**软件学院2032-2024学年度第二学期**

**《软件工程理论与实践课程设计》**

**实验报告**

|  |  |
| --- | --- |
| **班级：** | 软件工程2203班 |
| **项目名称：** | TransLink\_PW |
| **项目组组长：** | 潘隆财 |
| **项目组成员：** | 学号：U202210416  姓名：  学号：U202216408  姓名： |
| **指导老师：** | 刘小峰 |

**目录**

[1. 项目概述 3](#_Toc168591146)

[2. 需求描述 3](#_Toc168591147)

[2.1 功能性需求 3](#_Toc168591148)

[2.2 非功能性需求 3](#_Toc168591149)

[3. 系统设计 3](#_Toc168591150)

[3.1 架构设计 3](#_Toc168591151)

[3.2 界面原型设计 3](#_Toc168591152)

[3.3 详细设计 3](#_Toc168591153)

[4. 系统实现 3](#_Toc168591154)

[5. 系统界面展示 4](#_Toc168591155)

[6. 总结 4](#_Toc168591156)

## 项目概述

### 项目基本介绍

**UI界面：**

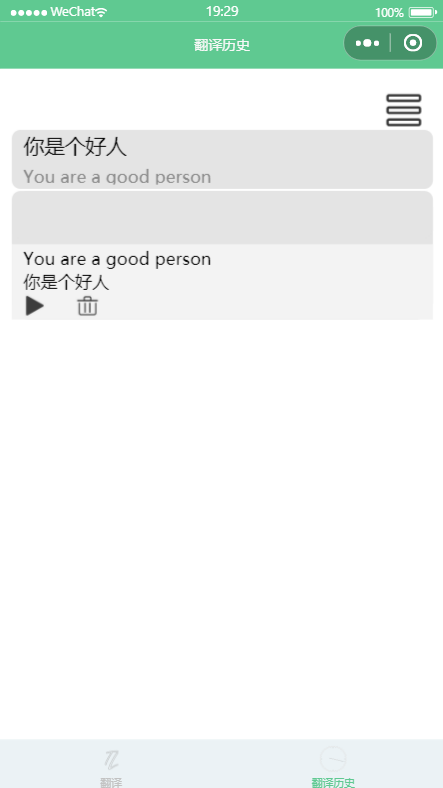
 

图1- 1UI界面

**基本介绍：**

TransLink\_PW是一款集多功能于一身的微信小程序，它不仅能够将输入文本快速准确地翻译成目标语言，还能够提供语音识别，图片识别快速地帮助用户获得输入文本。并且TransLink\_PW还提供了历史记录的功能，让用户能够轻松地管理翻译历史记录。

### Github仓库地址

[2042270217/TransLink\_PW (github.com)](https://github.com/2042270217/TransLink_PW)

### 人员基本分工

**潘隆财：**主要负责前端的开发和部分文档的撰写

1. 共同撰写文档：一起参与讨论于构思，完成需求分析，系统设计，界面原型设计等文档。
2. 原型设计：使用Mastergo设计小程序的初代界面原型。
3. 界面编写：完成了TransLink、History、ClipPicture页面的编写
4. 组件开发：为了更加方便地编写History界面，开发了singleHistory组件

**王珺戈：**主要负责后端的开发和部分文档的撰写

1. 共同撰写文档：一起参与讨论与构思，完成需求分析，系统设计等文档。
2. 系统设计：设计系统架构和组件的接口设计。
3. 组件编写：完成了utils中翻译与历史记录后台功能的编写
4. 组件应用：页面中完成接口调用

## 需求描述

### 功能性需求

#### 用户需求

1. **集成翻译服务 （Integrated Translation Service）**
2. 用户需求：

* 用户在输入框中输入文字，选择输入语言与目标语言，程序在输出框中给出翻译结果。
* 用户可以选择上传相册中的图片并选取图片范围进行翻译
* 用户可以拍摄图片并选取图片范围进行翻译
* 用户可以输入语音作为文本进行翻译
* 用户可以拍摄文字进行实时翻译
* 用户可以查看，复制或者删除翻译历史
* 用户可以选择多种翻译语言
* 用户可以复制或分享翻译结果
* 用户可以通过语音合成功能了解待翻译和翻译后文本的语音

1. **系统需求**

基础翻译功能：

1. **初始假设**

* 用户在输入框中输入文字，选择输入语言与目标语言，程序在输出框中给出翻译结果。

1. **正常状态**

* 用户在输入框中输入文字后，程序自动检测选择的输入语言与输入语言是否匹配，程序将输入文本显示为目标语言显示在输出框中。

1. **可能的错误**

* 程序会检测输入文本是否包含无法翻译的特殊字符，若是，程序会进行提示并要求用户重新输入
* 程序会检测输入文本是否过于复杂无法翻译，程序会进行提示并要求用户重新输入
* 网络连接不稳定，无法访问，程序会要求用户检测网络连接状态

1. **其他活动**

* 程序会记录用户上一次选择的翻译语言并在下一次启动时自动选择上一次的选择语言
* 程序会记录用户的翻译历史并在输入文本框中显示可供用户点击的历史记录

1. **完成的系统状态**

* 用户可以通过打开程序，并进入翻译界面来进行翻译。程序会自动检测输入语言，并在下拉菜单中选择相应的目标语言。
* 用户在输入框中输入待翻译的文本后，程序会进行翻译并在输出框中显示翻译结果。
* 程序记录了用户的翻译历史，并允许用户在历史记录中查看以前的翻译结果

集成翻译功能：

1. **初始假设**

* 用户在翻译界面中选择了图片翻译后，用户可以在相册或者通过拍照上传图片，用户需要选择翻译的输入语言与输出语言，之后程序会将翻译文本显示在图片上。

1. **正常状态**

* 用户在翻译界面选择输入语言和输出语言后，用户可以选择上传相册的图片或拍照上传图片，程序将待翻译文本显示在输入框并在输出框输出翻译结果。

1. **可能的错误**

* 网络连接不稳定，文件上传错误，程序会提醒用户检测网络状态并重新上传
* 文件过大无法上传，程序会提醒用户进行压缩或重新上传
* 系统权限不足，无法访问用户图库，程序会提醒用户是否授权

