词表 new\_word 的结构

属性	数据类型	说明
word_id	integer	生词主键
word	varchar	生词内容

表 4-3 本地单词测试表 wrong\_word 的结构

属性	数据类型	说明
word_id	integer	错词主键
word	varchar	错词内容
times	integer	错词出错次数

表 4-4 GAE 上用户账户数据表 user 的结构

属性	数据类型	说明
user_id	integer	用户 id
username	varchar	用户名
passwd	varchar	用户密码
datetime	datetime	用户注册时间

表 4-5 云端历史单词表 history\_word 的结构

属性	数据类型	说明
word_id	integer	单词主键
word	varchar	单词内容
user_id	integer	单词所属用户

表 4-5 云端生词表 new\_word 的结构

属性	数据类型	说明
word_id	integer	单词主键
word	varchar	单词内容
user_id	integer	单词所属用户

## 4.3 程序模块设计

### 4.3.1 软件总体模块设计

从功能上看，本软件共分为注册，登陆，单词云端同步（云端历史，云端生词），本地历史，本地生词，单词测试，单词测试结果等几部分，软件总体功能模块图如图 4-13 所示：

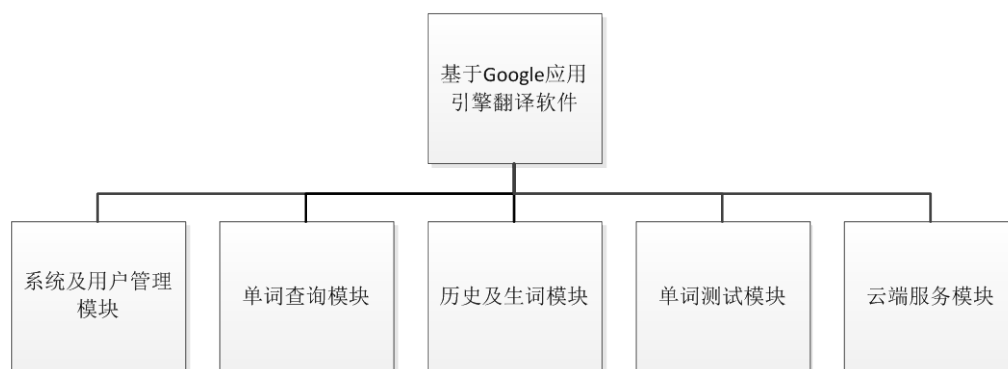


图 4-13 软件功能模块图

云端的 GAE Web Service 主要负责已注册用户的本地单词上传到云端以及云端单词下载到本地功能，本地 SQLite 数据库用于存储未注册用户使用该软件时的历史单词和生词记录。而翻译的核心功能则通过 HTTP 请求 Google APIs 提供的 Translate API v2 实现。

软件所有涉及网络访问部分均设置了网络监测模块，避免用户因网络问题无法获取查询结果却不知情的情况。另外，所有的网络访问部分均封装为线程（Thread）类，使用线程管理器执行网络线程，从而不会阻塞用户 UI 的访问，提供了良好的用户体验交互。

## 4.3.2 软件功能模块流程设计

### 4.3.2.1 网络检测模块流程

网络检测模块通过 Android API 直接获取手机网络连通状态，若网络可用不执行任何操作，否则中止当前服务提示用户检查网络可用性。模块功能流程图如图 4-14 所示：

