

软件学院 2022-2023 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 2102 班
项目名称:	TransWe——机器翻译微信小程序
项目组组长:	张骁凯
项目组成员:	学号: U202117281 姓名: 张骁凯 _____ 学号: U202117254 姓名: 陈德霆 _____
指导老师:	刘小峰

目录

1. 项目概述	1
1.1 项目基本介绍.....	1
 UI 界面	1
 项目介绍	1
 项目成员	2
 功能特性	2
 目录结构描述	2
 版本内容更新	3
 协议	3
1.2 Github 仓库地址.....	3
1.3 人员基本分工.....	3
1.3.1 张骁凯 (负责人):	3
1.3.2 陈德霆 (副负责人):	4
2. 需求描述	5
2.1 功能性描述	5
2.1.1 用户需求.....	5
2.1.2 系统需求.....	5
2.2 非功能性需求.....	7
2.2.1 性能需求:	7
2.2.2 安全性需求:	7
2.2.3 可用性需求:	8

2.2.4 可维护性需求:	8
2.2.5 可扩展性需求:	8
2.2.6 用户体验需求:	8
3. 系统设计	9
3.1 架构设计	9
3.1.1 表示层 (Presentation Layer)	9
3.1.2 应用层 (Application Layer)	9
3.1.3 服务层 (Service Layer)	9
3.2 界面原型设计	10
3.2.1. 主页面 (Main Page)	10
3.2.2. 语言选择页 (Settings Page)	10
3.2.3. 语音翻译页 (Voice Translation Page)	10
3.2.4. OCR 拍照翻译页面 (OCR Translation Page)	10
3.2.5. 翻译历史页 (Translation History Page)	11
3.3 详细设计	11
3.3.1. 组件设计	11
3.3.2. 组件接口设计	11
3.3.3 系统流程分析	15
4. 系统实现和测试	18
4.1 系统实现	18
4.1.1 assets/	18
4.1.2 components/	18

4.1.3 imgs/	41
4.1.4 pages/	41
4.1.5 TDD_test_cdt/	93
4.1.6 TDD_test_zxk/	94
4.1.7 utils/	96
4.1.8 ./	102
4.2 系统测试	108
4.2.1 TDD 测试	108
5. 系统界面展示	111
5.1 index 主页	111
5.2 choose_language 选择语言界面	111
5.3 getPic 拍照界面	112
5.4 OCR 拍照翻译结果	113
5.5 voice_translation 语音翻译页面	113
5.6 edit 文本编辑界面	115
5.7 history 翻译历史	116
6. 总结	117

1. 项目概述

1.1 项目基本介绍



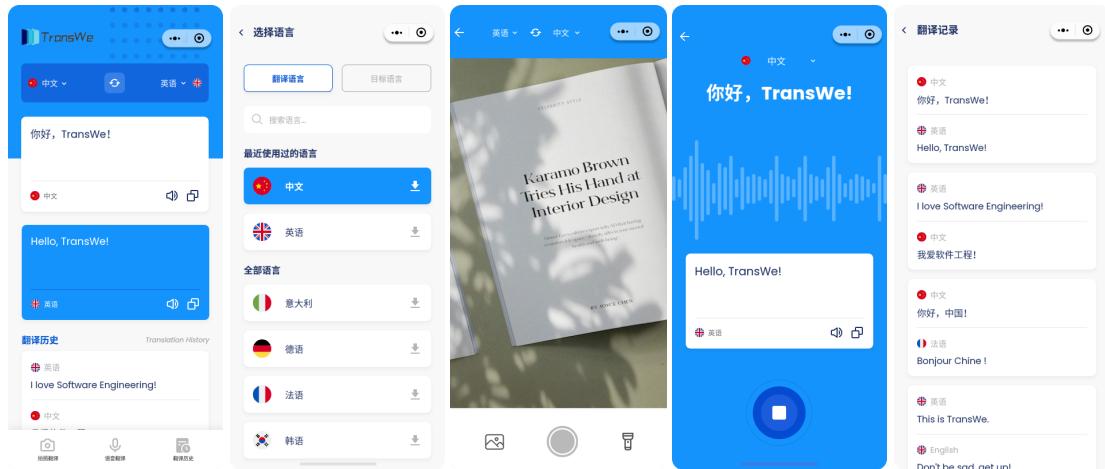
TransWe

Translate WeChat Mini Program

简体中文 | [English](#)

Author [dekt](#) Author [cdt](#) Stars 2 issues 4 open license [GPL-3.0](#)

UI界面



项目介绍

TransWe意为 `Translation+Wechat`，是一个功能强大的机器翻译微信小程序，它能够通过后台机器翻译服务快速、准确地翻译各种语言。它还支持第三方OCR、语音识别和语音合成集成，为用户提供更便捷、高效的翻译服务。

TransWe功能包括：

1. 机器翻译：TransWe使用后台机器翻译服务，支持多种语言翻译，包括英语、中文、法语、德语、日语、韩语等，能够准确、快速地翻译用户的文本。
2. OCR识别：TransWe支持第三方OCR识别，用户只需要上传图片或拍摄照片，就能将图片中的文字转换为文本进行翻译。
3. 语音识别：TransWe支持第三方语音识别，用户只需要录制音频，就能将音频中的语音转换为文本进行翻译。

4. 语音合成：TransWe支持第三方语音合成，用户能够将翻译结果通过语音合成功能转换为语音输出，提高用户的交互体验。

TransWe使用简单，功能强大，只需选择需要翻译的语言，输入要翻译的文本，点击“翻译”按钮，TransWe将自动完成翻译，如果需要使用OCR识别、语音识别或语音合成功能，可以选择相应功能按钮，按照提示进行操作即可。同时TransWe还是一款完全免费的小程序，旨在为用户提供更便捷、高效的翻译服务。无论是旅行、学习还是工作，TransWe都能帮助用户轻松应对语言难题。

🤝 项目成员



TransWe由张晓凯和陈德霆共同开发完成。

🚀 功能特性

TransWe是一款集多种功能于一身的微信小程序，旨在为用户提供便捷、高效的翻译服务。以下是TransWe的主要功能特性：

1. **多语言机器翻译**：TransWe支持多种语言的翻译，包括但不限于英语、中文、法语、德语、日语、韩语等。我们的后台机器翻译服务能够快速、准确地翻译用户的文本，满足用户在不同场景下的翻译需求。
2. **OCR识别**：TransWe集成了第三方OCR识别技术，用户只需上传图片或拍摄照片，我们的小程序就能将图片中的文字识别出来，转换为文本进行翻译。这一功能特别适用于处理图片中的外语文字，极大地提高了用户的翻译效率。
3. **语音识别**：TransWe支持第三方语音识别技术，用户只需录制音频，我们的小程序就能将音频中的语音识别并转换为文本进行翻译。这一功能使得用户在无法输入文字时，仍然可以轻松获取翻译服务。
4. **语音合成**：TransWe集成了第三方语音合成技术，用户可以将翻译结果转换为语音输出，提高了用户的交互体验，特别适用于视力不便或者需要听力辅助的用户。

TransWe的使用非常简单，用户只需选择需要翻译的语言，输入要翻译的文本，我们的小程序就会自动完成翻译。如果用户需要使用OCR识别、语音识别或语音合成功能，只需选择相应功能按钮，按照提示进行操作即可。

TransWe是一款完全免费的小程序，我们的目标是为用户提供最便捷、高效的翻译服务。无论是旅行、学习还是工作，TransWe都能帮助用户轻松应对语言难题，让语言交流变得无障碍。

☰ 目录结构描述

```
1 .
2   |-code                                //代码文件夹
3     |-assets                            //外部工具
4       \iconfont                         //矢量图库
5     |-components                        //组件文件夹
6       |-bottom-button                  //录音按钮
7       |-modal                           //复制按钮
8       |-play-icon                       //组件动画
9       |-result-bubble                 //翻译气泡
10      \waiting-icon                  //等待图标
11     |-imgs                             //小程序内部图片文件夹
```

```
12 |   |-pages //所有页面
13 |     |-choose_language //选择语言
14 |     |-edit //文本编辑页面
15 |     |-getPic //获取图片
16 |     |-history //翻译历史
17 |     |-history_test //翻译历史前端测试
18 |     |-index //主页
19 |     |-index_test //主页前端测试
20 |     |-OCR //拍照翻页界面
21 |     \-voice_translation //语音翻译界面
22 |   |-TDD_test_cdt //TDD开发语音翻译
23 |   |-TDD_test_zxk //TDD开发文本翻译
24 |   \-utils //插件
25 |     api.js //翻译api接口
26 |     conf.js //翻译配置
27 |     md5.min.js //获取MD5加密
28 |     util.js //获取时间配置
29 |-docs //所有文档
30 |   API.md //API接口文档
31 |   CurriculumDesignReport.md //课设报告文档
32 |   SystemArchitecture.md //系统架构文档
33 |   SystemDesign.md //系统设计文档
34 |   SystemRequirement.md //系统需求文档
35 |   UI_Design.md //UI设计文档
36 |   UserRequirement.md //用户需求文档
37 \-pics //图片
```

📅 版本内容更新

- 2023/06/02 TransWe v1.0: 基本实现全部功能

📃 协议

license [GPL-3.0](#)

警告

除GPLv3许可下的源代码外，其他方均禁止使用TransWe的名义作为下载器应用，TransWe的衍生产品亦同。

衍生品包括但不限于分叉和非官方构建。

1.2 Github仓库地址

[TransWe: https://github.com/dekrt/TransWe](https://github.com/dekrt/TransWe)

1.3 人员基本分工

1.3.1 张骁凯 (负责人) :

作为项目的负责人，张骁凯将承担项目的后端开发工作，工作内容丰富且富有挑战性，主要包括：

1. **负责Github仓库管理**: 张骁凯将负责整个Github仓库的管理工作，包括代码的整理、归档以及版本的管理。他还将负责编写详实的README文档，以便于其他团队成员以及可能接触项目的人对项目有明确的了解。

2. **将需求作为Issue录入**: 为了保证项目开发的流畅进行, 张晓凯将负责将所有的需求作为Issue录入Github, 这样可以保证每一个需求都能被详细跟踪, 从而高效地进行项目管理。
3. **共同进行系统设计**: 他将积极参与系统设计, 利用他的技术专长帮助设计出高效且易于维护的系统。
4. **用户界面设计**: 张晓凯将负责用户界面的设计和开发, 他会注重用户体验, 致力于创建出既易于使用又美观的用户界面。
5. **系统流程分析**: 他将通过绘制时序图进行系统流程分析, 这有助于更好地理解和设计系统。
6. **TDD测试**: 张晓凯将采用测试驱动开发 (TDD) 的方法, 为代码编写详尽的单元测试, 以确保代码的质量和稳定性。
7. **机器翻译服务**: 张晓凯将负责与后台机器翻译服务的接口开发和维护, 他将通过技术手段, 确保翻译服务的准确性和效率, 提供优质的用户体验。
8. **拍照翻译服务**: 他还将负责开发拍照翻译服务, 使用户能够通过拍照的方式获得翻译结果, 为用户提供更多的便利。
9. **前端页面编写**: 张晓凯将完成 index(小程序主页)、getPic(拍照界面)、OCR(获取翻译结果)、history(翻译历史)等页面的前端和后端代码编写, 他将负责实现用户界面设计, 同时也会对代码进行详细的测试, 以保证其正确性和稳定性。
10. **数据库管理**: 他将负责数据库的设计和管理, 通过合理的数据结构设计和索引优化, 确保用户数据的安全和完整。
11. **性能优化**: 张晓凯将专注于系统的性能优化, 通过合理的代码架构和算法优化, 他将确保系统在高并发情况下的稳定运行。
12. **编写文档**: 为了保证项目的可维护性和持续性, 他将负责编写清晰且详细的开发文档, 以便于后续的开发和维护工作。

1.3.2 陈德霆 (副负责人) :

陈德霆将主要负责项目的前端开发和用户体验设计, 包括但不限于:

1. **共同撰写需求**: 一起参与需求讨论和构思, 确保对项目的理解一致, 从而能够产生详尽而有用的项目需求文档。在撰写需求过程中, 努力保证每个要求都准确、清晰, 易于理解且可行。
2. **共同进行系统设计**: 在深入理解了项目需求之后, 参与到系统设计中来。确保每个设计决策都能满足项目的需求, 同时也会考虑到系统的可扩展性和可维护性。
3. **辅助用户界面设计**: 优化部分用户页面, 以用户为中心, 优化用户体验, 使其简单易用。
4. **系统设计**: 参与系统架构的设计, 负责组件设计以及组件接口的设计。
5. **语音合成功能**: 负责语音合成功能的开发和测试, 和相关页面集成, 确保这些功能的正常运行。
6. **组件开发**: 负责components中五个小组件的开发和测试, 每一个小组件都将被精心设计和编写, 以确保它们在整个系统中都能发挥关键的作用。
7. **页面编写**: 完成了choose_languege (语言选择页面)、edit (文本编辑页面)、voice_translation (语音翻译页面) 这几个页面的前端、后端代码编写, 实现了用户界面设计, 和界面的逻辑功能, 并进行了测试
8. **编写文档**: 包括但不限于设计文档、开发文档、测试文档。

2. 需求描述

2.1 功能性描述

2.1.1 用户需求

1 集成翻译服务 (Translation Service)

1. 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。

2. 用户需求标识: **TransWe-UR-TS**

2 集成第三方OCR功能 (Optical Character Recognition)

1. 用户需求:

- 用户可以选择使用图片转文字服务 (OCR)，并进行拍照（或选择图库中的图片）进行翻译。

2. 用户需求标识 : **TransWe-UR-OCR**

3 集成第三方语音识别(Speech Recognition):

1. 用户需求:

- 用户选择输入语言及目标语言，通过语音进行输入，程序以文字形式展示输入结果与翻译结果。

2. 用户需求标识: **TransWe-UR-SR**

4 集成第三方语音合成 (Speech Synthesis)

1. 用户需求:

- 用户在翻译完成后点击发声按钮，程序将翻译结果以语音的形式进行输出。

2. 用户需求标识: **TransWe-UR-SS**

2.1.2 系统需求

1 基础翻译功能 (TransWe-SR-TS)

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。

2. 正常状态:

- 用户在输入框中输入待翻译的文本，**程序会自动检测输入语言**，用户在下拉菜单中选择相应的目标语言。
- 用户也可以**手动选择输入语言**。输入完成后，程序将进行翻译并在输出框中显示翻译结果。

3. 有哪些会出错:

- 输入的文本中包含无法识别的字符或语言。程序会提示用户重新输入或手动选择语言。
- 输入的文本过长或复杂，程序无法进行翻译。程序会提示用户缩短输入文本或尝试其他翻译方式。
- 网络连接不稳定，程序无法进行翻译。程序会提示用户检查网络连接并重试。

4. 其他活动:

- 用户可以在下拉菜单中选择默认语言，程序会在下一次启动时自动选择该语言。
- 程序会记录用户的翻译历史，并允许用户在历史记录中查看以前的翻译结果。

5. 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。

2 集成第三方OCR功能的脚本 (TransWe-SR-OCR)

1. 初始假设:

- 用户需要使用一个微信翻译小程序，该小程序集成了第三方OCR功能，用户可以通过拍照或上传照片将图片中的待翻译文本识别成目标语言并显示到图片上。

2. 正常状态:

- 用户打开小程序，选择OCR功能，进入拍照界面。用户可以使用手机摄像头拍下待识别的图片，也可以按下图库按钮上传照片。进入图片编辑界面后，**用户可以选择整张图片或者框选一部分图片**，用户需要选择待翻译的语言种类和目标语言种类。检测到用户按下翻译按钮时，系统应该调用OCR服务对目标图片进行文字识别，并在界面上显示识别结果。
- 系统应该允许用户在翻译后重新选择图片，重新选择翻译语言并进行翻译。

3. 有哪些会出错:

- 系统权限不足，无法访问用户图库。
- 第三方OCR功能出现故障，导致无法完成文字识别。

4. 其他活动:

- 系统应该保证用户隐私，不记录用户的OCR识别记录。
- 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。

5. 完成的系统状态:

- 用户可以通过OCR功能成功识别图片中的文字并进行翻译。

3 语音识别功能 (TransWe-SR-SR)

1. 初始假设:

- 用户希望通过语音输入来输入翻译内容，系统需要进行语音识别功能，将语音转换为文本，再进行翻译操作。

2. 正常状态:

- 用户点击语音输入按钮，系统开始录音并将录音转换为文本格式，并输出到屏幕上。
- 系统对文本**进行分词**并进行翻译操作。
- 翻译结果以文本形式**呈现在界面上**。

3. 有哪些会出错:

- 语音输入的质量不好，无法被识别成文本。系统应该提示用户录音质量不好，请重试。

- 翻译服务不可用或异常。系统应该提示用户翻译服务暂时不可用，请稍后再试。

4. 其他活动：

- 系统应该保证用户隐私，不记录用户的语音输入内容。
- 系统应该对录音文件进行保护，确保不被未授权的人访问。

5. 完成的系统状态：

- 用户可以通过语音输入方式进行翻译操作。
- 系统可以对语音进行识别并将其转换为文本格式。
- 系统可以对文本进行分词和翻译操作，将翻译结果呈现在界面上。

4 语音合成功能 (TransWe-UR-SS)

1. 初始假设：

- 用户希望通过语音的形式来输出翻译内容，系统需要进行语音合成功能，将文本转换为语音，再通过扬声器播放。

2. 正常状态：

- 用户正常进行翻译后，点击语音输出按钮，系统开始进行语音合成并将翻译结果以**语音的形式进行输出**。
- 翻译结果以**文本形式呈现在界面上**。

3. 有哪些会出错：

- 系统语音合成功能出现故障，无法将文本正确转换为语音。
- 扬声器或音频设备出现故障，无法正常播放语音。

4. 其他活动：

- 当正在播放合成的语音时，图标的颜色将会改变；播放完成后回到原来的颜色。

5. 完成的系统状态：

- 用户可以通过语音输入并输出翻译内容，系统可以将文本转换为语音，并通过扬声器播放出来。系统记录了用户的输入和输出内容，并可以对其进行分析和统计。

2.2 非功能性需求

2.2.1 性能需求：

- 翻译响应时间：对于用户输入的文本，系统应在1秒内返回翻译结果。
- OCR识别和语音识别的处理时间：对于用户上传的图片或音频，系统应在5秒内完成识别并返回结果。
- 系统应能够支持高并发请求，即在用户量剧增的情况下，系统的性能不会显著下降。

2.2.2 安全性需求：

- 用户的个人信息和使用数据应得到充分保护，不得泄露给第三方。
- 系统应具备防止恶意攻击的能力，如DDoS攻击、SQL注入等。

2.2.3 可用性需求:

- 系统的正常运行时间应达到99.9%。
- 在出现故障时，系统应能在1小时内恢复正常。

2.2.4 可维护性需求:

- 系统应具备良好的模块化和文档化，以便进行维护和升级。
- 系统应能够容易地添加新的语言支持和新的功能。

2.2.5 可扩展性需求:

- 系统应设计成可扩展的架构，以便在未来可以添加更多的功能，如多语言支持、语音翻译等。

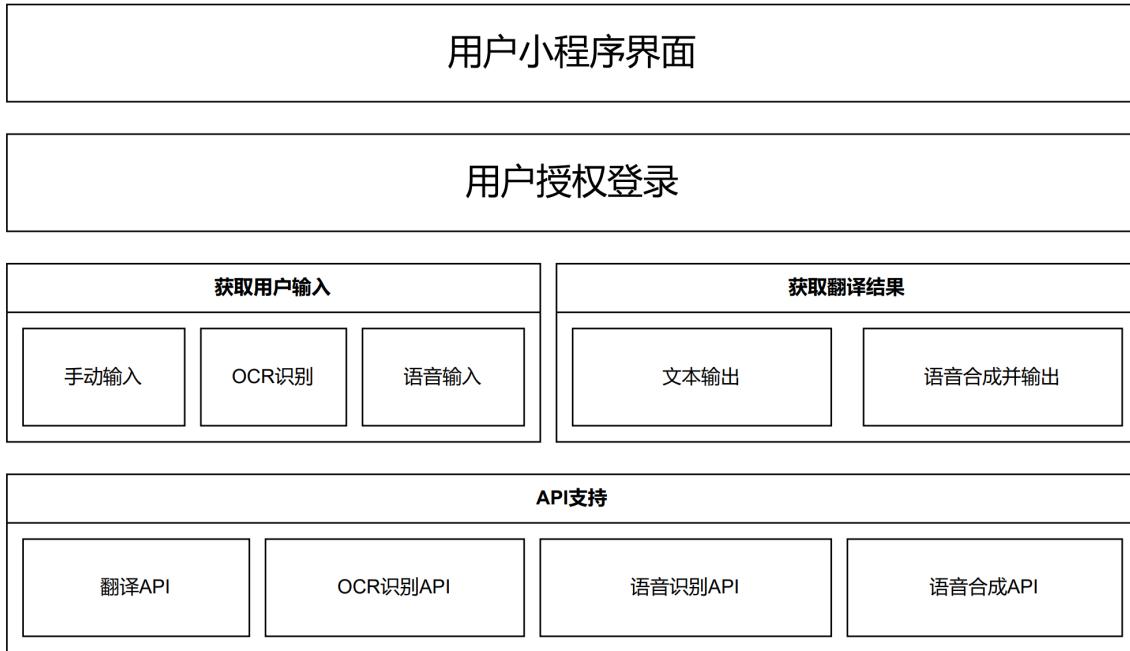
2.2.6 用户体验需求:

- 系统的用户界面应简洁易用，用户能够快速理解如何使用各项功能。
- 系统应提供用户反馈功能，用户可以方便地报告问题和提出建议。

3. 系统设计

3.1 架构设计

我们的翻译小程序采用分层架构的架构模式，架构示意图如下：



3.1.1 表示层 (Presentation Layer)

表示层是用户与系统交互的界面，包括用户界面、数据输入、输出等。这一层主要负责将用户请求传递给下一层，并将处理结果返回给用户。在我们的翻译软件中，表示层包括用户输入文本的界面、显示翻译结果的界面、对翻译结果进行输出的页面等，以及获取用户的授权信息。

3.1.2 应用层 (Application Layer)

应用层是系统的核心层，它实现了翻译的核心算法和业务逻辑，包括文本处理、翻译算法等。这一层主要负责接收并处理表示层传递的请求，然后调用其他层的服务，最后将处理结果返回给表示层。在我们的小程序中，应用层负责：

1. 获取用户输入：
 - 手动输入
 - OCR识别
 - 语音输入
2. 获取输出结果：
 - 文本输出
 - 语音合成输出

3.1.3 服务层 (Service Layer)

服务层为应用层提供支持，包括网络通信、数据访问、存储等服务。这一层主要负责处理数据的存储和访问，以及与其他系统的交互。在我们的翻译软件中，服务层可以包括调用翻译接口获取翻译结果、调取OCR接口获取OCR识别结果、调取语音合成API对翻译结果进行语音输出等。

总的来说，以上三个层级构成了一个完整的翻译软件系统，每个层级都负责不同的功能，各司其职。这种分层架构模式使得系统更加清晰、易于扩展和维护。

3.2 界面原型设计

3.2.1. 主页面 (Main Page)

主页是小程序的入口，包括小程序的基本信息和主要功能模块。页面包含四个模块，分别是文字翻译，录音按钮、OCR翻译按钮和翻译历史记录。

- **文本输入框**: 用户可以在这个框中手动输入需要翻译的文本。
- **翻译框**: 用户输入文本、点击这个按钮开始翻译。
- **语音播放器**: 用户点击播放语音。
- **语言选择器**: 用户可以在这里选择源语言和目标语言。
- **语音按钮**: 用户可以点击这个按钮，跳转到语音翻译页面。
- **OCR按钮**: 用户可以点击这个按钮，跳转到语音OCR页面。
- **翻译历史按钮**: 用户可以点击这个按钮，跳转到翻译历史页面。

3.2.2. 语言选择页 (Settings Page)