1. **其他活动**

* 系统应该对用户图库进行保护，确保不被未授权的其他脚本访问。

1. **完成的系统状态**

* 用户通过图片翻译功能翻译图片上的文字。

语音合成功能：

1. **初始假设**

* 用户在输入框和输出框单击播放按钮，即可播放语音合成片段。

1. **正常状态**

* 用户在翻译界面输入框中输入后，用户可以选择点击播放按钮，程序会自动播放输入文本语音合成的结果。

1. **可能的错误**

* 网络连接不稳定，未成功播放，程序应提醒用户检测网络状态并重新尝试
* 输入过大无法上传，程序会提醒用户进行删减
* 系统权限不足，无法播放音频，程序会提醒用户是否授权

1. **其他活动**

* 系统应该对用户内存进行定期的处理，防止内存占用过大。

1. **完成的系统状态**

* 用户通过语音合成功能得到合成的音频片段。

### 非功能性需求

1. **性能需求**

* 网络性能需求：用户的翻译请求应在一定的时间内返回翻译结果，在网络延迟过高的情况下应提醒用户网络错误
* 准确性：翻译结果必须准确无误，避免出现语法错误、歧义或不准确的翻译。
* 多语言支持：小程序需要支持多种语言的翻译，包括常见的全球主要语言和一些地区性语言。
* 资源利用率：翻译小程序需要合理利用系统资源，包括CPU、内存和存储空间，以避免资源浪费或过度占用。

1. **安全需求**

* 数据加密：对于传输和存储在小程序中的敏感数据（如用户输入的文本和翻译结果），需要采用加密算法确保数据的机密性和完整性。
* 用户身份验证：小程序需要提供安全的用户身份验证机制，确保只有经过授权的用户能够访问翻译服务，并防止未经授权的访问。
* 日志记录与审计：小程序需要记录用户的操作日志和系统事件，以便在出现安全问题时进行审计和追踪，并及时发现和应对潜在的安全威胁。
* 安全更新：小程序需要定期进行安全更新和补丁，以修复已知的安全漏洞和缺陷，并及时应对新的安全威胁。

1. **可用性需求**

* 高可靠性：小程序需要保持高度可靠，尽量避免出现故障或系统崩溃，以确保用户始终能够访问和使用翻译服务。
* 持续性：小程序需要提供持续的服务，确保在任何时间用户都能够访问和使用翻译功能，不受时间限制或服务中断的影响。
* 快速启动：小程序需要快速启动并响应用户请求，尽量减少等待时间，提高用户的满意度和使用体验。
* 简单易用：小程序界面设计简洁明了，操作简单直观，让用户能够轻松地完成翻译操作，无需复杂的学习过程。

1. **可维护性需求**

* 模块化设计：小程序需要采用模块化的设计方法，将功能模块进行分离和封装，以便于单独维护、测试和更新。
* 清晰的代码结构：小程序的代码需要具有良好的结构和命名规范，易于理解和维护，降低代码的复杂度和维护成本。
* 版本控制：小程序的代码和配置需要进行版本控制管理，使用版本控制系统（如Git）来跟踪和管理代码的变更历史，方便团队协作和代码回滚。
* 自动化测试：小程序需要建立完善的自动化测试体系，包括单元测试、集成测试和端到端测试等，确保代码的质量和稳定性，在进行代码修改或更新时能够及时发现和修复问题。
* 文档和注释：小程序需要提供清晰的文档和注释，包括代码注释、API 文档和操作手册等，帮助开发人员理解和使用代码，提高代码的可维护性。

1. **可扩展性需求**

* 松耦合架构：小程序需要采用松耦合的架构设计，减少模块之间的依赖关系，降低系统的耦合度，方便对系统进行扩展和修改。

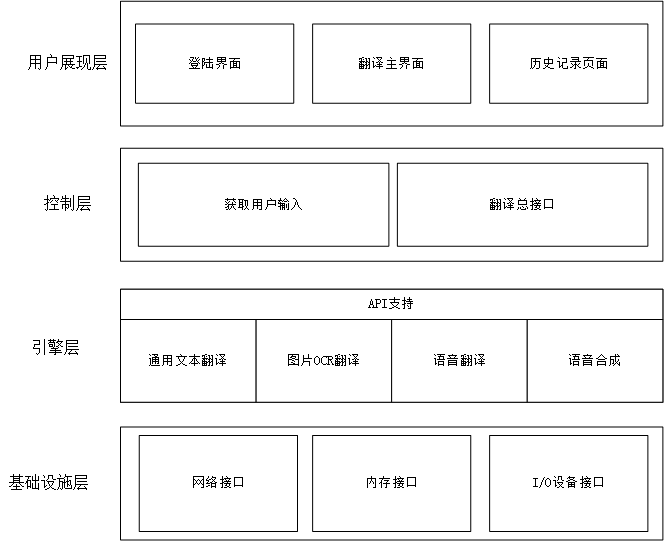
1. **用户体验需求**

* 直观的界面设计：小程序的界面设计需要简洁明了，符合用户的直觉和习惯，让用户能够快速理解和使用翻译功能。
* 易用性：小程序需要提供简单易懂的操作流程和界面布局，降低用户的学习成本和操作难度，让用户能够轻松地完成翻译任务。
* 快速加载和响应：小程序需要保持快速的加载速度和响应速度，尽量减少等待时间，提高用户的满意度和使用体验。
* 个性化定制：小程序可以提供个性化的定制选项，允许用户根据自己的喜好和习惯进行界面布局、主题风格等方面的定制，提升用户的满意度和忠诚度。
* 帮助与支持：小程序需要提供清晰的帮助文档和支持渠道，包括用户手册、操作指南、在线帮助等，帮助用户解决问题和学习如何使用翻译功能。
* 用户反馈机制：小程序需要提供用户反馈的渠道和机制，鼓励用户提供意见和建议，以持续改进和优化用户体验，提高用户满意度和忠诚度。