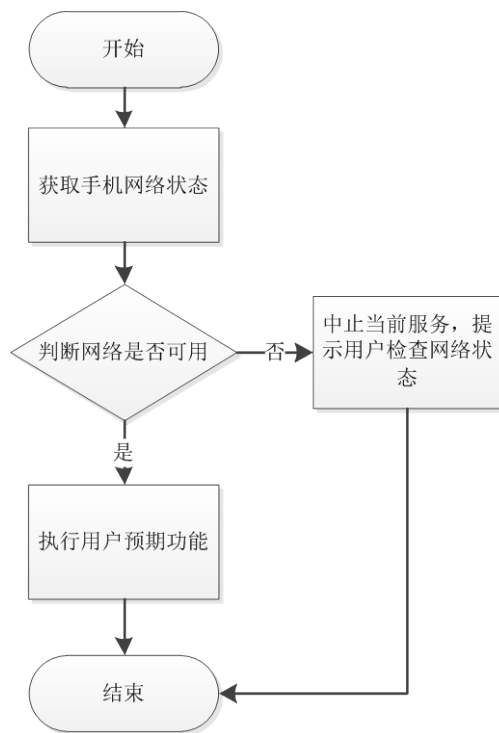


图 4-14 网络检测模块功能流程图

### 4.3.2.2 单词查询模块流程

单词查询模块首先获取用户选择翻译模式，然后将翻译模式和待翻译文本作为参数发送到谷歌翻译服务端，最后获取并展示翻译结果。模块功能流程图如图 4-15 所示：





图 4-15 单词查询模块功能流程图

#### 4.3.2.3 生词功能模块流程

生词功能分为添加生词、查询生词、删除生词三块，这三块功能流程图如图 4-16、图 4-17、图 4-18 所示：

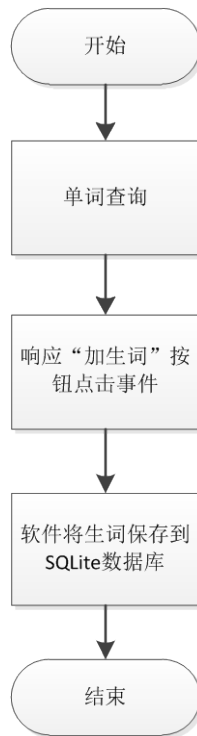


图 4-16 生词添加模块功能流程图

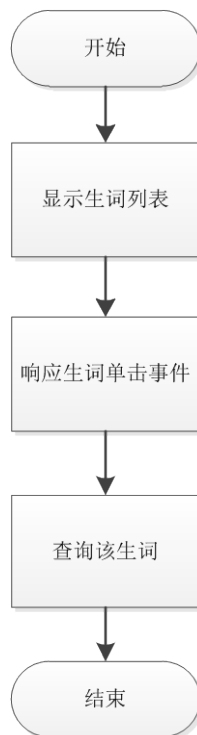


图 4-17 生词查询模块功能流程图

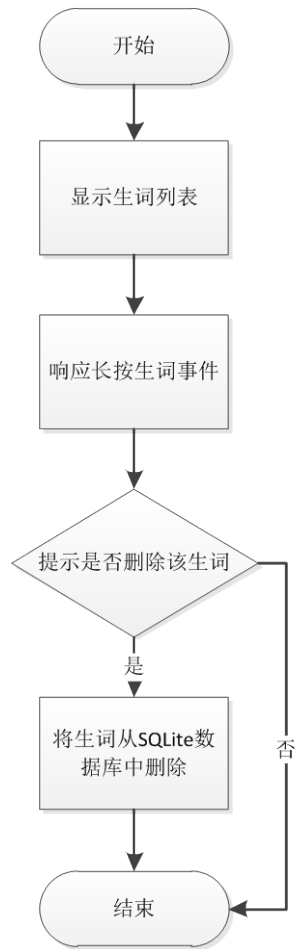


图 4-18 生词删除模块功能流程图

#### 4.3.2.4 历史单词功能模块流程

历史单词功能模块分为历史单词查询、添加、删除三块，其功能流程图如图 4-19、图 4-20、图 4-21 所示：



图 4-19 历史单词添加功能流程图

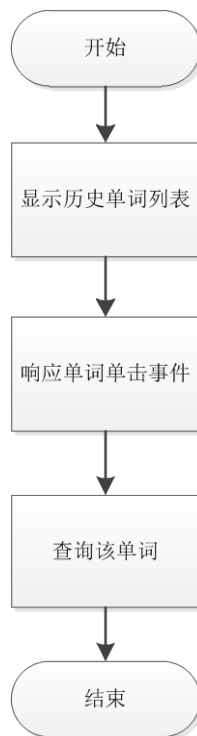


图 4-20 历史单词查询功能流程图

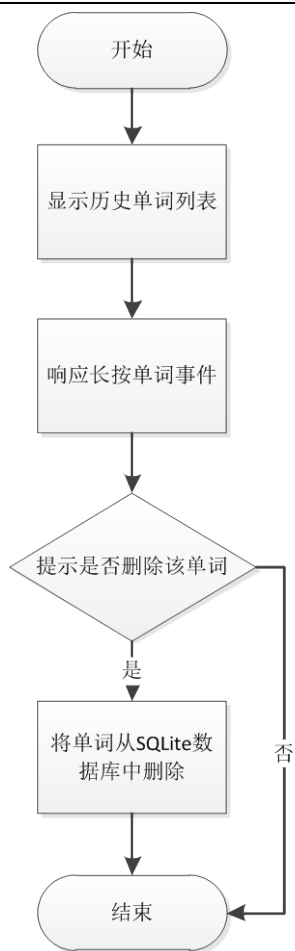


图 4-21 历史单词删除功能流程图

#### 4.3.2.5 云端服务模块功能流程

云端服务模块分为用户信息存储、生词及历史单词存储两块，其模块功能流程图如图 4-22、图 4-23 所示：

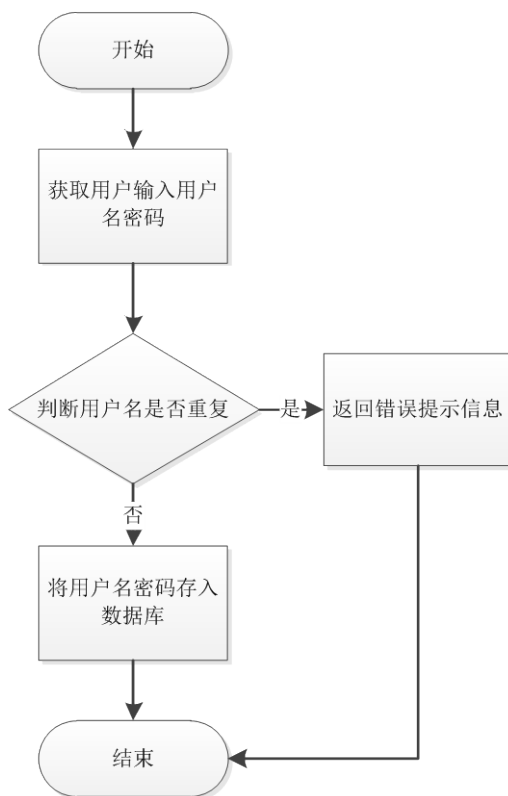


图 4-22 云端用户信息存储功能流程图

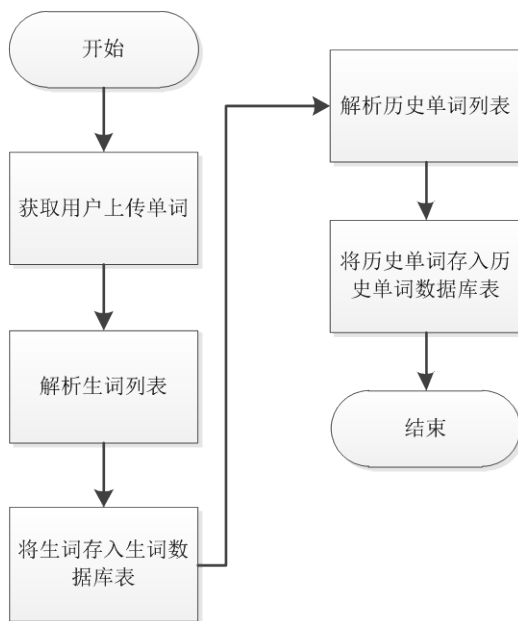


图 4-23 云端单词存储功能流程图

#### 4.3.2.6 用户功能模块流程

用户相关功能包括注册、登录，其功能流程图如图 4-24、图 4-25 所示：

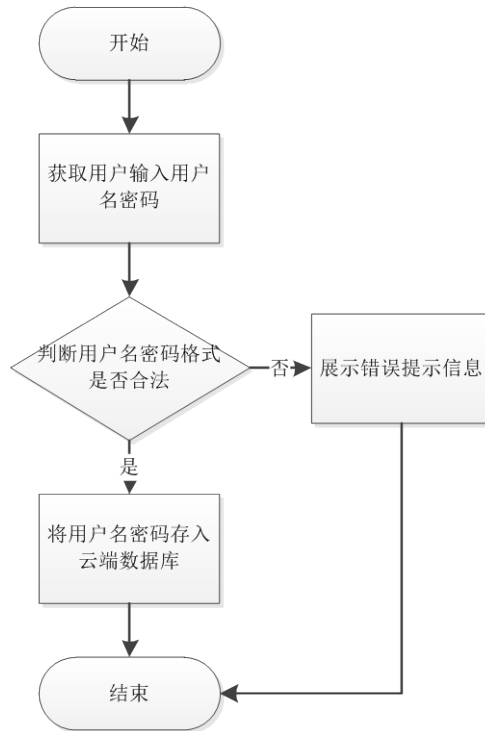


图 4-24 用户注册功能流程图

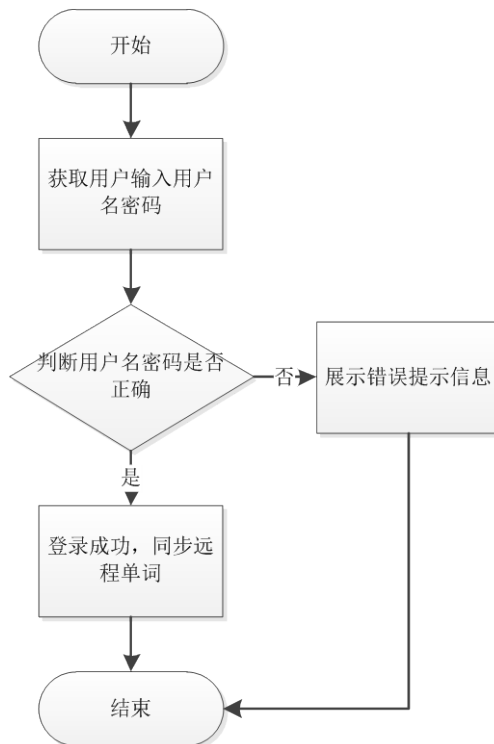


图 4-25 用户登录功能流程图

#### 4.3.2.7 单词测试功能模块流程

单词测试功能共流程图如图 4-26 所示：

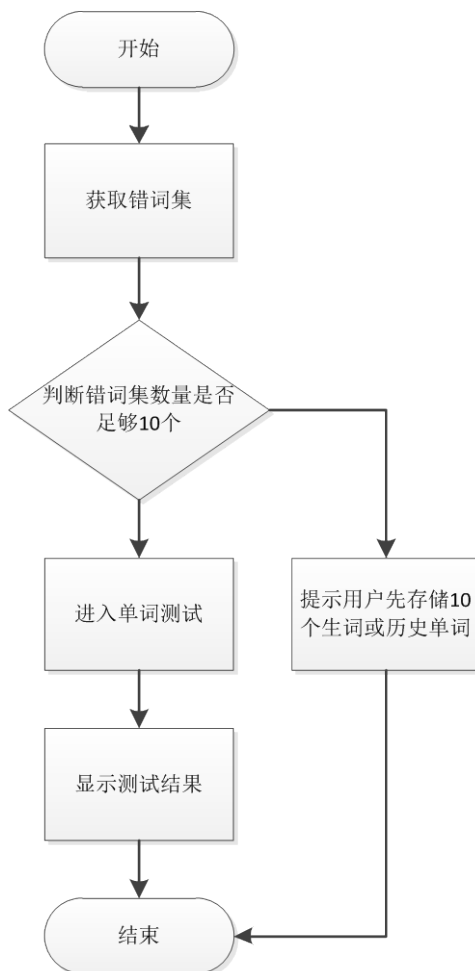


图 4-26 单词测试模块功能流程图

## 4.4 本章小结

本章通过对软件功能需求的分析，设计了软件 UI 界面原始设计图；分析了数据库设计需求，完成数据库 ER 模型及数据库数据字典；最后对每个功能模块流程做了详细流程设计，完成各个功能的程序流程图。