语言选择页提供用户多种翻译语言。用户可以选择支持的语言。

- **语言设置**: 用户可以在这里更改默认的源语言和目标语言。

3.2.3. 语音翻译页 (Voice Translation Page)

语音翻译页是用户输入需要翻译的语音的页面。页面主要包括录音按钮、语言选择器和翻译结果框。翻译结果框会以卡片形式保存下来，用户可以编辑录入的文字。

- **录音按钮**: 位于页面下方，用户可以点击此按钮开始进行语音输入，输入的语音将被实时转化为文字并显示在结果框内。
- **语言选择器**: 位于页面顶部，用户可以在此一键切换中英文。
- **翻译结果框**: 显示用户录音输入的文字以及翻译结果。翻译结果以卡片形式保存并展示，每个卡片包括原文和译文，用户可以删除不要的卡片。
- **编辑按钮**: 每个翻译结果卡片右上角都会有一个编辑按钮，用户点击后可以对录入的文字进行编辑，编辑完成后翻译结果将自动更新。
- **返回按钮**: 页面左上角有一个返回按钮，用户点击后可以返回主页面。

3.2.4. OCR拍照翻译页面 (OCR Translation Page)

OCR拍照翻译页面是用户能够通过拍照进行翻译的地方。

- **拍照按钮**: 用户可以点击这个按钮，利用手机相机进行拍摄，完成拍摄后，系统会自动进行识别并将图片中的文字进行翻译。
- **翻译结果展示区**: 系统完成翻译后，翻译结果将会在这个区域显示，用户可以查看翻译结果。

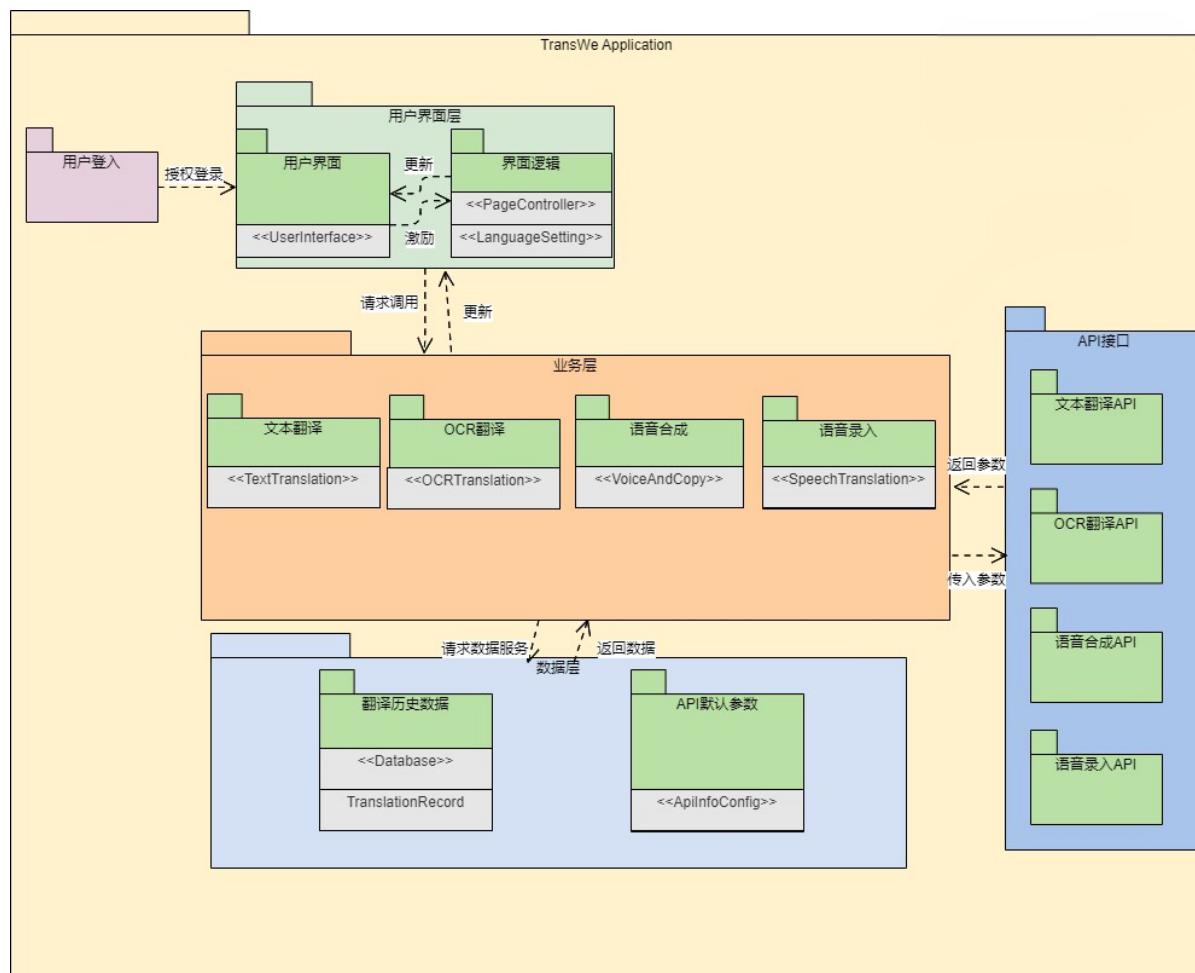
3.2.5. 翻译历史页 (Translation History Page)

翻译历史页面保存用户之前的文字翻译和语音翻译记录。

- **翻译历史**: 用户可以划动屏幕查询所有的本地翻译记录。每条历史以卡片形式保存并展示，每个卡片包括原文和译文。

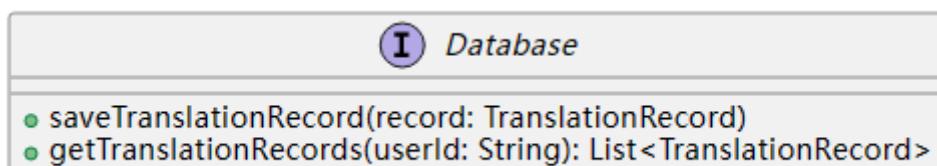
3.3 详细设计

3.3.1. 组件设计



3.3.2. 组件接口设计

1 数据库接口



- `saveTranslationRecord(record: TranslationRecord): void` : 保存翻译记录。
- `getTranslationRecords(userId: String): List<TranslationRecord>` : 获取指定用户的翻译记录列表。

2 语言设置接口

I	LanguageSetting
sourceLanguage: String	
targetLanguage: String	
setSourceLanguage(sourceLanguage: String)	
setTargetLanguage(targetLanguage: String)	
getSourceLanguage(): String	
getTargetLanguage(): String	

- `sourceLanguage: String`: 源语言属性。
- `targetLanguage: String`: 目标语言属性。
- `setSourceLanguage(sourceLanguage: String): void`: 设置源语言。
- `setTargetLanguage(targetLanguage: String): void`: 设置目标语言。
- `getSourceLanguage(): String`: 获取源语言。
- `getTargetLanguage(): String`: 获取目标语言。

3 OCR翻译接口

I	OCRTranslation
recognizeImage(image: ImageData, sourceLanguage: String, targetLanguage: String): String	

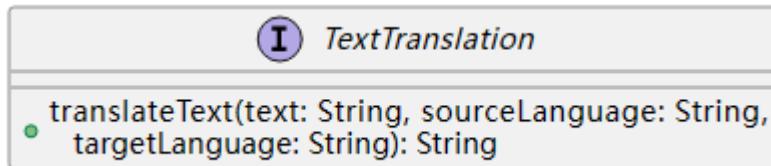
- `recognizeImage(image: ImageData, sourceLanguage: String, targetLanguage: String): String`: 将图像数据识别为文本，并将其翻译成指定的目标语言。

4 语音翻译接口

I	SpeechTranslation
recognizeSpeech(audio: AudioData, sourceLanguage: String, targetLanguage: String): String	

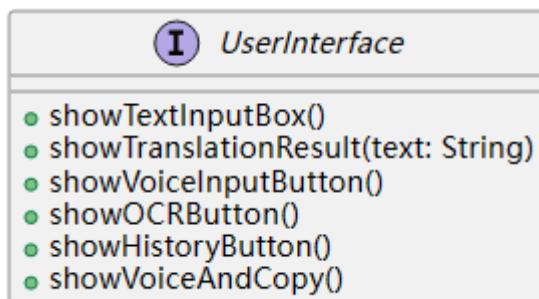
- `recognizeSpeech(audio: AudioData, sourceLanguage: String, targetLanguage: String): String`: 将音频数据识别为文本，并将其翻译成指定的目标语言。

5 文本翻译接口



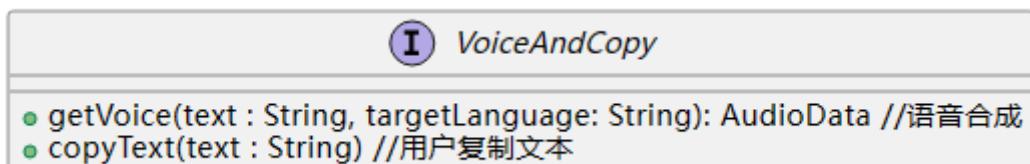
- `translateText(text: String, sourceLanguage: String, targetLanguage: String): String`：将指定的文本翻译成指定的目标语言。

6 用户界面接口



- `showTextInputBox(): void`：显示文本输入框。
- `showTranslationResult(text: String): void`：显示翻译结果。
- `showVoiceInputButton(): void`：显示语音输入按钮。
- `showOCRButton(): void`：显示 OCR 按钮。
- `showHistoryButton(): void`：显示历史记录按钮。
- `showVoiceAndCopy(): void`：显示语音合成和复制按钮。

7 语音录入接口



- `getVoice(text: String, targetLanguage: String): AudioData`：将指定的文本转换为语音，并返回音频数据。
- `copyText(text: String): void`：将指定的文本复制到剪贴板。

8 页面控制接口



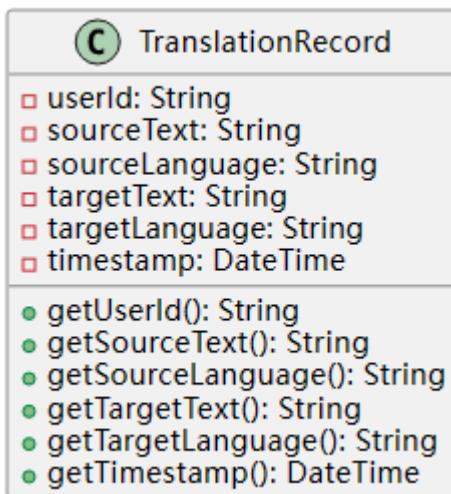
- `+getToPage(page: String)`: 表示该接口具有一个公共方法 `getToPage`，该方法接受一个参数 `page`，类型为 `String`，用于获取指定页面的内容。

9 API信息配置接口



- `-api_config: ApiConfig`: 表示该接口具有一个私有属性 `api_config`，其类型为 `ApiConfig`。私有属性只能在该类内部访问。
- `+setApiConfig(api_config : ApiConfig)`: 表示该接口具有一个公共方法 `setApiConfig`，该方法接受一个参数 `api_config`，类型为 `ApiConfig`，用于设置 `api_config` 的值。
- `+getApiConfig(): ApiConfig`: 表示该接口具有一个公共方法 `getApiConfig`，该方法返回 `api_config` 的值，类型为 `ApiConfig`。

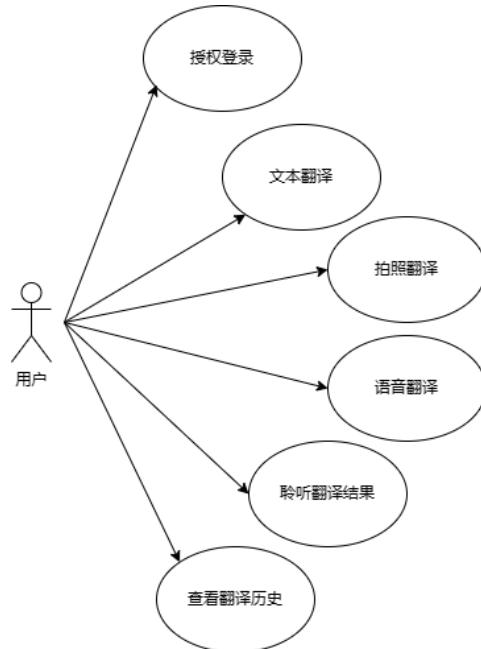
10 翻译记录



- `userId: String`: 用户 ID 属性。
- `sourceText: String`: 源文本属性。
- `sourceLanguage: String`: 源语言属性。
- `targetText: String`: 目标文本属性。
- `targetLanguage: String`: 目标语言属性。