## 系统设计

### 架构设计

小程序的设计架构为分层架构，以下为架构示意图：



#### 4.1.1. 用户展现层

* 用户展现层是用户与小程序交互的界面，用于控制用户的数据输入输出，查看历史界面等功能。用户展现层分为登陆界面，翻译主界面和历史记录界面，登陆界面负责用户对小程序的授权，翻译主界面主要负责用户的交互，历史记录界面负责用户查看、复制的功能

#### 4.1.2 控制层

* 控制层负责接收用户的请求，解析请求参数，调用相应的业务逻辑处理，并将处理结果返回给客户端。在小程序中，控制层用于接受并解析用户发送的请求，提取请求参数和数据，以及翻译的API封装的翻译函数的实现，用于处理用户展现层的交互和利用引擎层提供的API接口。

#### 4.1.3 引擎层

* 引擎层位于控制层之下，负责实现系统的核心业务逻辑和功能。在本小程序中即调用API接口实现翻译功能和语音合成功能，并将数据传输给控制层展现给用户。

#### 4.1.4 基础设施层

* 基础设施层主要包括通过微信开发平台提供的API函数实现对用户内存，网络，I/O设备的访问，实现获取图片，发送用户请求等更基础的功能，并为引擎层的API函数提供支持。

### 界面原型设计

#### 翻译界面

翻译界面是小程序的入口，包含了小程序的主要功能，包括用户输入模块，语言选择模块，翻译输出模块以及翻译方式选择。

* 输入框：用户可以在这个框中手动输入需要翻译的文本。
* 语音播放器（输入框中）：用户点击按钮可以播放需要翻译的文本的语音。
* 复制按钮（输入框中）：用户点击按钮可以复制输入框的内容到剪切板
* 翻译按钮：用户点击这个按钮后，翻译框会显示翻译后的文本。
* 语言选择框：用于点击左边的按钮可以选择源语言，点击右边的按钮可以选择目标语言，点击中间的按钮可以交换源语言和目标语言。
* 翻译框：用户可以在这个框中看到翻译后的文本。
* 语音播放器（翻译框中）：用户点击按钮可以播放翻译后的文本的语音。
* 复制按钮（翻译框中）：用户点击按钮可以复制翻译框的内容到剪切板
* 麦克风按钮：用户点击按钮可以开始录制语音，再次点击结束录制
* 相机按钮：用户点击后可以选择拍照或者上传图片进行翻译，随后进入裁剪界面

#### 历史记录界面

历史记录界面可以查看以及删除翻译历史记录

* 多选按钮：用户点击后可以激活左侧的复选框
  + 全选按钮：用户点击后可以全选复选框
  + 删除按钮：用户点击后可以删除选择的历史记录
* 单个历史记录：用户点击后可以展开历史记录
  + 语音播放器：用户点击按钮可以播放翻译后的文本的语音。
  + 删除按钮：用户点击后可以删除该条历史记录

#### 裁剪界面

裁剪界面可以裁剪用户上传的图片，小程序会识别图片的文字进行翻译

* 裁剪框：用户拖动裁剪框可以设置裁剪的范围，单击裁剪框可以预览裁剪后的图片
* 裁剪按钮：用户点击后小程序自动裁剪图片，并且跳转到翻译界面

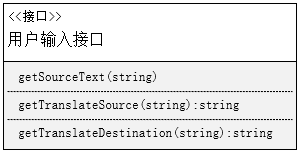
### 详细设计

#### 接口设计

1. **用户展现层接口：**

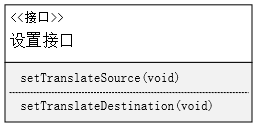
* translate()：将用户输入信息传送给控制层

1. **控制层接口：**
2. **获取用户输入接口：**



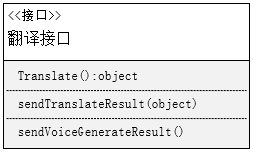
* getSourceText(string)：获取用户的输入待翻译文本或图片或语音等
* getTranslateSource(string)：获取用户的源语言
* getTranslateDestination(string)：获取用户的待翻译语言

1. **设置接口：**



* setTranslateSource(void)：将输入语言设置为用户选择的输入语言
* setTranslateDestination(void)：将翻译语言设置为用户选择的翻译语言

1. **翻译接口：**



* Translate()：通过引擎层接口实现翻译
* sendTranslateResult(object)：将翻译结果发送至用户展现层
* sendVoiceGenerateResult()：将语音合成的结果发送至用户展现层

1. **引擎层接口：**
2. **语音合成API**



* GenerateVoice(text)：用于合成语音并播放

1. **翻译API**



* GenerateTranslate(q,from,to,Object)：通用文本翻译API
* PictureTranslate(from,to,that,Photourl)：图片翻译API

1. **基础设施层接口：**



* wx.request()：向API提供平台发送网络请求
* wx.setStorage()：向用户内存存储历史记录
* wx.getFileSystemManager()：微信小程序的文件管理API，负责语音合成中写入文件与播放文件
* wx.getStroageInfo()：实现获取本地存储的历史记录的信息，用于遍历历史记录并展现在history页面中