## 第5章 程序实现

### 5.1 文件结构与用途

本软件共分为两个项目，分别为本地的客户端 **Android** 翻译软件项目（项目名 **Translator**）和服务端的 **GAE** 云端存储 **Web Service** 项目（项目名 **GTranslator**）。

#### 5.1.1 Translator 文件目录结构说明

Translator 项目主要源代码文件结构目录如下图所示：

translator / src / org / uestc / translator / 


















All works done!		
 <b>cngddfzlw</b> authored 2 days ago <span>latest commit</span>		
..		
 core	2 days ago	<a href="#">All works done!</a> [cngddfzlw]
 AddNewWordActivity.java	5 days ago	don't know what [cngddfzlw]
 AppContext.java	2 days ago	All works done! [cngddfzlw]
 DbHelper.java	2 days ago	All works done! [cngddfzlw]
 DeleteWordActivity.java	5 days ago	don't know what [cngddfzlw]
 DicActivity.java	3 days ago	Finish menu items change dynamically [cngddfzlw]
 HistoryActivity.java	a month ago	add delete new words and history [cngddfzlw]
 LoginActivity.java	3 days ago	Finish menu items change dynamically [cngddfzlw]
 LoginingActivity.java	5 days ago	don't know what [cngddfzlw]
 MainActivity.java	2 days ago	All works done! [cngddfzlw]
 NewWordActivity.java	a month ago	add delete new words and history [cngddfzlw]
 QueryActivity.java	5 days ago	don't know what [cngddfzlw]
 RegingActivity.java	5 days ago	don't know what [cngddfzlw]
 RegisterActivity.java	5 days ago	don't know what [cngddfzlw]
 TestResultActivity.java	2 days ago	All works done! [cngddfzlw]
 WordTestActivity.java	2 days ago	All works done! [cngddfzlw]