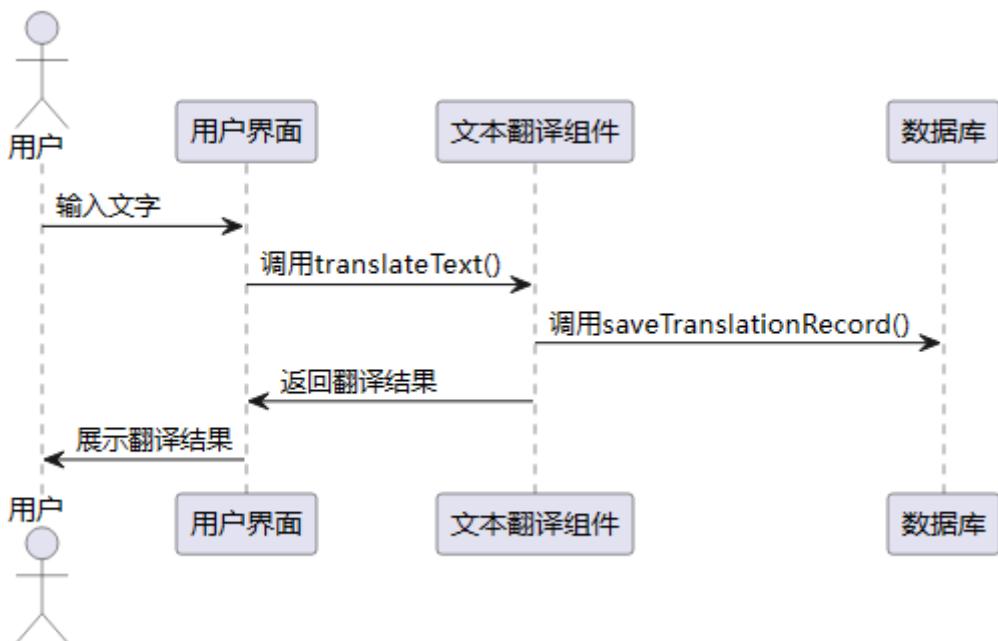
- `timestamp: DateTime`: 时间戳属性。
- `getUserId(): String`: 获取用户 ID。
- `getSourceText(): string`: 获取源文本。
- `getSourceLanguage(): String`: 获取源语言。
- `getTargetText(): string`: 获取目标文本。
- `getTargetLanguage(): String`: 获取目标语言。
- `getTimestamp(): DateTime`: 获取时间戳。

3.3.3 系统流程分析

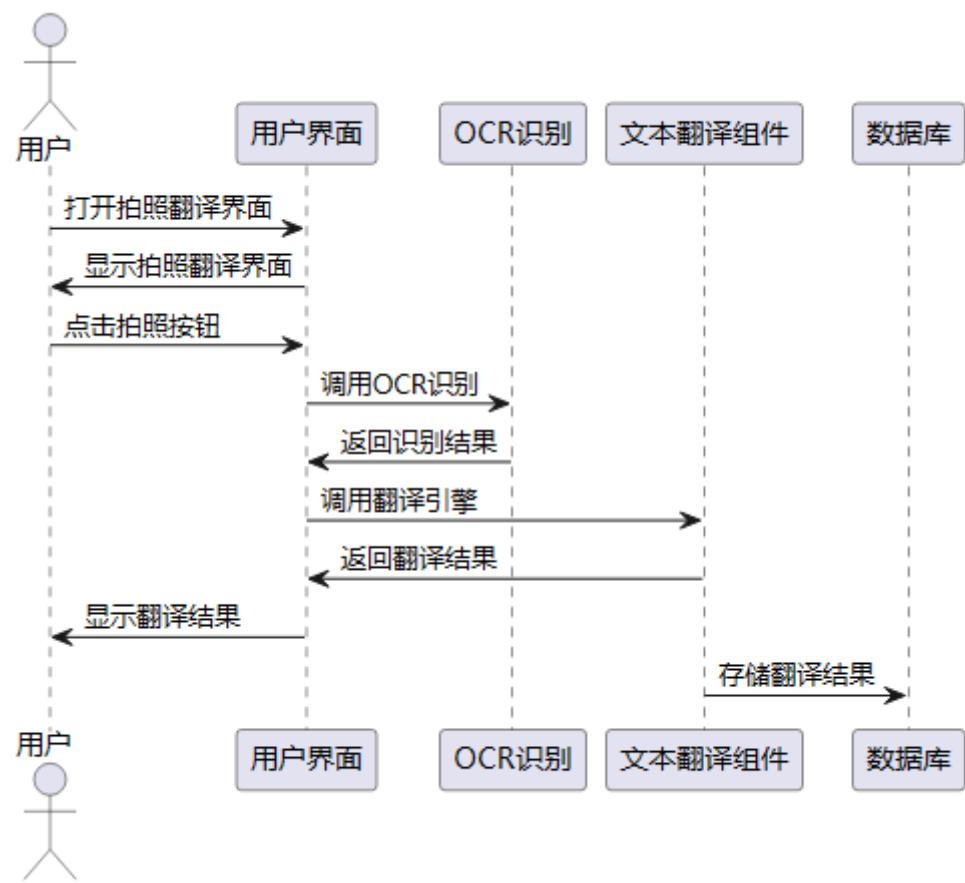


结合上述用例图，我们得出以下的时序图：

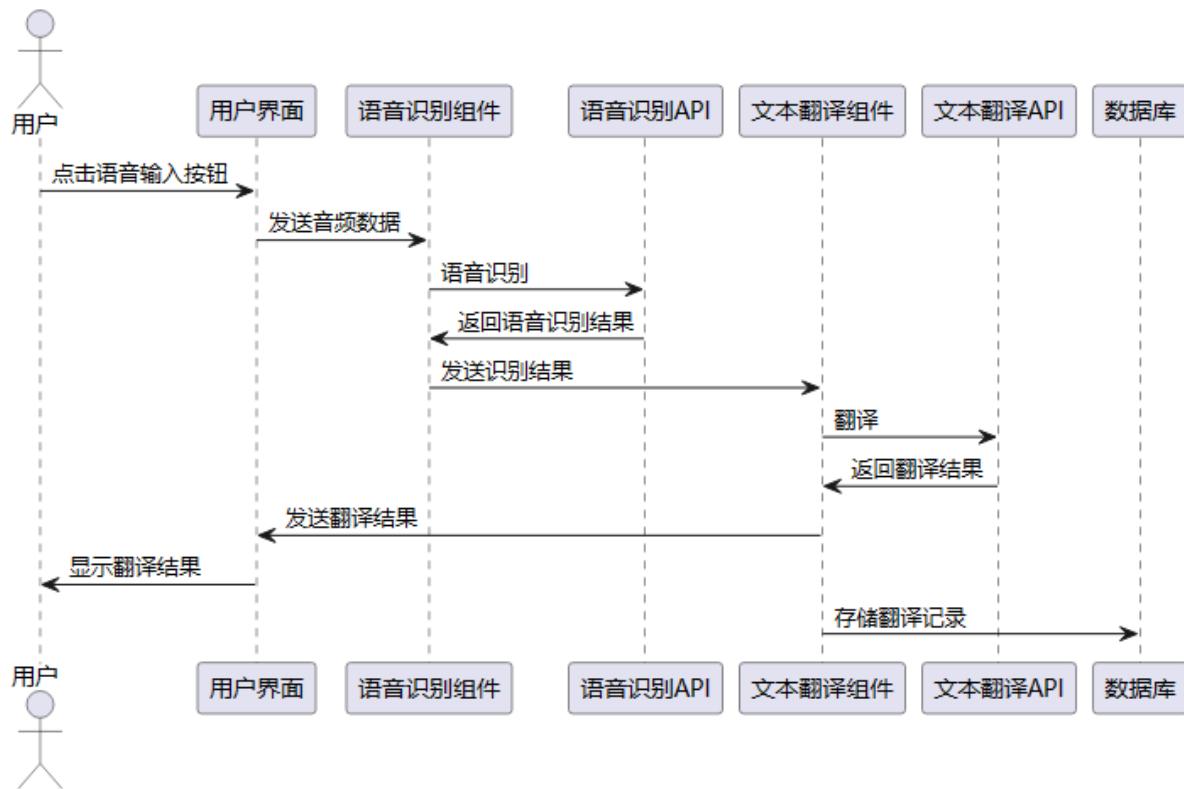
1 文本翻译时序图



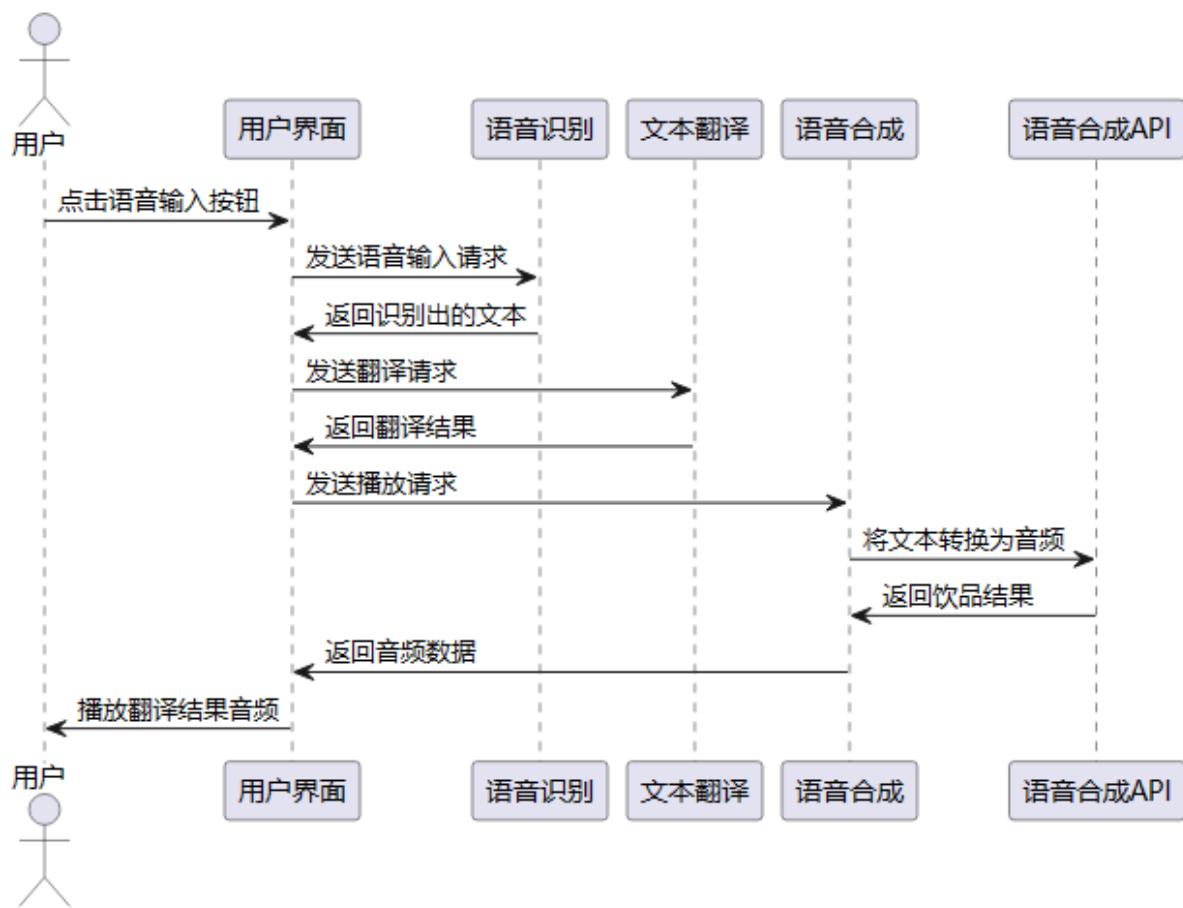
2 拍照翻译时序图



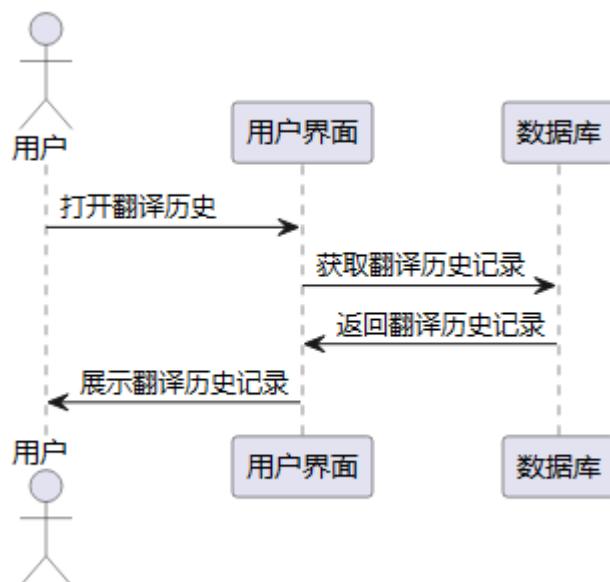
3 语音翻译时序图



4 语音合成时序图



5 查看翻译历史时序图



4. 系统实现和测试

4.1 系统实现

开发语言: Html(Wxml) + Css(Wxss) + javascript;