## 系统实现

### components/

#### components/image-cropper

image-cropper是在微信开发者社区中寻找的插件，在项目中实现了裁剪图片的主要功能

#### components/singleHistory

singleHistory组件定义了历史记录界面中单个历史记录的呈现形式，主要实现的功能有：

* 记录输入文本与输出文本信息，以及管理历史记录所需的id
* 能够实现单击转换展开和折叠的样式

singleHistory.js

1. Component({
2. */\*\**
3. \* 组件的属性列表
4. \*/
5. properties: {
6. currentText:{
7. type:String,
8. value:'历史'
9. },
10. outputText:{
11. type:String,
12. value:'History'
13. },
14. index:{
15. type:Number,
16. value:0
17. }
18. },
19. */\*\**
20. \* 组件的初始数据
21. \*/
22. data: {
23. shouldUnfold:false
24. },
25. */\*\**
26. \* 组件的方法列表
27. \*/
28. methods: {
29. checkDel(){
30. this.triggerEvent('del',this.properties.index)
31. },
32. unfold(){
33. this.setData({
34. shouldUnfold:!this.data.shouldUnfold
35. })
36. },
37. playCurrentVoice(){
38. console.log("playOutputVoice~" + this.data.outputText);
39. const generateVoice = require('../../utils/generateVoice.js');
40. generateVoice.GenerateVoice(this.data.outputText);
41. }
42. }
43. })

singleHistory.wxml

1. <block wx:if="{{shouldUnfold}}">
2. <view class="singleHistory\_big">
3. <view class="top" bind:tap="unfold">
4. </view>
5. <view class="mainBody">
6. <view class="textBody">
7. <text class="singleCurrentHistory">{{currentText}} </text>
8. <text class="singleOutputHistory">{{outputText}} </text>
9. </view>
10. <view class="buttonBody">
11. <image class="button" src="../../asset/pic/bf.png" mode="scaleToFill" bind:tap="playCurrentVoice" />
12. <image class="button" src="../../asset/pic/sc.png" mode="scaleToFill" bind:tap="checkDel" />
13. </view>
14. </view>
15. </view>
16. </block>
17. <block wx:else>
18. <view class="singleHistory\_normal" bind:tap="unfold">
19. <view class="textBody\_normal">
20. <text class="singleCurrentHistory\_normal" max-lines="{{1}}" overflow="clip">{{currentText}} </text>
21. <text class="singleOutputHistory\_normal" max-lines="{{1}}" overflow="clip">{{outputText}} </text>
22. </view>
23. </view>
24. </block>

singleHistory.scss：

1. .singleHistory\_normal {
2. height: 100rpx;
3. background-color: #e4e4e4;
4. margin-right: 20rpx;
5. margin-left: 20rpx;
6. margin-top: 5rpx;
7. border-radius: 15rpx;
8. }
9. .singleCurrentHistory\_normal {
10. height: 55rpx;
11. width: 650rpx;
12. margin-top: 5rpx;
13. overflow: hidden;
14. white-space: nowrap; *// 处理元素内的空白：不换行*
15. text-overflow: ellipsis; *//文本溢出处理方式：显示省略号*
16. font-size: larger;
17. }
18. .singleOutputHistory\_normal {
19. height: 35rpx;
20. width: 650rpx;
21. margin-bottom: 5rpx;
22. color: rgb(139, 139, 139);
23. font-size: medium;
24. overflow: hidden;
25. white-space: nowrap; *// 处理元素内的空白：不换行*
26. text-overflow: ellipsis; *//文本溢出处理方式：显示省略号*
27. }
28. .textBody\_normal{
29. display: flex;
30. flex-direction: column;
31. margin-top: 5rpx;
32. margin-bottom: 5rpx;
33. margin-right: 20rpx;
34. margin-left: 20rpx;
35. align-items: flex-start;
36. justify-content: center;
37. }
38. .singleHistory\_big {
39. max-height: 450rpx;
40. display: flex;
41. flex-direction: column;
42. background-color: #e4e4e4;
43. margin-right: 20rpx;
44. margin-left: 20rpx;
45. margin-top: 5rpx;
46. border-radius: 30rpx;
47. }
48. .top{
49. background-color: #e4e4e4;
50. border-radius:15rpx;
51. height: 80rpx;
52. }
53. .mainBody{
54. display: flex;
55. margin-top: 10rpx;
56. background-color: #f3f3f3;
57. flex-direction: column;
58. max-height: 350rpx;
59. }
60. .textBody{
61. display: flex;
62. flex:4;
63. flex-direction: column;
64. margin-top: 5rpx;
65. margin-bottom: 5rpx;
66. margin-right: 20rpx;
67. margin-left: 20rpx;
68. max-height: 300rpx;
69. }
70. .buttonBody{
71. flex:1;
72. margin-right: 20rpx;
73. margin-left: 20rpx;
74. margin-bottom: 5rpx;
75. display: flex;
76. gap: 50rpx;
77. max-height: 50rpx;
78. }
79. .button{
80. width: 40rpx;
81. height: 40rpx;
83. }