图 5-1 org.uestc 包下所有文件

translator / src / org / uestc / translator / core / 









All works done!		
	cngddfzlw authored 2 days ago	latest
..		
 Data.java	3 days ago	Finish words sync func [cngddfzlw]
 DIHisWordsThread.java	3 days ago	Finish words sync func [cngddfzlw]
 DINewWordsThread.java	3 days ago	Finish words sync func [cngddfzlw]
 LocalDatabase.java	2 days ago	All works done! [cngddfzlw]
 LoginThread.java	5 days ago	add synchronize cloud words func [cngddfzlw]
 RegisterThread.java	5 days ago	don't know what [cngddfzlw]
 RemoteDatabase.java	3 days ago	Finish words sync func [cngddfzlw]
 Translator.java	2 days ago	All works done! [cngddfzlw]
 UploadWordsThread.java	3 days ago	Finish words sync func [cngddfzlw]
 UsernameDupThread.java	5 days ago	don't know what [cngddfzlw]
 Validator.java	3 days ago	Finish menu items change dynamically [cngddfzlw]

图 5-2 org.uestc.core 包下所有文件

其中 org.uestc 包下为 Android 程序所有 Activity，主要功能是提供用户及单词数据初始化，准备 UI 界面布局等。org.uestc.core 包下为软件数据库操作封装类，网络访问线程封装类，静态数据封装类，数据验证类以及核心翻译功能类。两个包分别为视图包和服务类包。其源代码文件具体说明如下表：

表 5-1 Translator 工程的文件用途说明

包 名 称	文 件 名	说 明
org.uestc	AddNewWordActivity.java	添加生词 Activity
org.uestc	AppContext.java	软件上下文全局数据储存类
org.uestc	DbHelper.java	数据库操作基类，包括创建，打开，关闭数据库等操作
org.uestc	DeleteWordActivity.java	删除生词及历史单词 Activity
org.uestc	DicActivity.java	单词查询界面 Activity
org.uestc	HistoryActivity.java	历史单词界面 Activity
org.uestc	LoginActivity.java	登录界面 Activity

续表 5-1 Translator 工程的文件用途说明

org.uestc	LoginingActivity.java	正在登陆界面 Activity
org.uestc	MainActivity.java	软件主 Activity
org.uestc	NewWordActivity.java	生词界面 Activity
org.uestc	QueryActivity.java	单词查询 Activity
org.uestc	RegingActivity.java	正在注册 Activity
org.uestc	RegisterActivity.java	注册界面 Activity
org.uestc	TestResultActivity.java	单词测试结果 Activity
org.uestc	WordTestActivity.java	单词测试界面 Activity
org.uestc.core	Data.java	全局静态数据类
org.uestc.core	DIHisWordsThread.java	云端下载历史单词线程类
org.uestc.core	DINewWordsThread.java	云端下载生词线程类
org.uestc.core	LocalDatabase.java	本地数据库操作类
org.uestc.core	LoginThread.java	登陆线程类
org.uestc.core	RegisterThread.java	注册线程类
org.uestc.core	RemoteDatabase.java	云端数据库操作类
org.uestc.core	Translator.java	翻译功能核心类
org.uestc.core	UploadWordsThread.java	上传云端单词线程类
org.uestc.core	UsernameDupThread.java	用户名重复检测线程类
org.uestc.core	Validator.java	数据格式验证类

Android 的资源文件保存在/res 的子目录中。/res/values 目录中保存的是用来自定义字符串和颜色的文件，/res/layout 目录中保存的是 XML 格式的布局文件，/res/menu 目录中保存的是软件菜单布局文件。软件所有的界面布局以及显示的文字资源均来自于这些 XML 配置文件，文件控制了软件界面展示效果，具体代码所呈现的效果如 4.1 界面设计中截图所示，此处不再赘述。

### 5.1.2 GTranslator 文件目录结构说明

GTranslator 主要源代码文件如下图：

[gtranslator](#) / [src](#) / [org](#) / [uestc](#) / [gtranslator](#) / [+](#)












Finish server-end develop		
 <b>cngddflzw</b> authored 4 days ago		
..		
 <a href="#">Data.java</a>	4 days ago	Finish server-end develop [ <a href="#">cngddflzw</a> ]
 <a href="#">DIHisWordsServlet.java</a>	4 days ago	Finish server-end develop [ <a href="#">cngddflzw</a> ]
 <a href="#">DINewWordsServlet.java</a>	4 days ago	Finish server-end develop [ <a href="#">cngddflzw</a> ]
 <a href="#">LoginServlet.java</a>	6 days ago	add synchronize words func [ <a href="#">cngddflzw</a> ]
 <a href="#">PMF.java</a>	4 days ago	Finish server-end develop [ <a href="#">cngddflzw</a> ]
 <a href="#">RegisterDupServlet.java</a>	6 days ago	add synchronize words func [ <a href="#">cngddflzw</a> ]
 <a href="#">RegisterServlet.java</a>	6 days ago	add synchronize words func [ <a href="#">cngddflzw</a> ]
 <a href="#">UploadWordsServlet.java</a>	4 days ago	Finish server-end develop [ <a href="#">cngddflzw</a> ]
 <a href="#">hisword.java</a>	4 days ago	Finish server-end develop [ <a href="#">cngddflzw</a> ]
 <a href="#">new_word.java</a>	4 days ago	Finish server-end develop [ <a href="#">cngddflzw</a> ]