开发框架: 微信原生框架 + Less

开发环境: 微信开发者工具 + Vscode

4.1.1 assets/

4.1.1.1 assets/iconfont.wxss

这段代码在全局在全局引用了Iconfont的Icon图标。

```
1 @font-face {font-family: "iconfont";
2   src: url('https://at.alicdn.com/t/font_811118_f7oh8iao9yd.ttf')
3   format('truetype')
4 }
5
6 .iconfont {
7   font-family:"iconfont" !important;
8   font-size:16px;
9   font-style:normal;
10  -webkit-font-smoothing: antialiased;
11  -moz-osx-font-smoothing: grayscale;
12 }
13 .icon-down:before { content: "\e600"; }
14
15 .icon-close:before { content: "\e78f"; }
16
17 .icon-arrow-right:before { content: "\e682"; }
18
19 .icon-duihao:before { content: "\e601"; }
20
21 .icon-right:before { content: "\e790"; }
22
```

4.1.2 components/

翻译组件介绍(语音翻译页面配置)

4.1.2.1 components/bottom-button

录音按钮组

以下代码主要功能是:

1. 根据语言配置生成对应的录音按钮。
2. 当按下按钮时, 开始录音, 并将按钮样式改为按下状态。
3. 当松开按钮时, 结束录音, 并将按钮样式改回正常状态。

4. 当按钮被禁用时，修改按钮样式为禁用状态。

- **bottom-button.js**

```
1 // 引入语言配置文件
2 import { language } from '../../utils/conf.js'
3
4 // 获取全局应用实例
5 let app = getApp();
6
7 // 获取全局按钮配置
8 let buttons = app.globalData.buttons;
9
10 // 根据语言配置生成按钮配置
11 language.forEach(item => {
12     buttons.push({
13         buttonText: item.lang_name, // 按钮文本
14         lang: item.lang_content, // 按钮对应的语言
15         lto: item.lang_to[0], // 目标语言
16         msg: item.hold_talk, // 提示信息
17         buttonType: 'normal', // 按钮类型
18     })
19 })
20
21 // 按钮对应的背景图片
22 let buttonBackground = {
23     zh_CN: {
24         normal: '../../imgs/R.png',
25         press: '../../imgs/R1.png',
26         disabled: '../../imgs/R.png',
27     },
28     en_US: {
29         normal: '../../imgs/R.png',
30         press: '../../imgs/R1.png',
31         disabled: '../../imgs/R.png',
32     }
33 }
34
35 // 组件定义
36 Component({
37     // 组件的属性列表
38     properties: {
39         // 按钮是否禁用
40         buttonDisabled: {
41             type: Boolean,
42             value: false,
43             observer: function (newVal, oldVal) {
44                 let buttonType = newVal ? 'disabled' : 'normal'
45                 this.changeButtonType(buttonType)
46             }
47         },
48     },
49
50     // 组件的初始数据
51     data: {
```

```

52     buttons: buttons, // 按钮配置
53     buttonBackground: buttonBackground, // 按钮背景图片
54     currentButtonType: 'normal', // 当前按钮类型
55   },
56
57   // 组件的方法列表
58   methods: {
59     // 按下按钮开始录音
60     streamRecord(e) {
61       if (this.data.buttonDisabled) {
62         return
63       }
64       // 先清空背景音
65       wx.stopBackgroundAudio()
66
67       let currentButtonConf = e.currentTarget.dataset.conf
68
69       this.changeButtonType('press', currentButtonConf.lang)
70
71       this.triggerEvent('recordstart', {
72         buttonItem: currentButtonConf
73       })
74     },
75
76     // 松开按钮结束录音
77     endStreamRecord(e) {
78       let currentButtonConf = e.currentTarget.dataset.conf
79
80       this.triggerEvent('recordend', {
81         buttonItem: currentButtonConf
82       })
83     },
84
85     // 修改按钮样式
86     changeButtonType(buttonType, buttonLang) {
87       let tmpButtons = this.data.buttons.slice(0)
88
89       tmpButtons.forEach(button => {
90         if (!buttonLang || buttonLang == button.lang) {
91           button.buttonType = buttonType
92         }
93       })
94
95       this.setData({
96         buttons: tmpButtons
97       })
98     },
99   }
100 });
101

```

- bottom-button.json

```
1  {
2      "component": true
3 }
```

- **bottom-button.wxml**

```
1 <!-- 按钮组容器，当hidden为true时隐藏 -->
2 <view class="button-wrap" hidden="{{hidden}}>
3     <!-- 图片大容器 -->
4     <view class="img-big-wrap">
5         <!-- 按钮容器 -->
6         <view class="button-container">
7             <!-- 使用wx:for指令遍历buttons数组，生成对应的按钮 -->
8             <view wx:for="{{buttons}}" wx:for-item="button" wx:key="lang"
9                 class="button-item">
10                <view catchtouchstart="streamRecord"
11                    catchtouchend="endStreamRecord"
12                    data-conf="{{button}}"
13                    class="button-press">
14                    <image class="button-background" src="
15                        {{buttonBackground[button.lang][button.buttonType]}}></image>
16                </view>
17            </view>
18        </view>
19    </view>
```

- **bottom-button.wxss**

```
1 /* 按钮组容器样式，使用flex布局，内容居中 */
2 .button-wrap {
3     display: -webkit-flex;
4     display: flex;
5     -webkit-justify-content: center;
6     justify-content: center;
7 }
8
9 /* 图片大容器样式，宽度100%，使用flex布局，背景色为#1494fc */
10 .img-big-wrap {
11     width: 100%;
12     display: -webkit-flex;
13     display: flex;
14     background: #1494fc;
15 }
16
17 /* 按钮容器样式，使用flex布局，高度100%，宽度100%，内容居中，底部外边距20px */
18 .button-container{
19     display: flex;
20     display: -webkit-flex;
21     height: 100%;
22     width: 100%;
23     box-sizing: border-box;
24     justify-content: space-between;
25     -webkit-justify-content: space-between;
```

```
26 align-items: center;
27 -webkit-align-items:flex-start;
28 justify-content: center;
29 margin-bottom: 20px;
30 padding: 50rpx 0 38rpx 0;
31 z-index: 1;
32 }
33
34 /* 按钮项样式，使用flex布局，方向为列，内容从上开始，居中对齐，宽度75px */
35 .button-item {
36 display: flex;
37 display: -webkit-flex;
38 flex-direction: column;
39 -webkit-flex-direction: column;
40 justify-content: flex-start;
41 -webkit-justify-content: flex-start;
42 align-items: center;
43 -webkit-align-items: center;
44 width: 75px;
45 box-sizing: border-box;
46 z-index: 2;
47 }
48
49 /* 按钮标签样式，字体大小28rpx，颜色#9B9B9B，字母间距0，上外边距15rpx */
50 .button-label {
51 font-size: 28rpx;
52 color: #9B9B9B;
53 letter-spacing: 0;
54 margin: 15rpx 0 0 0;
55 }
56
57 /* 按钮按下样式，位置相对，使用flex布局，高度150rpx，宽度100%，圆角100rpx，内容居中对齐 */
58 .button-press {
59 position: relative;
60 display: flex;
61 display: -webkit-flex;
62 height: 150rpx;
63 width: 100%;
64 border-radius: 100rpx;
65 justify-content: center;
66 -webkit-justify-content: center;
67 align-items: center;
68 -webkit-align-items:center;
69 }
70
71 /* 按钮背景样式，位置相对，高度150rpx，宽度100%，左边距0，z-index为3 */
72 .button-background {
73 position: relative;
74 height: 150rpx;
75 width: 100%;
76 /* border-radius: 100rpx; */
77 left: 0;
78 z-index: 3;
79 }
```

4.1.2.2 components/modal

工具组件

以下代码主要功能是：

1. 三个操作项：复制源文本、复制目标文本和删除条目。
2. 当点击复制源文本或复制目标文本时，会调用 setclip

- **modal.js**

```
1 // 导入语言配置
2 import { language } from '../../utils/conf.js'
3
4 // 获取第一种语言配置
5 const tips_language = language[0]
6
7 // 定义模态框中的操作项
8 let modalItems = [
9   {
10     type: 'copySource', // 复制源文本
11     text: tips_language.copy_source_text
12   },
13   {
14     type: 'delete', // 删除条目
15     text: tips_language.delete_item
16   },
17   {
18     type: 'copyTarget', // 复制目标文本
19     text: tips_language.copy_target_text
20   },
21 ]
22
23 // 定义组件
24 Component({
25   // 组件的属性列表
26   properties: {
27     // 条目数据
28     item: {
29       type: Object,
30       value: {},
31     },
32     // 是否显示模态框
33     modalShow: {
34       type: Boolean,
35       value: true,
36     },
37     // 条目索引
38     index: {
39       type: Number,
40     },
41   },
42
43   // 组件的初始数据
44   data: {
45     modalItems: modalItems, // 模态框操作项
46   }
47 }
```

```
46 },
47
48 // 组件的方法列表
49 methods: {
50   // 删除条目并关闭模态框
51   deleteBubbleModal: function() {
52     this.triggerEvent('modaldelete', {
53       item: this.data.item,
54       index: this.data.index,
55     }, { bubbles: true, composed: true })
56     this.leaveBubbleModal()
57   },
58
59   // 点击操作项
60   itemTap: function(e) {
61     let itemType = e.currentTarget.dataset.type
62     let item = this.data.item
63
64     switch(itemType) {
65       case 'copySource': // 复制源文本
66         this.setClip(item.text)
67         break;
68       case 'copyTarget': // 复制目标文本
69         this.setClip(item.translateText)
70         break
71       case 'delete': // 删除条目
72         this.deleteBubbleModal()
73         break
74       default:
75         break
76     }
77   },
78
79   // 复制到剪贴板
80   setClip: function(text) {
81     wx.setClipboardData({
82       data: text,
83       success: (res) => {
84         this.leaveBubbleModal()
85         wx.showToast({
86           title: "已复制到剪切板",
87           icon: "success",
88           duration: 1000,
89           success: function (res) {
90             console.log("show succ");
91           },
92           fail: function (res) {
93             console.log(res);
94           }
95         });
96       }
97     });
98   },
99
100  // 关闭模态框
101  leaveBubbleModal: function() {
```

```
102     this.triggerEvent('modalLeave', {
103         modalShow: this.data.modalShow
104     })
105     },
106     }
107 );
108
```