### pages/

#### pages/TransLink

翻译界面主要实现的功能有：

* 记录输入文本以及翻译后的输出文本
* 记录支持的语言以及输入与输出文本对应的语言
* 记录是否在录音状态
* 记录图片裁剪界面传递过来的图片URL

TransLink.js：

1. const app = getApp();
2. *//引入插件：微信同声传译*
3. const plugin = requirePlugin('WechatSI');
4. *//获取全局唯一的语音识别管理器recordRecoManager*
5. const manager = plugin.getRecordRecognitionManager();
6. Page({
7. */\*\**
8. \* 页面的初始数据
9. \*/
10. data: {
11. currentText: '',
12. outputText: '',
13. languageArr: [
14. "自动检测",
15. "中文",
16. "英语",
17. "西班牙语",
18. "法语",
19. "阿拉伯语",
20. "俄语",
21. "韩语",
22. "德语",
23. "日语",
24. "葡萄牙语",
25. "意大利语",
26. "荷兰语",
27. "泰语",
28. "希腊语",
29. "波兰语",
30. "捷克语",
31. "匈牙利语",
32. "瑞典语",
33. "芬兰语",
34. "保加利亚语",
35. "丹麦语",
36. "罗马尼亚语",
37. "斯洛文尼亚语",
38. "爱沙尼亚语",
39. "越南语",
40. "繁体中文",
41. "粤语",
42. "文言文",
43. ],
44. currentLanguage: 0,
45. targetLanguage: 1,
46. recordState: false,
47. imgURL: ''
48. },
49. */\*\**
50. \* 生命周期函数--监听页面加载
51. \*/
52. onLoad(options) {
53. this.initRecorder();
54. },
55. initRecorder() {
56. const that = this;
57. *// 正常开始录音识别时会调用此事件*
58. manager.onStart = function (res) {
59. console.log("成功开始录音识别")
60. }
61. *//识别结束事件*
62. manager.onStop = function (res) {
63. console.log("结束录音", res.result)
64. const re = res.result
65. that.setData({
66. currentText: re
67. })
68. }
69. manager.onRecognize = function (res) {
70. console.log(res.result)
71. }
72. },
73. */\*\**
74. \* 生命周期函数--监听页面显示
75. \*/
76. onShow() {
77. var app = getApp()
78. console.log(app.globalData.reservePhotoURL)
79. this.setData({
80. imgURL: app.globalData.reservePhotoURL
81. })
82. const pictureTranslate = require('../../utils/pictureTranslate.js');
83. const language = require('../../utils/language.js');
84. pictureTranslate.PictureTranslate(language.findLanguageCode(this.data.languageArr[this.data.currentLanguage]), language.findLanguageCode(this.data.languageArr[this.data.targetLanguage]), this,this.data.imgURL)
85. },
86. playCurrentVoice() {
87. console.log("playCurrentVoice~" + this.data.currentText)
88. const generateVoice = require('../../utils/generateVoice.js');
89. generateVoice.GenerateVoice(this.data.currentText);
90. },
91. confirmText(event) {
92. this.setData({
93. currentText: event.detail.value
94. })
95. },
96. translate() {
97. const generalTranslate = require('../../utils/generalTranslate.js');
98. const history = require('../../utils/history.js');
99. const language = require('../../utils/language.js');
100. generalTranslate.GeneralTranslate(this.data.currentText, language.findLanguageCode(this.data.languageArr[this.data.currentLanguage]), language.findLanguageCode(this.data.languageArr[this.data.targetLanguage]), this);
101. },
102. setCurrentLanguage(event) {
103. this.setData({
104. currentLanguage: event.detail.value
105. })
106. console.log(this.data.languageArr[this.data.currentLanguage])
107. },
108. setTargetLanguage(event) {
109. this.setData({
110. targetLanguage: event.detail.value
111. })
112. },
113. changeLanguage() {
114. const temp = this.data.currentLanguage
115. this.setData({
116. currentLanguage: this.data.targetLanguage
117. })
118. this.setData({
119. targetLanguage: temp
120. })
121. },
122. playOutputVoice() {
123. console.log("playOutputVoice~" + this.data.outputText);
124. const generateVoice = require('../../utils/generateVoice.js');
125. generateVoice.GenerateVoice(this.data.outputText);
126. },
127. acceptVoice() {
128. if (!this.data.recordState) {
129. manager.start()
130. this.setData({
131. recordState: true
132. })
133. } else {
134. manager.stop();
135. this.setData({
136. recordState: false
137. })
138. }
139. },
140. takePhoto() {
141. console.log("take a photo")
142. wx.chooseMedia({
143. count: 1,
144. mediaType: ['image'],
145. success(res) {
146. const photo = res.tempFiles[0].tempFilePath
147. wx.navigateTo({
148. url: `/pages/ClipPicture/ClipPicture?photoPath=${photo}`
149. })
150. }
151. })
152. },
153. setClipboardCur() {
154. wx.setClipboardData({
155. data: this.data.currentText
156. })
157. },
158. setClipboardOut() {
159. wx.setClipboardData({
160. data: this.data.outputText
161. })
162. }
163. })

TransLink.wxml：根据模块分为了输入文本框、语言选择框和输出文本框

1. *<!-- inputBox -->*
2. <view class="inputBoxBody">
3. <view class="inputBoxTextBody">
4. <textarea class="inputText" auto-focus confirm-type="done" placeholder="请输入文本" value="{{currentText}}" maxlength="100" bindinput="confirmText" />
5. </view>
6. <view class="inputBoxButton">
7. <view class="leftButton">
8. <image class="button" src="../../asset/pic/bf.png" mode="scaleToFill" bind:tap="playCurrentVoice" />
9. <image class="button" src="../../asset/pic/fz.png" mode="scaleToFill" bind:tap="setClipboardCur" />
10. </view>
11. <text class="text">{{currentText.length}}/100</text>
12. <view class="rightButton">
13. <image class="buttonBig" src="../../asset/pic/fy.png" mode="scaleToFill" bind:tap="translate" />
14. </view>
15. </view>
16. </view>
17. *<!-- selectLanguage -->*
18. <view class="selectBody">
19. <picker range="{{languageArr}}" value="{{currentLanguage}}" bindchange="setCurrentLanguage">
20. <text class="languageText">{{languageArr[currentLanguage]}}</text>
21. </picker>
22. <image class="selectButtonMid" src="../../asset/pic/qh.png" mode="scaleToFill" bind:tap="changeLanguage"></image>
23. <picker range="{{languageArr}}" value="{{targetLanguage}}" bindchange="setTargetLanguage">
24. <text class="languageText">{{languageArr[targetLanguage]}}</text>
25. </picker>
26. </view>
27. *<!-- outputBox -->*
28. <view class="outputBoxBody">
29. <view class="outputBoxTextBody">
30. <textarea class="outputText" value="{{outputText}}" auto-focus confirm-type="done" placeholder="翻译结果" disabled="true" />
31. </view>
32. <view class="inputBoxButton">
33. <view class="leftButton">
34. <image class="button" src="../../asset/pic/bf.png" mode="scaleToFill" bind:tap="playOutputVoice" />
35. <image class="button" src="../../asset/pic/fz.png" mode="scaleToFill" bind:tap="setClipboardOut" />
36. </view>
37. </view>
38. </view>
39. <view class="buttons">
40. <block wx:if="{{!recordState}}">
41. <image class="buttonItem" src="../../asset/pic/mkf.png" mode="scaleToFill" bind:tap="acceptVoice" />
42. </block>
43. <block wx:else>
44. <image class="buttonItem" src="../../asset/pic/bf.png" mode="scaleToFill" bind:tap="acceptVoice" />
45. </block>
46. <image class="buttonItem" src="../../asset/pic/xj.png" mode="scaleToFill" bind:tap="takePhoto" />
47. </view>