图 5-3 GTranslator 核心源代码目录

下表是各文件及其功能简要说明：

表 5-2 GTranslator 工程的文件用途说明

包 名 称	文 件 名	说 明
org.uestc.gtranslator	Data.java	全局静态数据类
org.uestc.gtranslator	DIHisWordsServlet.java	历史单词下载 Servlet
org.uestc.gtranslator	DINewWordsServlet.java	生词下载 Servlet
org.uestc.gtranslator	LoginServlet.java	用户登录 Servlet
org.uestc.gtranslator	PMF.java	持久化管理器工厂类
org.uestc.gtranslator	RegisterDupServlet.java	用户名重复性检查 Servlet
org.uestc.gtranslator	RegisterServlet.java	注册 Servlet
org.uestc.gtranslator	UploadWordsServlet.java	用户单词同步 Servlet

续表 5-2 GTranslator 工程的文件用途说明

org.uestc.gtranslator	HistoryWord.java	历史单词实体类，类名与云端数据库表实体名对应，类内的各个成员变量与数据库表实体内的变量名对应。
org.uestc.gtranslator	NewWord.java	生词实体类

## 5.2 核心功能实现解析

### 5.2.1 翻译核心类 Translator.java

翻译核心类 Translator.java 是本软件功能核心翻译功能提供类，它以特定的格式请求由 Google APIs 提供的翻译功能 API，获取翻译后的特定格式文本，最后解析翻译结果并呈现在用户 UI 上。Google Translate API 可以动态地在多种语言间翻译文本，它有免费版和收费版，此处使用的是免费版。其部分核心代码如下：

```
URL url = new URL(urlStr);
URLConnection conn = (URLConnection) url.openConnection();
    conn.setDoOutput(true); // 可以发送数据
conn.setDoInput(true); // 可以接收数据
conn.setRequestMethod("POST"); // POST方法
// 必须注意此处需要设置UserAgent，否则google会返回 403
conn.setRequestProperty
    ("User-Agent", "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)");
conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
conn.connect();
// 写入的POST数据
OutputStreamWriter osw = new OutputStreamWriter(conn.getOutputStream(), "UTF-8");
    osw.write("q=" + queryString + "&sl=" + srcLang + "&tl=" + tgLang + "&tc=1");
    osw.flush();
    osw.close();
// 读取响应数据
```

```
BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream()));  
String s;  
StringBuilder result = new StringBuilder("");  
while ((s = in.readLine()) != null)  
    result.append(s);  
queryResult = result.toString();  
return queryResult;
```

其中 urlStr 即为 Google APIs 提供的翻译 Api 接口, 软件只需提供翻译源语言, 翻译目标语言以及待翻译文本三项数据, 将这些数据 POST 到 API URL, 最后 Google 的 Web Service 会返回特定格式的翻译完成文本, 程序通过 InputStreamReader 读取返回的信息并最后交付给 UI 进程展示结果。其效果图如图 5-4 所示:



图 5-4 翻译核心类效果图

### 5.2.2 主 Activity 类 MainActivity.java

MainActivity.java 主要提供的功能是准备各个 Activity 在初始化时需要的数据, 例如生词, 历史单词等; 初始化各个 Activity 的初始样式; 准备软件菜单在点击后所需要执行的操作; 根据用户登录状态动态改变软件 UI 样式和内容; 用户退出软件时保存本地历史单词以及生词到 SQLite 数据库。以下是其核心代码摘要片段:

```
// 初始化生词
appContext.setNewWordSet(ldb.getNewWords());

// 初始化历史词语
appContext.setHistorySet(ldb.getHistory());

// 初始化tabhost
tabs = (TabHost) findViewById(R.id.mainTabhost);
tabs.setup(this.getLocalActivityManager());

// 初始化词典tab
TabHost.TabSpec spec = tabs.newTabSpec(getString(R.string.dic));
Intent intent = new Intent().setClass(this, DicActivity.class);
// tab选择时重新加载tab的Activity内容
intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
spec.setIndicator(getString(R.string.dic));
spec.setContent(intent);
tabs.addTab(spec);

// 初始化生词tab
intent = new Intent().setClass(this, NewWordActivity.class);
intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
spec = tabs.newTabSpec(getString(R.string.newWord));
spec.setIndicator(getString(R.string.newWord));
spec.setContent(intent);
tabs.addTab(spec);
```

```
// 初始化历史tab
intent = new Intent().setClass(this, HistoryActivity.class);
intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
spec = tabs.newTabSpec(getString(R.string.history));
spec.setIndicator(getString(R.string.history));
spec.setContent(intent);
tabs.addTab(spec);
tabs.setCurrentTab(0);

// 修改tab样式
TabWidget tabWidget = tabs.getTabWidget();
int count = tabWidget.getChildCount();
for (int i = 0; i < count; i++) {
    View view = tabWidget.getChildTabViewAt(i);
    view.setLayoutParams().height = 150; //tabWidget.getChildAt(i)
    final TextView tv = (TextView) view.findViewById(android.R.id.title);
    tv.setTextSize(30);
    tv.setTextColor(Color.BLACK);
}

@Override
public boolean onPrepareOptionsMenu(Menu menu) {
    // 根据登录状态动态改变菜单内容
    menu.clear();
    AppContext ac = (AppContext) getApplicationContext();
    if (ac.getUsername() == null || ac.getUsername().equals("")) {
        getMenuInflater().inflate(R.menu.unlog_menu, menu);
        super.onPrepareOptionsMenu(menu);
    } else {
        getMenuInflater().inflate(R.menu.logged_menu, menu);
        super.onPrepareOptionsMenu(menu);
    }
}
```



```
menu.getItem(0).setTitle("当前用户: " + ac.getUsername());  
}  
return true;  
}
```

可以发现生词和历史以及单词查询的界面 Tab 都是由 MainActivity 来准备的, 其中对他么添加了 FLAG\_ACTIVITY\_CLEAR\_TOP 标记, 此标记可以使得所有 tab 切面在切换时重新刷新数据, 藉此可以显示最新的生词和历史内容。onPrepareOptionsMenu()则可以提供动态改变软件主菜单的功能, 其原理是判断用户登录状态, 在用户登录后菜单中会显示用户名, 并加入上传单词到云端的能。onStop()方法为软件停止运行时的回调方法, 它用于存储本地生词和历史到 SQLite 数据库。其效果图如图 5-5 所示:



图 5-5 主 Activity 显示效果图

### 5.2.3 本地数据库初始化类 DbHelper.java

SQLiteDatabase 是 Android SDK 中操作数据库的核心类之一。使用

SQLiteDatabase 可以打开数据库，也可以对数据库进行操作。然而为了数据库升级的需要以及使用更方便，往往使用 SQLiteOpenHelper 的子类来完成创建、打开数据库及各种数据库操作。SQLiteOpenHelper 是个抽象类，在该类中有如下两个抽象方法，SQLiteOpenHelper 的子类必须实现这两个方法。

```
public abstract void onCreate(SQLiteDatabase db);
public abstract void onUpdate(SQLiteDatabase db,int oldVersion,int newVersion);
```

SQLiteOpenHelper 会自动检测数据库文件是否存在。如果存在，会打开这个数据库，在这种情况下就不会调用 onCreate() 方法。如果数据库文件不存在，SQLiteOpenHelper 首先会创建一个数据库文件，然后打开这个数据库，最后调用 onCreate() 方法。因此，onCreate() 方法一般用来在新创建的数据库中建立表、视图等数据库组建。也就是说 onCreate() 方法在数据库文件第一次创建时调用。DbHelper.java 继承自 SQLiteOpenHelper，其功能是初始化 SQLite 数据库，包括数据库各表的元数据记录，以及建表和删表操作等。其核心代码片段如下图：

```
public static final String DB_NAME = "dictionary";
public static final String TB_NEW_WORD = "new_words";// 生词表
public static final String TB_HISTORY = "history";// 历史表
public static final String TB_MISTAKE = "mistake";// 错词表
public static final String COL_NW_WORD_ID = "word_id";// 生词id
public static final String COL_NW_WORD = "word";// 生词内容
public static final String COL_MISATAKE_WORD_ID = "word_id";// 生词内容
public static final String COL_MISTAKE_WORD = "word";// 错词内容
public static final String COL_MISTAKE_TIME = "time";// 错词次数
public static final String COL_HIS_WORD_ID = "word_id";// 历史单词id
public static final String COL_HIS_WORD = "word";// 历史单词内容

public DBHelper(Context context, String name, CursorFactory factory,
    int version) {
    super(context, name, factory, version);
    // TODO Auto-generated constructor stub
}

@Override
```

```

public void onCreate(SQLiteDatabase arg0) {
    arg0.execSQL("CREATE TABLE IF NOT EXISTS " + TB_NEW_WORD +
        " ( " + COL_NW_WORD_ID + " integer primary key autoincrement," +
        COL_NW_WORD + " varchar not null unique )");
    arg0.execSQL("CREATE TABLE IF NOT EXISTS " + TB_HISTORY +
        " ( " + COL_HIS_WORD_ID + " integer primary key autoincrement," +
        COL_HIS_WORD + " varchar not null )");
    arg0.execSQL("CREATE TABLE IF NOT EXISTS " + TB_MISTAKE +
        " ( " + COL_MISATAKE_WORD_ID + " integer primary key autoincrement," +
        COL_MISTAKE_WORD + " varchar not null," +
        COL_MISTAKE_TIME + " integer not null )");
}

@Override
public void onUpgrade(SQLiteDatabase arg0, int arg1, int arg2) {
    // TODO Auto-generated method stub
    arg0.execSQL("DROP TABLE IF EXISTS " + TB_NEW_WORD);
    arg0.execSQL("DROP TABLE IF EXISTS " + TB_HISTORY);
    arg0.execSQL("DROP TABLE IF EXISTS " + TB_MISTAKE);
    onCreate(arg0);
}

```

#### 5.2.4 云端数据库操作类 RemoteDatabase.java

RemoteDatabase.java 主要通过对 GAE 上的 Web Service 数据库的操作，完成对注册用户的云端单词的上传和下载同步功能。此处本地软件与 GAE Web Service 的交互实现方式是用 HTTP 请求实现的：首先将待上传单词拼接成特定格式的字符串，然后加入到 POST 数据里传输到服务端，再由服务端解析这些数据保存到云端数据库；下载云端数据同理，先由 GAE 服务端将数据按特定格式凭借成长字符串，下载下来以后再由本地客户端解析，并展现给用户使用。以下为该类核心代码片段：

```
// 拼接历史单词和生词
```

```
String resultStr = "";
String hisWordsStr = "";
String newWordsStr = "";

if (hisWords.size() > 0) {
    StringBuilder hisWordsBuilder = new StringBuilder();
    for (String word : hisWords) {
        hisWordsBuilder.append(word + "|");
    }
    hisWordsStr = hisWordsBuilder.toString()
        .substring(0, hisWordsBuilder.length() - 1);
}

if (newWords.size() > 0) {
    StringBuilder newWordsBuilder = new StringBuilder();
    for (String word : newWords) {
        newWordsBuilder.append(word + "|");
    }
    newWordsStr = newWordsBuilder.toString()
        .substring(0, newWordsBuilder.length() - 1);
}

String urlStr = "http://uestctranslator.appspot.com/uploadwords";
URL url = new URL(urlStr);
URLConnection conn = (URLConnection) url.openConnection();
conn.setDoOutput(true);// 可以发送数据
conn.setDoInput(true);// 可以接收数据
conn.setRequestMethod("POST");// POST方法
conn.setRequestProperty
("User-Agent", "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)");
conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
conn.connect();
//写入的POST数据
```

```

OutputStreamWriter osw = new OutputStreamWriter(conn.getOutputStream(), "UTF-8");
osw.write(USERNAME_TAG + "=" + username + "&" + HISWORD_TAG + "="
+
hisWordsStr + "&" + NEWWORD_TAG + "=" + newWordsStr);
osw.flush();
osw.close();
// 读取响应数据
BufferedReader in = new BufferedReader(
new InputStreamReader(conn.getInputStream()));
String s;
StringBuilder result = new StringBuilder("");
while ((s = in.readLine()) != null)
    result.append(s);
resultStr = result.toString();