- **modal.json**

```
1 {
2     "component": true
3 }
```

- **modal.wxml**

```
1 <!-- 如果modalShow为true，则显示模态框 -->
2 <view wx:if="{{modalShow}}" style="height:100%;width:100%">
3     <view class="modal-wrapper">
4         <!-- 模态框的三角形部分 -->
5         <view class="modal-triangle"></view>
6         <!-- 模态框的主体部分，包含一些可点击的选项 -->
7         <view class="menu-modal">
8             <!-- 遍历modalItems数组，为每个选项创建一个视图元素 -->
9             <view wx:for="{{modalItems}}" wx:key="type" class="menu-modal-item"
10                data-type="{{item.type}}" bindtap="itemTap">{{item.text}}</view>
11         </view>
12     </view>
13     <!-- 如果modalShow为true，则显示一个透明的遮罩层，点击遮罩层可以关闭模态框 -->
14     <view wx:if="{{modalShow}}" class="modal-hidden"
bindtouchstart="leaveBubbleModal"></view>
```

- **modal.wxss**

```
1 /* 模态框容器样式 */
2 .modal-wrapper {
3     position: relative;
4     color: #FFFFFF;
5     height: 70rpx;
6     width: 80%;
7     margin: 0 auto;
8     z-index: 70;
9     opacity: 0.9;
10 }
11
12 /* 模态框中三角形的样式 */
13 .modal-triangle {
14     position: relative;
15     margin: 0 auto;
16     top: 28px;
17     height: 0;
18     width: 0;
19     border: 5px solid #000000;
```

```
20     transform: rotate(45deg);
21 }
22
23 /* 模态框隐藏状态的样式 */
24 .modal-hidden {
25     position: fixed;
26     top: 0;
27     left: 0;
28     width: 100%;
29     height: 100%;
30     background-color: #FFFFFF;
31     opacity: 0;
32     z-index: 69;
33 }
34
35 /* 模态菜单的样式 */
36 .menu-modal {
37     height: 70rpx;
38     font-size: 14px;
39     position: absolute;
40     top: 0;
41     width: 100%;
42     display: flex;
43     display: -webkit-flex;
44     -webkit-align-items: center;
45     align-items: center;
46     box-sizing: border-box;
47 }
48
49 /* 模态菜单项的样式 */
50 .menu-modal-item {
51     color: #FFFFFF;
52     position: relative;
53     width: 35%;
54     height: 100%;
55     display: flex;
56     display: -webkit-flex;
57     align-items: center;
58     -webkit-align-items: center;
59     justify-content: center;
60     -webkit-justify-content: center;
61     background-clip: content-box;
62     background-color: #000000;
63 }
64
65 /* 第一个模态菜单项的样式，添加左上和左下的圆角 */
66 .menu-modal-item:first-child {
67     border-top-left-radius: 8px;
68     border-bottom-left-radius: 8px;
69 }
70
71 /* 最后一个模态菜单项的样式，添加右上和右下的圆角 */
72 .menu-modal-item:last-child {
73     border-top-right-radius: 8px;
74     border-bottom-right-radius: 8px;
75 }
```

```

76
77 /* 模态菜单项之间的分隔线样式 */
78 .menu-modal-item + .menu-modal-item {
79   border-left: 1rpx solid #FFFFFF;
80 }
81
82 /* 模态菜单项被按下时的样式 */
83 .menu-modal-item:active {
84   background-color: #9e9e9e;
85 }
86

```

4.1.2.3 components/play-icon

播放加载组件

以下代码主要功能是：

1. 监听播放状态的变化，当播放状态从'loading'变为'playing'时，根据加载动画的播放次数和剩余的播放时间，决定是立即将播放状态设置为'playing'，还是等待剩余的播放时间后再将播放状态设置为'playing'。
2. 当播放状态变为'loading'时，记录加载开始的时间。

- **play-icon.js**

```

1 // 加载图标的路径
2 const loadingIcon = '../../imgs/loading.gif'
3
4 Component({
5   properties: {
6     // 播放状态，可能的值有'wait'、'loading'和'playing'
7     playType: {
8       type: String,
9       value: 'wait',
10      // 当playType的值发生变化时，会触发这个函数
11      observer: function(newVal, oldVal){
12        // 当播放状态从'loading'变为'playing'时
13        if(oldVal == 'loading' && newVal == 'playing') {
14          // 加载动画的周期为1240ms
15          let loadingTransitionTime = 1240;
16          // 获取当前时间
17          let nowTime = + new Date()
18          // 获取加载开始的时间
19          let loadingStartTime = this.data.loadingStartTime
20          // 计算加载的时间
21          let loadingTime = nowTime - loadingStartTime
22          // 计算加载动画播放的完整次数
23          let loadingCount = parseInt(loadingTime / loadingTransitionTime);
24          // 计算加载动画剩余的播放时间
25          let timeLeft = loadingTransitionTime - loadingTime %
26            loadingTransitionTime;
27
28          // 如果加载动画播放了至少一次，并且剩余的播放时间大于1秒
29          if(loadingCount > 0 && timeLeft > 1000) {
30            // 直接将播放状态设置为'playing'，并清空加载图标
31          }
32        }
33      }
34    }
35  }
36)

```

```

30         this.setData({
31             realPlayType: newVal,
32             loadingImg: '',
33         })
34     } else {
35         // 否则，等待剩余的播放时间后，再将播放状态设置为'playing'
36         setTimeout( ()=>{
37             this.setData({
38                 realPlayType: newVal,
39             })
40             , timeLeft)
41         }
42     } else if (newVal == 'loading'){
43         // 当播放状态变为'loading'时，记录加载开始的时间，并将播放状态设置
44         // 为'loading'
45         this.setData({
46             loadingStartTime: + new Date(),
47             realPlayType: newVal,
48         })
49     } else {
50         // 对于其他的播放状态，直接更新播放状态
51         this.setData({
52             realPlayType: newVal,
53         })
54     },
55 },
56 },
57
58 data: {
59     // 实际在wxml中使用的播放状态
60     realPlayType: 'wait',
61     // 加载开始的时间
62     loadingStartTime: 0,
63 },
64
65 ready: function () {
66     // 组件准备就绪时执行的函数
67 },
68
69 detached: function() {
70     // 组件被移除时执行的函数
71 },
72
73 methods: {
74     // 组件的方法列表
75 }
76 });

```

- play-icon.json

```
1 {
2     "component": true,
3     "usingComponents": {
4     }
5 }
```

- play-icon.wxml

```
1 <!-- 主视图，包含音乐播放图标 -->
2 <view class="play-loud-icon">
3     <!-- 根据播放状态显示或隐藏的主播放图标 -->
4     <image src="../../imgs/play_loud.png" class="play-loud-img play-icon-main"
{{realPlayType == 'loading' ? 'is-hide' : ''}}></image>
5
6     <!-- 当播放状态不是'loading'时，以下内容生效 -->
7     <block wx:if="{{realPlayType != 'loading'}}">
8
9         <!-- 当播放状态是'playing'时，显示动画效果 -->
10        <block wx:if="{{realPlayType=='playing'}}">
11            <image src="../../imgs/play_loud_1.png" class="play-loud-img play-
animation" ></image>
12            <image src="../../imgs/play_loud_2.png" class="play-loud-img play-
animation1"></image>
13        </block>
14
15         <!-- 当播放状态不是'playing'时，显示静态图标 -->
16         <block wx:else>
17             <image src="../../imgs/play_loud_1.png" class="play-loud-img">
18         </block>
19     </block>
20
21         <!-- 当播放状态是'loading'时，显示过渡效果 -->
22         <block wx:else="{{realPlayType != 'loading'}}">
23             <view class="play-transition"></view>
24         </block>
25     </view>
```

- play-icon.wxss

```
1 /* 容器的样式 */
2 .play-loud-icon {
3     position: relative;
4     height: 40rpx;
5     width: 40rpx;
6 }
7
8 /* 音乐播放图标的样式 */
9 .play-loud-img {
10    position: absolute;
11    left: 0;
12    bottom: 0;
13    height: 40rpx;
14    width: 40rpx;
```

```
15 }
16
17 /* 主播放图标的过渡效果 */
18 .play-icon-main {
19   transition: all .2s ease-out;
20 }
21
22 /* 加载状态的图标样式 */
23 .play-loading-img {
24   position: absolute;
25   height: 40rpx;
26   width: 40rpx;
27   left: -1rpx;
28   top: 0rpx;
29 }
30
31 /* 隐藏元素的样式 */
32 .is-hide {
33   opacity: 0;
34 }
35
36 /* 过渡效果的样式，采用背景图base64实现 */
37 .play-transition {
38   position: absolute;
39   height: 40rpx;
40   width: 40rpx;
41   background: transparent url(...) no-repeat;
42   background-size: 40rpx 40rpx;
43   left: -1rpx;
44   top: 0rpx;
45 }
46
47 /* 音乐播放动画的共享样式 */
48 .play-animation,
49 .play-animation1 {
50   -webkit-animation-delay: 200ms;
51   animation-delay: 200ms;
52   -webkit-animation: tranOpacity 1200ms ease-in-out infinite;
53   animation: tranOpacity 1200ms ease-in-out infinite;
54 }
55
56 /* 音乐播放动画1的样式 */
57 .play-animation {
58   -webkit-animation-name: tranOpacity;
59   animation-name: tranOpacity;
60 }
61
62 /* 音乐播放动画2的样式 */
63 .play-animation1 {
64   -webkit-animation-name: tranOpacity1;
65   animation-name: tranOpacity1;
66 }
67
68 /* 动画1的关键帧定义 */
69 @-webkit-keyframes tranOpacity {
70   0% {
```

```

71     opacity: 0;
72   }
73   35% {
74     opacity: 1;
75   }
76   100% {
77     opacity: 1;
78   }
79 }
80
81 @keyframes tranOpacity {
82   0% {
83     opacity: 0;
84   }
85   35% {
86     opacity: 1;
87   }
88   100% {
89     opacity: 1;
90   }
91 }
92
93 /* 动画2的关键帧定义 */
94 @-webkit-keyframes tranOpacity1 {
95   0% {
96     opacity: 0;
97   }
98   35% {
99     opacity: 0;
100  }
101  100% {
102    opacity: 1;
103  }
104 }
105
106 @keyframes tranOpacity1 {
107   0% {
108     opacity: 0;
109   }
110   35% {
111     opacity: 0;
112   }
113   100% {
114     opacity: 1;
115   }
116 }
117