TransLink.scss：

1. .mainTexBody {
2. height: 200rpx;
3. display: flex;
4. flex-direction: row;
5. align-items: flex-end;
6. justify-content: space-around;
7. background-color: rgb(158, 214, 186, 0.3);
8. margin-right: 20rpx;
9. margin-left: 20rpx;
10. border-radius: 20rpx;
11. }
12. .mainTex {
13. height: 200rpx;
14. width: 200rpx;
15. }
16. .inputBoxBody {
17. display: flex;
18. height: 400rpx;
19. background-color: #e4e4e4;
20. margin-left: 20rpx;
21. margin-right: 20rpx;
22. margin-top: 10rpx;
23. margin-bottom: 10rpx;
24. border-radius: 30rpx;
25. flex-direction: column;
26. }
27. .inputBoxTextBody {
28. flex: 5;
29. margin-left: 20rpx;
30. margin-right: 20rpx;
31. margin-top: 10rpx;
32. margin-bottom: 10rpx;
33. flex-direction: row;
34. }
35. .inputBoxButton {
36. flex: 1;
37. display: flex;
38. margin-left: 20rpx;
39. margin-right: 20rpx;
40. margin-top: 10rpx;
41. margin-bottom: 10rpx;
42. align-items: center;
43. justify-content: space-around;
44. }
45. .leftButton {
46. display: flex;
47. flex: 3;
48. }
49. .rightButton {
50. display: flex;
51. flex: 1;
52. }
53. .button {
54. width: 50rpx;
55. height: 50rpx;
56. }
57. .buttonBig {
58. width: 70rpx;
59. height: 70rpx;
60. }
61. .inputText {
62. font-size: 35rpx;
63. }
64. .selectBody {
65. display: flex;
66. height: 80rpx;
67. background-color: #e4e4e4;
68. margin-left: 20rpx;
69. margin-right: 20rpx;
70. margin-top: 10rpx;
71. margin-bottom: 10rpx;
72. border-radius: 20rpx;
73. flex-direction: row;
74. align-items: center;
75. justify-content: space-around;
76. }
77. .languageText {
78. flex: 1;
79. font-size: 40rpx;
80. }
81. .selectButton {
82. width: 100rpx;
83. height: 50rpx;
84. }
85. .selectButtonMid{
86. width: 100rpx;
87. height: 50rpx;
88. position: absolute;
89. left:325rpx;
90. }
91. .outputBoxBody {
92. display: flex;
93. height: 400rpx;
94. background-color: #e4e4e4;
95. margin-left: 20rpx;
96. margin-right: 20rpx;
97. margin-top: 10rpx;
98. margin-bottom: 10rpx;
99. border-radius: 30rpx;
100. flex-direction: column;
101. }
102. .outputBoxTextBody {
103. flex: 5;
104. margin-left: 20rpx;
105. margin-right: 20rpx;
106. margin-top: 10rpx;
107. margin-bottom: 10rpx;
108. flex-direction: row;
109. }
110. .outputBoxButton {
111. flex: 1;
112. display: flex;
113. margin-left: 20rpx;
114. margin-right: 20rpx;
115. margin-top: 10rpx;
116. margin-bottom: 10rpx;
117. }
118. .leftButton {
119. display: flex;
120. flex: 3;
121. gap: 20rpx;
122. }
123. .text{
124. flex: 2;
125. font-size: large;
126. }
127. .rightButton {
128. display: flex;
129. flex-direction: column;
130. *// align-items: flex-end;*
131. *// justify-content:space-around;*
132. flex: 1;
133. }
134. .button {
135. width: 50rpx;
136. height: 50rpx;
137. }
138. .buttonBig {
139. width: 70rpx;
140. height: 70rpx;
141. }
142. .outputText {
143. font-size: 35rpx;
144. }
145. .buttons{
146. display: flex;
147. height: 200rpx;
148. flex-direction: row;
149. align-items:center;
150. justify-content: space-around;
151. }
152. .buttonItem{
153. height: 120rpx;
154. width: 120rpx;
155. }

#### pages/History

历史记录界面实现的功能主要有：

### utils/

#### utils/info.js

info.js主要记录有：

* API调用的appid与secertKey

1. //api相关id和密钥等
2. const appid = '20240428002037792';
3. const secretKey = 'ioGRpqn7V4hlCd\_z6VMZ';
4. module.exports = {
5. appid,
6. secretKey,
7. }

#### utils/language.js

language.js主要功能有：

* 记录语言设置的中文对应的代码编号
* 定义convert函数将输入的文本转换为对应的编号

1. //转换语言为代码
2. const languages = {
3. "auto": "自动检测",
4. "zh": "中文",
5. "yue": "粤语",
6. "wyw": "文言文",
7. "kor": "韩语",
8. "fra": "法语",
9. "th": "泰语",
10. "ara": "阿拉伯语",
11. "pt": "葡萄牙语",
12. "de": "德语",
13. "el": "希腊语",
14. "nl": "荷兰语",
15. "bul": "保加利亚语",
16. "est": "爱沙尼亚语",
17. "fin": "芬兰语",
18. "cs": "捷克语",
19. "slo": "斯洛文尼亚语",
20. "swe": "瑞典语",
21. "cht": "繁体中文",
22. "en": "英语",
23. "jp": "日语",
24. "spa": "西班牙语",
25. "ru": "俄语",
26. "it": "意大利语",
27. "pl": "波兰语",
28. "dan": "丹麦语",
29. "rom": "罗马尼亚语",
30. "hu": "匈牙利语",
31. "vie": "越南语"
32. };
33. function findLanguageCode(languageName) {
34. const entry = Object.entries(languages).find(([code, name]) => name === languageName);
35. return entry ? entry[0] : null;
36. }
37. function convertLanguageCode(inputCode) {
38. // 定义一个对象，将输入代码映射到标准语言代码
39. const codeMap = {
40. "zh": "CHN\_ENG",
41. "en": "ENG",
42. "jp": "JAP",
43. "kor": "KOR",
44. "fra": "FRE",
45. "spa": "SPA",
46. "pt": "POR",
47. "de": "GER",
48. "it": "ITA",
49. "ru": "RUS"
50. };
51. // 返回对应的标准语言代码，如果找不到则返回默认值"CHN\_ENG"
52. return codeMap[inputCode] || "CHN\_ENG";
53. }
54. module.exports = {
55. findLanguageCode,
56. convertLanguageCode,
57. }