```

从代码片段可以看出所有的单词均会使用“&”符号连接起来，在上传和下载后在通过 `String.split()` 方法切分字符串恢复成单词，并存入 `List` 或者 `Set` 中，最后展现在本地客户端列表上或者通过 `PMF` 生成的数据库管理器存入云端数据库。

### 5.2.5 单词测试 Activity 类 WordTestActivity.java

`WordTestActivity.java` 主要提供用户单词测试功能，它从用户生词和历史单词中选取 10 个用户以往在测试中错的最多的单词，让用户进行互译默写测试，并在最后展现测试结果，将测试中出错的词语继续加入错词库，以供下次测试使用。如果用户还未进行过单词测试，则直接从历史和生词中随即选取 10 个单词进行测试。如果历史和生词不足 10 个，则提醒用户先添加生词或增加历史查询。其核心代码片段如下：

```

// 从数据库找出测试中错的最多的 10 个单词
final LocalDatabase db = LocalDatabase.getInstance(WordTestActivity.this);
final AppContext ac = (AppContext) getApplicationContext();
Set<String> wordsSet = new HashSet<String>(db.getMistakeWords());
wordsSet.addAll(db.getHistory());

```

```

wordsSet.addAll(db.getNewWords());
// 如果生词和历史单词少于 10，提示用户先多储存生词和历史单词
if (wordsSet.size() < 10) {
    Dialog dialog = new AlertDialog.Builder(WordTestActivity.this).setIcon(
        android.R.drawable.btn_star).setTitle(null).setMessage(
            "生词及历史单词不足 10 个，请先存储 10 个生词和历史单词
    ").setPositiveButton("确定",
        new DialogInterface.OnClickListener() {

            @Override
            public void onClick(DialogInterface dialog,
            int which) {
                WordTestActivity.this.finish();
            }
            }).create();
            dialog.show();
            return;
        }

// 储存原词，翻译单词，正确翻译单词
List<String> wordsList = new ArrayList<String>(wordsSet);
ac.setOriginWords(wordsList);
ac.setTransWords(new ArrayList<String>());

// 将正确单词结果置空
for (int i = ac.getCorrectWords().size(); i <= 10; i++) {
    ac.getCorrectWords().add("");
}

// 一次性查出所有正确结果
ExecutorService exec = Executors.newCachedThreadPool();
int crrCount = 0;

```

```
for (String originWord : wordsList) {  
    // 自动判断中英互译模式  
    Pattern p = Pattern.compile("[A-z]+");  
    Matcher m = p.matcher(originWord);  
    String srcLang, tgLang;  
    if (m.find()) {  
srcLang = "en";  
tgLang = "zh_CN";  
    } else {  
srcLang = "zh_CN";  
tgLang = "en";  
    }  
    Translator t = new Translator(srcLang, tgLang,  
        originWord, ac.getCorrectWords(), crrCount++);  
    exec.execute(t);  
}  
exec.shutdown();
```

从代码可以看出，对于翻译的单词是中译英还是英译中，软件采用自动判断翻译方式的方法，如果文本包含英文字母，则自动将源语言设置为英文，目标语言设置为中文，反之则将源语言设置为中文，目标语言设置为英文，然后一次性获取所有翻译结果，最后在用户做完单词测试以后，将正确结果和用户翻译结果转为小写，比对正确翻译结果和用户填写翻译，最后展示翻译结果，其中翻译正确单词用绿色表示，翻译错误单词用红色表示。其效果图如图 5-6、图 5-7 所示：



图 5-6 单词测试界面效果

原词	你的翻译	正解
北京	beijing	beijing
飞行	fly	flight
坐	sit	sit
行走	walk	walk
晃悠	swim	swinging
我可以	i can	i can
加拿大	canada	canada
伊拉克	iraq	iraq
美国	america	united states
中国	china	china

图 5-7 单词测试结果界面效果



### 5.3 本章小结

本章分析了软件程序代码文件结构，简要介绍各代码文件功能，对核心功能实现代码文件代码摘要做了简要分析，并展示了核心代码实现效果图。

## 第6章 软件测试

### 6.1 网络检查测试

在进行网络连接相关的功能前，软件会检测 Wifi 和手机 2G/3G 网络连通情况，如果网络无法连通，软件会提示用户检查网络连通情况，并停止当前正在提供的服务。测试用例表如表 6-1 所示：

表 6-1 网络连接状态测试用例表

编号	功能模块	输入数据	预期输出	实际输出	测试结果
1	网络可用性检测	关闭手机数据网络及 Wifi	提示用户检查网络状态	提示用户检查网络状态	通过

### 6.2 注册测试

注册部分会对用户注册时填写的用户名密码进行检测，如填写文本不符合规范，软件会给出相应提示，并终止注册流程，待用户修正填写文本后再继续注册。其测试用例表如表 6-2 所示：

表 6-2 注册测试用例表

编号	功能模块	输入数据	预期输出	实际输出	测试结果
1	注册功能	正确格式且合法用户名密码	注册成功	注册成功	通过
2	注册功能	错误格式用户名密码	提示用户检查用户名密码格式	提示用户检查用户名密码格式	通过
3	注册功能	已存在用户名	提示用户用户名已存在	提示用户用户名已存在	通过

续表 6-2 注册测试用例表

4	注册功能	两次密码输入不一致	提示用户两次密码输入不一致	提示用户两次密码输入不一致	通过
---	------	-----------	---------------	---------------	----

6.3 登录测试

当用户登录时输入用户名或密码不正确时，会给予相应提示，并终止登录进程。登录成功后，将自动同步登录用户的云端单词到本地，并赋予同步本地单词到云端的功能。其测试用例表如表 6-3 所示：