```

4.1.2.4 components/result-bubble

翻译框组件

以下代码主要功能是：

1. 接收一个对象item作为输入，该对象包含文本信息以及音频路径等数据。
2. 根据item对象中的数据，触发文字的翻译，并且播放翻译后的音频。

3. 提供了一些界面操作，例如弹出和关闭模态框，以及触发播放和停止音频的操作。

- **result-bubble.js**

```
1 // 引入语言配置
2 import { language } from '../../utils/conf.js'
3
4 // 定义一个组件
5 Component({
6     // 定义组件的属性
7     properties: {
8         // 属性:item, 类型:Object, 观察函数进行数据监听
9         item: {
10             type: Object,
11             value: {},
12             observer: function(newVal, oldVal) {
13                 // 当记录状态为2(翻译完成), 且文本有变化, 触发重新翻译事件
14                 if(this.data.recordStatus == 2 && oldVal.text && oldVal.text != ''
15                     && newVal.text != oldVal.text) {
16                     this.triggerEvent('translate', {
17                         item: this.data.item,
18                         index: this.data.index,
19                     })
20                 }
21                 // 翻译内容改变触发音频播放, 或者结束播放动画
22                 if(newVal.autoPlay && newVal.translateVoicePath != oldVal.translateVoicePath){
23                     this.autoPlayTranslatevoice()
24                 } else if(newVal.translateVoicePath == "") {
25                     this.playAnimationEnd()
26                 }
27             },
28             // 编辑界面展示标志
29             editShow: {
30                 type: Boolean,
31                 value: false,
32             },
33             // 项目索引
34             index: {
35                 type: Number,
36             },
37             // 当前翻译的音频路径
38             currentTranslatevoice: {
39                 type: String,
40                 observer: function(newVal, oldVal){
41                     if(newVal != '' && newVal != this.data.item.translateVoicePath) {
42                         this.playAnimationEnd()
43                     }
44                 },
45             },
46             // 记录状态: 0-正在识别, 1-正在翻译, 2-翻译完成
47             recordStatus: {
48                 type: Number,
49                 value: 2,
```

```
50     },
51   },
52
53   // 定义组件的内部数据
54   data: {
55     // 语言类型
56     tips_language: language[0],
57
58     // 是否显示模态框
59     modalShow: false,
60
61     // 语音播放状态
62     playType: 'wait',
63
64     // 待定义动画
65     waiting_animation: {},
66     waiting_animation_1: {},
67
68     // 编辑图标路径
69     edit_icon_path: '../../imgs/edit.png'
70   },
71
72   // 组件生命周期函数-在组件布局完成后执行
73   ready: function () {
74     if(this.data.item.autoPlay) {
75       this.autoPlayTranslateVoice()
76     }
77   },
78
79   // 组件生命周期函数-在组件实例被从页面节点树移除时执行
80   detached: function() {
81     // console.log("detach")
82   },
83
84   // 定义方法
85   methods: {
86     // 显示模态框
87     showModal: function() {
88       this.setData({modalShow: true})
89     },
90
91     // 离开模态框
92     modalLeave: function() {
93       this.setData({modalShow: false})
94     },
95
96     // 点击播放图标，根据播放状态和音频过期时间，来决定播放，停止还是触发过期事件
97     playTranslateVoice: function() {
98       let nowTime = parseInt(+ new Date() / 1000)
99       let voiceExpiredTime = this.data.item.translateVoiceExpiredTime || 0
100
101       if(this.data.playType == 'playing') {
102         wx.stopBackgroundAudio()
103         this.playAnimationEnd()
104       } else if(nowTime < voiceExpiredTime) {
105         this.autoPlayTranslateVoice()
```

```
106      } else {
107        this.setData({
108          playType: 'loading',
109        })
110        this.triggerEvent('expired', {
111          item: this.data.item,
112          index: this.data.index,
113        })
114      }
115    },
116
117    // 自动播放翻译后的音频，音频播放结束后，会结束播放动画
118    autoPlayTranslateVoice: function (path, index) {
119      let play_path = this.data.item.translateVoicePath
120
121      if(!play_path) {
122        console.warn("no translate voice path")
123        return
124      }
125
126      wx.onBackgroundAudioStop(res => {
127        console.log("play voice end", res)
128        this.playAnimationEnd()
129      })
130
131      this.playAnimationStart()
132
133      wx.playBackgroundAudio({
134        dataurl: play_path,
135        title: '',
136        success: (res) => {
137          this.playAnimationStart()
138        },
139        fail: (res) => {
140          console.log("failed played", play_path);
141          this.playAnimationEnd()
142        },
143        complete: function (res) {
144          console.log("complete played");
145        }
146      })
147    },
148
149    // 开始播放动画
150    playAnimationStart: function() {
151      this.setData({
152        playType: 'playing',
153      })
154    },
155
156    // 结束播放动画
157    playAnimationEnd: function() {
158      this.setData({
159        playType: 'wait',
160      })
161    },

```

```
162     }
163   });
164 }
```

- **result-bubble.json**

```
1 {
2   "component": true,
3   "usingComponents": {
4     "modal": "/components/modal/index",
5     "waiting-icon": "/components/waiting-icon/index",
6     "play-icon": "/components/play-icon/index"
7   }
8 }
```

- **result-bubble.wxml**

```
1 <!-- 消息气泡容器，长按显示模态框 -->
2 <view class="bubble-wrap" bindlongpress="showModal" >
3   <!-- 模态框容器，当状态为2（翻译完成）时显示 -->
4   <view class="modal-wrap" wx:if="{{recordstatus == 2}}">
5     <!-- 模态框组件 -->
6     <modal
7       modal-show="{{modalShow}}"      <!-- 控制模态框显示隐藏 -->
8       index="{{index}}"           <!-- 项目索引 -->
9       item="{{item}}"            <!-- 当前项目 -->
10      bindmodalleave="modalLeave"> <!-- 模态框离开事件处理函数 -->
11    </modal>
12  </view>
13
14  <!-- 创建时间显示 -->
15  <view class="create-time">{{item.create}}</view>
16
17  <!-- 消息内容区域 -->
18  <view class="section-body" data-index="{{index}}>
19    <!-- 发送消息区域 -->
20    <view class="send-message">
21      <!-- 消息文本内容 -->
22      <view data-id="{{item.id}}" class="text-content" data-index="{{index}}>
23        <!-- 消息详情 -->
24        <view class="text-detail text-detail-{{item.lfrom}}>
25          <!-- 消息文本 -->
26          {{item.text}}
27          <!-- 若正在识别（状态为0），则显示等待图标 -->
28          <waiting-icon wx:if="{{recordstatus == 0}}"/></waiting-icon>
29        </view>
30      </view>
31
32      <!-- 编辑图标，点击进入编辑页面 -->
33      <navigator
34        hover-class="navigator-hover"
35        data-text="{{item.text}}"
36        data-id="{{item.id}}>
```

```

37     data-index="{{index}}"
38     class="edit-icon"
39     wx:if="{{editShow}}"
40     data-item="{{item}}"
41     url="{{'/pages/edit/edit?content='+item.text+'&index='+index}}">
42         <!-- 编辑图标图片 -->
43         <image class="edit-icon-img" src="{{editIconPath}}></image>
44     </navigator>
45 </view>
46
47 <!-- 若正在翻译（状态大于0），显示分割线 -->
48 <view class="line-between" wx:if="{{recordStatus > 0}}></view>
49
50 <!-- 翻译后的消息区域 -->
51 <view class="translate-message" >
52     <!-- 消息文本内容 -->
53     <view class="text-content">
54         <!-- 消息详情 -->
55         <view class="text-detail text-detail-{{item.lto}}">
56             <!-- 翻译后的文本 -->
57             {{item.translateText}}
58             <!-- 若正在翻译（状态为1），则显示等待图标 -->
59             <waiting-icon wx:if="{{recordStatus == 1}}></waiting-icon>
60         </view>
61     </view>
62
63     <!-- 若翻译完成（状态为2），显示播放图标，点击播放翻译音频 -->
64     <view class="play-icon" catchtap="playTranslateVoice"
65     catchtouchstart="playTranslateVoice" wx:if="{{recordStatus == 2

```

- result-bubble.wxss

```

1  /* 消息气泡容器 */
2 .bubble-wrap {
3     position: relative;
4 }
5
6 /* 等待点样式 */
7 .wait-point {
8     display:inline-block;
9     width:6px;
10    height:6px;
11    border-radius:3px;
12    background-color: #ddd;
13    margin: 0 2px;
14 }
15
16 /* 加载状态样式 */
17 .loading {
18     position: relative;
19 }
20
21 /* 分割线样式 */
22 .line-between {

```

```
23 height: 1px;
24 width: 100%;
25 background: #F1F1F1;
26 overflow: hidden;
27 margin: 30rpx 0;
28 }
29
30 /* 创建时间文本样式 */
31 .create-time {
32   font-size: 28rpx;
33   color: #B2B2B2;
34   margin-bottom: 10px;
35   display: flex;
36   justify-content: center;
37 }
38
39 /* 消息内容区域样式 */
40 .section-body {
41   word-wrap: break-word;
42   border-radius: 10px;
43   position: relative;
44   width: 100%;
45   background: #FFFFFF;
46   box-shadow: 0 2px 16px 2px rgba(0, 0, 0, 0.03);
47   padding: 50rpx 60rpx;
48   box-sizing: border-box;
49   min-height: 260rpx;
50 }
51
52 /* 消息详情文本样式 */
53 .text-detail {
54   font-size: 36rpx;
55   line-height: 1.231;
56   vertical-align: text-bottom;
57   box-sizing: border-box;
58   font-family: "PingFang-SC-Regular", "SimSun", "Microsoft Yahei";
59 }
60
61 /* 英文和中文消息详情样式 */
62 .text-detail-en_US {
63   line-height: 1.231;
64 }
65 .text-detail-zh_CN {
66   line-height: 1.41;
67 }
68
69 /* 发送和翻译消息样式 */
70 .translate-message,
71 .send-message {
72   position: relative;
73   padding: 0 2px;
74 }
75 .send-message .text-detail {
76   color: #9B9B9B;
77 }
```

```
79 /* 编辑图标样式 */
80 .edit-icon {
81   position: absolute;
82   display: flex;
83   align-items: center;
84   right: 8rpx;
85   bottom: 7rpx;
86   padding: 0 8rpx;
87 }
88 .edit-icon-img {
89   width:40rpx;
90   height:40rpx;
91 }
92
93 /* 播放图标样式 */
94 .play-icon {
95   position: absolute;
96   right: 3rpx;
97   bottom: 7rpx;
98   padding: 0 8rpx;
99   display: flex;
100  align-items: center;
101 }
102
103 /* 编辑和播放图标点击范围扩大 */
104 .edit-icon::before,
105 .play-icon::before {
106   content:"";
107   position:absolute;
108   top:-10rpx;
109   left:-10rpx;
110   bottom:-10rpx;
111   right:-10rpx;
112 }
113
114 /* 消息文本内容样式 */
115 .text-content {
116   margin: 0 48px 0 0;
117   box-sizing: border-box;
118 }
119
120 /* 模态框容器样式 */
121 .modal-wrap {
122   position: absolute;
123   width: 100%;
124   box-sizing:border-box;
125 }
126
127 /* 重置navigator样式 */
128 .navigator-hover {
129   background-color: #fff;
130 }
```

4.1.2.5 components/waiting-icon

等待加载组件

以下代码主要功能是：

1. 控制等待动画的开始，停止
2. 设置等待动画的间隔

- **waiting-icon.js**

```
1 // 定义小程序组件
2 Component({
3   properties: {
4     // 这里定义了组件的属性
5   },
6
7   // 组件的初始数据
8   data: {
9     waiting_animation: {},      // 定义等待动画对象
10    waiting_animation_1: {},    // 定义另一个等待动画对象
11  },
12
13 // 组件的生命周期函数，在组件布局完成后执行
14 ready: function () {
15   console.log("ready waitting")
16
17   // 创建两个等待动画，一个持续600毫秒，一个持续400毫秒
18   this.waiting_animation = wx.createAnimation({
19     duration: 600
20   })
21   this.waiting_animation_1 = wx.createAnimation({
22     duration: 400
23   })
24
25   // 设置动画循环
26   this.setWaitInterval()
27 },
28
29 // 组件生命周期函数，在组件实例被从页面节点树移除时执行
30 detached: function() {
31   // 当组件被移除时，清除动画
32   this.clearAnimation()
33 },
34
35 methods: {
36   // 清除动画的函数
37   clearAnimation: function() {
38     this.endWaitAnimation()
39   },
40
41   // 结束动画的函数，将清除循环，并重置动画对象
42   endWaitAnimation: function() {
43     clearInterval(this.data.waiting_interval)
44     this.setData({ waiting_animation : {} })
45     this.setData({ waiting_animation_1: {} })
```

```

46     },
47
48     // 开始动画的函数，设置动画的参数并启动动画
49     startWaitAnimation: function () {
50         this.waiting_animation.opacity(0).scale(1.2, 1.2).step()
51         this.waiting_animation.opacity(1).scale(1, 1).step()
52         this.setData({ waiting_animation: this.waiting_animation.export() })
53
54         this.waiting_animation_1.opacity(0).scale(1.2, 1.2).step()
55         this.waiting_animation_1.opacity(1).scale(1, 1).step()
56         this.setData({ waiting_animation_1: this.waiting_animation_1.export() })
57     },
58
59     // 设置动画循环的函数，将清除并重新启动动画循环
60     setWaitInterval: function() {
61         this.endWaitAnimation()
62
63         // 创建一个新的循环，每600毫秒启动一次动画
64         this.data.waiting_interval = setInterval( ()=>{
65             this.startWaitAnimation()
66             },600 )
67         },
68     }
69 });
70

```

- **waiting-icon.json**

```

1  {
2      "component": true,
3      "usingComponents": {
4          }
5  }
6