#### utils/history.js

history.js主要功能有：

* 向内存中保存翻译历史记录
* 获得历史记录的信息，用于遍历历史记录
* 删除某一条历史记录
* 清空历史记录

1. // history相关功能
2. function setHistory(texInput, texOutput) {
3. let key = new Date().getTime().toString();
4. const tex = {
5. "input": texInput,
6. "output": texOutput,
7. };
8. SetTransHistory(texInput,texOutput,key)
9. wx.setStorageSync(key, tex);
10. }
11. function getHistoryInfo() {
12. wx.getStorageInfo({
13. success(res) {
14. console.log(res.keys)
15. console.log(res.currentSize)
16. console.log(res.limitSize)
17. }
18. })
19. }
20. function getHistory(key) {
21. wx.getStorage({
22. key: key,
23. success (res) {
24. console.log(res.data)
25. }
26. })
27. }
28. function clearHistory() {
29. wx.clearStorage({
30. success: res => {
31. console.log(res);
32. }
33. })
34. }
35. function removeHistory(key) {
36. wx.removeStorage({
37. key: key,
38. success(res) {
39. console.log(res)
40. }
41. });
42. }
43. function SetTransHistory(textInput,textOutput,key){
44. const app = getApp();
45. const HisArr=app.globalData.transHistory
46. const length=HisArr.length
47. HisArr.push({id:length,currentText:textInput,outputText:textOutput,key:key})
48. app.globalData.transHistory=HisArr
49. }
50. module.exports = {
51. setHistory,
52. getHistoryInfo,
53. getHistory,
54. clearHistory,
55. removeHistory,
56. }

#### utils/generateVoice.js

generateVoice.js主要功能有：

* 语音合成API调用
* 将获得的二进制文件写入内存中并播放

1. //语音合成（仅支持中英文）
2. function GenerateVoice(text) {
3. wx.request({
4. url: 'https://aip.baidubce.com/oauth/2.0/token?client\_id=MZNRF0KTHRh9jML40568xuKd&client\_secret=RzxUlh8V7fBYTenVr9tZRdecYLLBryZt&grant\_type=client\_credentials',
5. headers: {
6. 'Content-Type': 'application/json',
7. 'Accept': 'application/json'
8. },
9. success(data) {
10. wx.request({
11. url: 'http://tsn.baidu.com/text2audio',
12. responseType: 'arraybuffer',
13. data: {
14. tex: encodeURIComponent(text),
15. tok: data.data.access\_token,
16. cuid: '9zoLLvj1Wk7NSKPlsJ0pQQmWw7B8lSNx',
17. ctp: '1',
18. lan: 'zh',
19. },
20. success(msg) {
21. const wxAudio = wx.createInnerAudioContext({}); // 音频API
22. const fs = wx.getFileSystemManager(); // 文件管理器API
23. // 查看保存文件，（小程序开发工具右上角 “详情”->“基本信息”->“文件系统”）
24. const target = `${wx.env.USER\_DATA\_PATH}/${new Date().getTime()}.mp3`;
25. fs.writeFile({
26. filePath: target,
27. data: msg.data,
28. encoding: 'binary',
29. success: (res) => {
30. wxAudio.src = target
31. }
32. })
33. // onCanplay监听音频进入可以播放状态的事件
34. wxAudio.onCanplay(() => {
35. wxAudio.play()
36. })
37. }
38. })
39. }
40. })
41. }
42. module.exports = {
43. GenerateVoice,
44. }

#### utils/generalTranslate.js

generalTranslate.js主要功能有：

* 通用文本翻译API调用
* 实现了引擎层接口

1. //通用文本翻译
2. const md5 = require('./md5.js');
3. const info = require('./info.js')
4. const utils = require('./utils.js');
5. const history = require('./history.js');
6. function GeneralTranslate(q, from, to, Object) {
7. const url = 'https://fanyi-api.baidu.com/api/trans/vip/translate';
8. let salt = utils.generateRandomString(10);
9. let sign = info.appid + q + salt + info.secretKey;
10. sign = md5.MD5(sign);
11. wx.request({
12. url: url,
13. method: 'POST',
14. data: {
15. q: q,
16. from: from,
17. to: to,
18. appid: info.appid,
19. salt: salt,
20. sign: sign,
21. },
22. header: {
23. 'Content-Type': 'application/x-www-form-urlencoded',
24. },
25. success(data) {
26. console.log(data);
27. let tex = '';
28. for (let i = 0; i < data.data.trans\_result.length; i++) {
29. tex += data.data.trans\_result[i].dst + '\n';
30. }
31. Object.setData({
32. outputText: tex,
33. })
34. history.setHistory(q, tex);
35. history.getHistoryInfo();
36. }
37. })
38. }
39. module.exports = {
40. GeneralTranslate,
41. }