表 6-3 登录测试用例表

编号	功能模块	输入数据	预期输出	实际输出	测试结果
1	登录功能	正确的用户名密码	登录成功	登录成功	通过
2	登录功能	错误的用户名密码	提示用户用户名密码错误	提示用户用户名密码错误	通过

6.4 单词查询测试

单词查询功能通过读取用户选择的翻译模式，将单词查询结果显示在软件界面上。其测试用例表如表 6-4 所示：

表 6-4 单词查询功能测试用例表

编号	功能模块	输入数据	预期输出	实际输出	测试结果
1	单次查询	待查询文本	翻译后文本	翻译后文本	通过

6.5 生词测试

生词功能模块包括添加、删除、查询，其测试用例表如表 6-5 所示：

表 6-5 生词功能测试用例表

编号	功能模块	输入数据	预期输出	实际输出	测试结果
1	生词添加	生词	保存生词到数据库，显示生词添加成功	保存生词到数据库，显示生词添加成功	通过
2	生词查询	无	显示生词翻译结果	显示生词翻译结果	通过
3	生词删除	无	从数据库删除该生词	从数据库删除该生词	通过

## 6.6 历史单词测试

历史单词功能模块包括历史单词添加、删除、查询，其测试用例表如表 6-6 所示：

表 6-6 历史单词测试用例表

编号	功能模块	输入数据	预期输出	实际输出	测试结果
1	历史单词添加	单词查询	保存历史单词到数据库	保存历史单词到数据库	通过
2	历史单词查询	无	历史单词生词翻译结果	历史单词生词翻译结果	通过
3	历史单词删除	无	从数据库删除该历史单词	从数据库删除该历史单词	通过

## 6.7 单词同步测试

单词同步功能将单词从本地上传到云端服务器，在下次登录时同步到本地客户端。其测试用例表如表 6-7 所示：

表 6-7 单次同步测试用例表

编号	功能模块	输入数据	预期输出	实际输出	测试结果
1	同步云端单词到本地	云端单词	云端单词同步到本地	云端单词同步到本地	通过
2	同步本地单词到云端	本地单词	本地单词同步到云端	云端单词同步到本地	通过

## 6.8 单词测试功能测试

单词测试功能测试用例表如表 6-8 所示：

表 6-8 单词测试功能测试用例表

编号	功能模块	输入数据	预期输出	实际输出	测试结果
1	单词测试	测试单词翻译文本	单词测试结果列表	单词测试结果列表	通过

## 6.9 本章小结

本章对软件各功能模块进行测试，测试结果表明软件各功能模块实现均与设计要求相吻合，无异常、错误输出。

## 第7章 结束语

本软件从功能上讲，包括查词，历史，生词，云同步，单词测试等功能，已经完全达到了一个可进入 Android 市场的 App 的要求。从架构讲，其囊括了 UI 设计，本地后台功能开发，服务端功能开发等方面，是一个综合考量各方面编程技术的项目。从实现技术上讲，软件综合运用了本地与远程的交互功能，在客户端和服务端使用了多线程，SQLite，NoSQL 等技术。在程序实现上，多次复用了自定义接口，如网络检测、提示运行的界面 UI 等。

从用户角度来说，使用此软件可以实现单词查询，单词学习，学习水平测试等目的，在解决翻译疑难的同时也能巩固单词记忆。在用户体验上，在程序运行时都给出了程序运行状态提示，以及用户操作指导提示等，让用户在使用本软件时更简明易懂。

本次软件设计及开发，让我从软件设计，用户体验，客户端开发，界面开发，服务端开发等方面学到知识，巩固了以往编程中不熟悉的点，让我对软件设计开发有了全局的认识和理解。

## 参考文献

- [1] (美) 巴萨姆/ (美) 塞若/ (美) 贝茨.Head First Servlets&JSP[M].(苏钰函/林剑).北京: 北京电力出版社, 2006, 105-133
- [2] 李波/史江萍/王祥凤.Android 4.X 从入门到精通[M].北京: 清华大学出版社, 2012, 41-70
- [3] Dan Sanderson.GAE 编程指南[M].(唐学韬/何继业).北京: 机械工业出版社华章公司, 2010, 120-170
- [4] 李刚.疯狂Android讲义[M].北京: 电子工业出版社, 2011, 206-237
- [5] 杨丰盛.Android 应用开发揭秘[M].北京: 机械工业出版社, 2010,70-95
- [6] 李雪飞/吴明辉.Android 开发入门教程[M].北京: 人民邮电出版社, 2010, 144-153
- [7] 范怀宇.Android 开发精要[M].北京: 机械工业出版社, 2012, 80-99
- [8] 马尔科.加尔根塔.Learning Android[M].(李亚舟/任中龙/杜钢).北京: 电子工业出版社, 2012, 270-299
- [9] Charles Serverance.Using Google App Engine[M].America: O'Reilly Media, 2009, 133-174
- [10] Eugene Ciurana.Developing with Google App Engine[M].America: Apress, 2009, 35-70
- [11] Eugene Ciurana.Beginning Google App Engine[M].America: Apress, 2009, 322-340
- [12] Miller Frederic P/Vandome Agnes F/McBrewster John.Google App Engine[M].America: Alphascript, 2010, 166-210
- [13] James Goodwill/Ikai Lan.Professional Google App Engine Programming with Java.France: Wrox, 2010, 301-320

## 致谢

本课题的设计和论文撰写过程中，我的指导老师赵洋老师给予了极大的帮助，为我提供了大量的相关资料。在完成课题设计的过程中，老师在总体设计思想上给予了许多重要的实际指导，拓展了我的思路，使得我得以完成了任务。在此，我对赵洋老师的指导表示衷心的感谢。

同时，我也衷心的感谢电子科技大学计算机学院四年来对我的精心栽培，良好的校园环境、精良的师资队伍、浓厚的校园文化都让我深深怀念，都为我进入社会提供了莫大的帮助。