#### utils/pictureTranslate.js

pictureTranslate.js主要功能有：

* 图像识别功能API调用
* 识别图像中文字并显示在输入框中

1. //引入js
2. //翻译图片
3. function PictureTranslate(from, to, that, Photourl) {
4. const generalTranslate = require('./generalTranslate.js');
5. const language = require('./language.js');
6. wx.request({
7. url: 'https://aip.baidubce.com/oauth/2.0/token?client\_id=MZNRF0KTHRh9jML40568xuKd&client\_secret=RzxUlh8V7fBYTenVr9tZRdecYLLBryZt&grant\_type=client\_credentials',
8. headers: {
9. 'Content-Type': 'application/json',
10. 'Accept': 'application/json'
11. },
12. success: data => {
13. let url = 'https://aip.baidubce.com/rest/2.0/ocr/v1/general\_basic?access\_token=' + data.data.access\_token;
14. const fs = wx.getFileSystemManager();
15. fs.readFile({
16. filePath: Photourl,
17. encoding: 'base64',
18. success: msg => {
19. wx.request({
20. url: url,
21. method: 'POST',
22. header: {
23. 'Content-Type': 'application/x-www-form-urlencoded',
24. },
25. data: {
26. language\_type: language.convertLanguageCode(from),
27. image: msg.data,
28. },
29. success: data => {
30. let tex = '';
31. for (let i = 0; i < data.data.words\_result.length; i++) {
32. tex += data.data.words\_result[i].words + '\n';
33. }
34. that.setData({
35. currentText: tex,
36. })
37. // generalTranslate.GeneralTranslate(tex, from, to, that);
38. }
39. })
40. }
41. })
42. }
43. })
44. }
45. module.exports = {
46. PictureTranslate,
47. }

## 系统界面展示

### TransLink翻译界面

在翻译界面中

* 用户可以在输入文本框中输入想要翻译的文本，文本下方会显示当前字数以及字数上限
* 用户也可以点击下方的播放按钮播放输入文本对应的语音
* 用户也可以点击复制按钮复制输入文本到剪切板
* 用户输入完毕后，可点击翻译按钮进行翻译，小程序会默认自动检测输入文本的语言，并翻译为中文，显示在下方的输出文本框
* 用户也可以点击语言来唤出语言选择框，选择语言后，小程序会按照用户选择的语言进行翻译
* 用户也可以点击切换按钮交换源语言和目标语言
* 用户可以点击输出文本框中的播放与复制按钮，效果与输入文本框中的一致，但是作用对象变成了输出文本
* 用户点击麦克风按钮后，小程序会开始录音，再次点击麦克风按钮后，小程序会结束录音，并将语言识别结果自动显示在输入文本框中
* 用户可以点击照相机按钮进入ClipPicture界面，在该界面确认图片后，小程序会将图片识别结果显示在输入文本框中

图6- 1翻译界面

### ClipPicture图片裁剪界面

在裁剪界面中，用户可以方便地截取想要翻译的内容

* 用户可以拖动裁剪框选择需要翻译的内容
* 图片裁剪界面会根据裁剪框在界面中的位置与大小，灵活地对裁剪区域进行缩放，以便用户能够清晰地看到裁剪区域的内容
* 确认无误后，用户可以点击裁剪按钮确认裁剪，随后小程序将跳转到翻译界面，图片识别的内容也会填充到输入文本框中

图6- 2图片裁剪界面

### 历史记录界面

在历史界面，用户可以查看本设备上的历史记录，并且可以进行删除历史记录的操作

* 用户可以单级某条历史记录，该记录会展开，显示完整的输入输出文本，并且带有播放和删除按钮
* 如果用户不需要该条记录，可以点击删除按钮，再次点击消息提示框中的确定按钮后，即可删除该条历史记录

图6- 3删除单条历史记录

* 用户也可以点击右上角的多选按钮激活左侧的复选框
* 用户可以自行选择需要删除的历史记录，也可以点击上方的全选按钮全选或者全不选历史记录
* 选择好历史记录后，可以点击删除按钮删除选择的历史记录

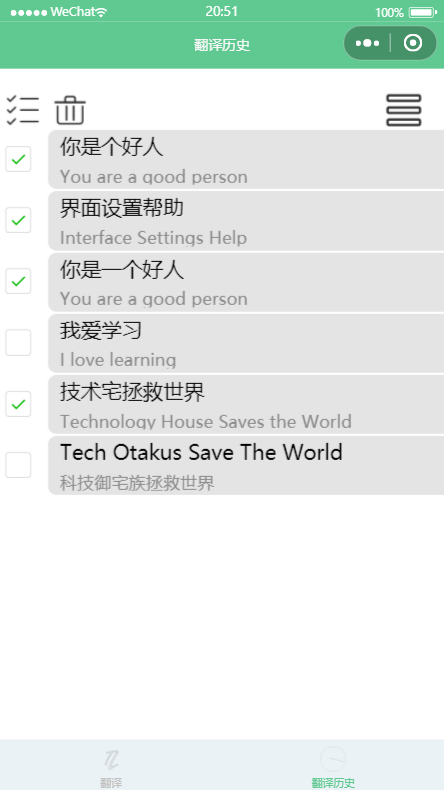
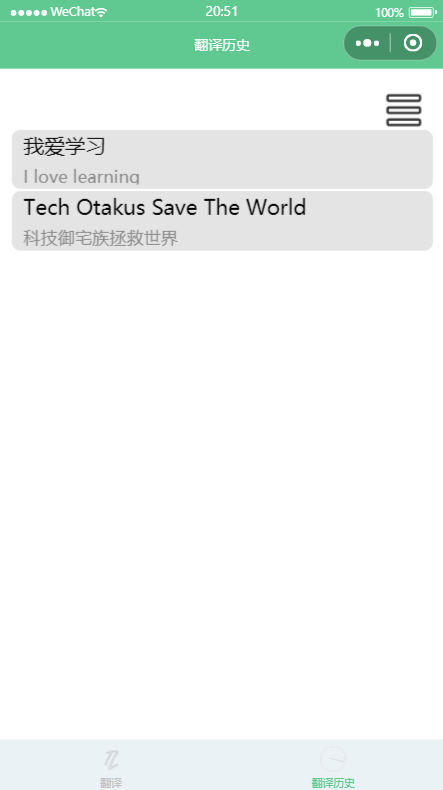
 

图6- 4删除多条历史记录

## 总结