```

- **waiting-icon.wxml**

```

1 <view class="loading">
2     <view class="loading-icon">.</view>
3     <view animation="{{waiting_animation}}" class="loading-icon">.</view>
4     <view animation="{{waiting_animation_1}}" class="loading-icon">.</view>
5 </view>

```

- **waiting-icon.wxss**

```
1  .loading {
2    position: relative;
3    display: inline;
4  }
5
6  .loading-icon {
7    display: inline;
8  }
9
10
```

4.1.3 imgs/

包含微信小程序内部需要用到的静态图片。

4.1.4 pages/

4.1.4.1 pages/choose_language

切换翻译语言的页面，该页面包含两个语言列表，分别代表源语言和目标语言，用户可以通过点击列表中的项来选择语言。选择的语言信息会被保存到本地存储和全局变量中，并在页面显示时更新。

1 choose_language.js

```
1 // 获取全局应用实例
2 const app = getApp()
3
4 // 导入工具函数
5 const util = require('../utils/util.js')
6
7 // 定义页面
8 Page({
9   // 页面初始数据
10  data: {
11    // 语言列表，每一种语言包含一个图像、语言名称、音频等信息
12    list_apbtoDn5: [
13      {
14        "image": "...", "text": "中文", "chs": '中文', 'lang': 'zh',
15        'sound': 'zh_CN'
16      },
17      {
18        "image": "...", "text": "英语", "chs": '英文', 'lang': 'en',
19        'sound': 'en_US'
20      },
21      {
22        "image": "...", "text": "德语", "chs": '德语', 'lang': 'de'
23      },
24      {
25        "image": "...", "text": "韩语", "chs": '韩语', 'lang': 'kor'
26      }
27    ],
28    selectedIndex1: null, // 记录第一次选择的语言的索引
29    selectedIndex2: null, // 记录第二次选择的语言的索引
30    isClicked1: false, // 是否已经完成第一次点击
31    isClicked2: false, // 是否已经完成第二次点击
32  }
33})
```

```

30     langList: app.globalData.langList, // 全局语言列表
31     curLang: {}, // 当前语言
32     fromLang: {},
33     targetLang: {}, // 目标语言
34     currentsound: '', // 当前语音
35   },
36
37 // 页面分享功能
38 onShareAppMessage() {
39   return {};
40 },
41
42 // 处理第一次点击
43 handleTap1: function() {
44   this.setData({
45     isClicked1: false,
46     isClicked2: false,
47   });
48 },
49
50 // 处理第二次点击
51 handleTap2: function() {
52   this.setData({
53     isClicked2: true,
54     isClicked1: true,
55   });
56 },
57
58 // 处理点击事件，获取点击的语言信息，并保存到对应的数据中
59 handleTap: function(e) {
60   let langObj = e.currentTarget.dataset
61   var index = langObj.index;
62   var sound = langObj.sound;
63   console.log("Clicked item index and sound : ", index, sound);
64   if (!this.data.isClicked1) {
65     this.setData({
66       selectedIndex1: index,
67       fromLang: langObj
68     });
69     app.globalData.fromLang = langObj
70   }else{
71     this.setData({
72       selectedIndex2: index,
73       targetLang: langObj,
74       currentsound: sound
75     });
76     // 把当前的语言和音频信息保存到本地存储中
77     wx.setStorageSync('currentsound', this.data.currentsound);
78     wx.setStorageSync('language', langObj)
79
80     // 把当前的语言保存到全局变量中
81     this.setData({ 'curLang': langObj })
82     app.globalData.curLang = langObj
83   }
84   console.log("Selected item: ", this.data.curLang, this.data.targetLang);
85 },

```

```
86 // 当页面显示时，更新当前语言信息
87 onShow: function () {
88     this.setData({ curLang: app.globalData.curLang })
89 },
90 });
91 
```

2 choose_language.json

```
1 {
2     "usingComponents": {}
3 }
```

3 choose_language.wxml

该段代码的主要功能是显示一个语言列表，用户可以通过点击来选择“翻译语言”和“目标语言”。选择的语言会以不同的样式显示，以区分当前

```
1 <!-- 页面主体是一个垂直方向的 flex 布局 -->
2 <view class="flex-col page space-y-44">
3     <!-- 头部区域是一个水平的 flex 布局 -->
4     <view class="flex-row items-center group space-x-8">
5         <!-- 点击图片，会导航到首页 -->
6         <navigator url="../../pages/index/index" open-type="redirect"
7             style="display: inline-block">
8             <image class="image" src="..." />
9         </navigator>
10        <!-- 头部文字提示 -->
11        <text class="text">选择语言</text>
12    </view>
13
14    <!-- 内容区域也是一个垂直方向的 flex 布局 -->
15    <view class="flex-col group_2 space-y-30">
16        <!-- 包含两个选项：翻译语言和目标语言 -->
17        <view class="flex-row space-x-15">
18            <view class="flex-row {{isClicked1 ? 'text-wrapper_2' : 'text-
19             wrapper'}}" bindtap="handleTap1">
20                <text class="font_1 {{isClicked1 ? 'text_2' : 'text_3'}}>翻译语言
21            </text>
22        </view>
23        <view class="flex-row {{isClicked2 ? 'text-wrapper' : 'text-
24             wrapper_2'}}" bindtap="handleTap2">
25            <text class="font_1 {{isClicked1 ? 'text_3' : 'text_2'}}>目标语言
26        </text>
27    </view>
28
29    <!-- 语言列表区域 -->
30    <view class="flex-col space-y-18">
31        <text class="self-start font_2 text_4">全部语言</text>
32        <!-- 遍历语言列表，生成每一项 -->
33        <view class="flex-col space-y-16">
34            <view
```

```

31         class="flex-row items-center {{selectedIndex1 == i && isClicked1
32 == false || selectedIndex2 == i && isClicked1 == true ? 'list-item' : 'list-
33 item_2'}} space-x-24"
34         wx:for="{{list_aPbtoDn5}}"
35         wx:key="index"
36         wx:for-item="item"
37         wx:for-index="i"
38         data-chs="{{item.chs}}"
39         data-lang="{{item.lang}}"
40         data-index="{{i}}"
41         data-src="{{item.image}}"
42         data-sound="{{item.sound ? item.sound : 'en_US'}}"
43         bindtap="handleTap"
44     >
45         <!-- 语言图标 -->
46         <image class="image_2" src="{{item.image}}"/>
47         <!-- 语言名称 -->
48         <text class="{{selectedIndex1 == i && isClicked1 == false ||
49 selectedIndex2 == i && isClicked1 == true ? 'text_5' : 'text_6'}}">
50             {{item.text}}
51         </text>
52     </view>

```

4 choose_language.wxss

```

1  /* 页面样式，包含内边距，背景色，宽度，溢出处理等样式 */
2  .page {
3      padding: 72rpx 0 264rpx;
4      background-color: #f8f8f9;
5      border-radius: 40rpx;
6      width: 100%;
7      overflow-y: auto;
8      overflow-x: hidden;
9      height: 100%;
10 }
11
12 /* 处理间距类 .space-y-44 的子元素之间的垂直间距 */
13 .space-y-44 > view:not(:first-child),
14 .space-y-44 > text:not(:first-child),
15 .space-y-44 > image:not(:first-child) {
16     margin-top: 88rpx;
17 }
18
19 /* 处理类 .group 的内边距 */
20 .group {
21     padding: 0 16rpx;
22 }
23
24 /* 处理间距类 .space-x-8 的子元素之间的水平间距 */
25 .space-x-8 > view:not(:first-child),
26 .space-x-8 > text:not(:first-child),

```

```
27 .space-x-8 > image:not(:first-child) {  
28   margin-left: 16rpx;  
29 }  
30  
31 /* 图片样式，包括宽高设置 */  
32 .image {  
33   width: 48rpx;  
34   height: 48rpx;  
35 }  
36  
37 /* 文本样式，包含字体，颜色，大小等 */  
38 .text {  
39   color: #1e3163;  
40   font-size: 36rpx;  
41   font-family: Poppins;  
42   font-weight: 700;  
43   line-height: 34rpx;  
44 }  
45  
46 /* 处理类 .group_2 的内边距 */  
47 .group_2 {  
48   padding: 0 48rpx;  
49 }  
50  
51 /* 处理间距类 .space-y-30 的子元素之间的垂直间距 */  
52 .space-y-30 > view:not(:first-child),  
53 .space-y-30 > text:not(:first-child),  
54 .space-y-30 > image:not(:first-child) {  
55   margin-top: 60rpx;  
56 }  
57  
58 /* 处理间距类 .space-x-15 的子元素之间的水平间距 */  
59 .space-x-15 > view:not(:first-child),  
60 .space-x-15 > text:not(:first-child),  
61 .space-x-15 > image:not(:first-child) {  
62   margin-left: 80rpx;  
63 }  
64  
65 /* 定义文本容器样式，包括内边距、背景颜色、圆角半径、阴影、高度、边框和宽度等 */  
66 .text-wrapper {  
67   display: inline;  
68   padding: 28rpx 85rpx 40rpx 85rpx;  
69   background-color: #ffffff;  
70   border-radius: 16rpx;  
71   filter: drop-shadow(0px 0px 1rpx #0000000a, 0px 0px 2rpx #0000000f, 0px  
8rpx 8rpx #0000000a);  
72   height: 50rpx;  
73   border: solid 2rpx #0064e1;  
74   width :40%;  
75 }  
76  
77 /* 定义文本容器2的样式，包括内边距、背景颜色、圆角半径、高度、边框和宽度等 */  
78 .text-wrapper_2 {  
79   display: inline;  
80   padding: 28rpx 85rpx 40rpx 85rpx;  
81   flex: 1 1 310rpx;
```

```
82 border-radius: 16rpx;
83 height: 50rpx;
84 border: solid 2rpx #a7a7a7;
85 width: 40%;
86 }
87
88 /* 定义字体样式，包括字体大小、字体系列和行高 */
89 .font_1 {
90   font-size: 28rpx;
91   font-family: Poppins;
92   line-height: 26rpx;
93 }
94
95 /* 定义文本样式，包括颜色、字体权重、背景图像 */
96 .text_3 {
97   color: transparent;
98   font-weight: 700;
99   background-image: linear-gradient(180deg, #0064e1 0%, #0845c2 100%);
100 -webkit-background-clip: text;
101 }
102
103 /* 定义文本2的样式，颜色为灰色 */
104 .text_2 {
105   color: #a8a8a8;
106 }
107
108 /* 设置间距类 .space-y-18 的子元素之间的垂直间距 */
109 .space-y-18 > view:not(:first-child),
110 .space-y-18 > text:not(:first-child),
111 .space-y-18 > image:not(:first-child) {
112   margin-top: 36rpx;
113 }
114
115 /* 定义字体2的样式，包括字体大小、字体系列、行高和颜色 */
116 .font_2 {
117   font-size: 32rpx;
118   font-family: Poppins;
119   line-height: 29rpx;
120   color: #1e3163;
121 }
122
123 /* 定义文本4的样式，字体权重为700，行高为30rpx */
124 .text_4 {
125   font-weight: 700;
126   line-height: 30rpx;
127 }
128
129 /* 设置间距类 .space-y-16 的子元素之间的垂直间距 */
130 .space-y-16 > view:not(:first-child),
131 .space-y-16 > text:not(:first-child),
132 .space-y-16 > image:not(:first-child) {
133   margin-top: 32rpx;
134 }
135
136 /* 定义图片2的样式，包括宽度和高度 */
137 .image_2 {
```

```

138   width: 64rpx;
139   height: 64rpx;
140 }
141
142 /* 定义文本5的样式，包括颜色、字体大小、字体系列、字体权重和行高 */
143 .text_5 {
144   color: #ffffff;
145   font-size: 35rpx;
146   font-family: Poppins;
147   font-weight: 700;
148   line-height: 30rpx;
149 }
150
151 /* 设置文本6的上边距 */
152 .text_6 {
153   margin-top: 10rpx;
154 }
155
156 /* 设置间距类 .space-x-24 的子元素之间的水平间距 */
157 .space-x-24 > view:not(:first-child),
158 .space-x-24 > text:not(:first-child),
159 .space-x-24 > image:not(:first-child) {
160   margin-left: 48rpx;
161 }
162
163 /* 定义列表项样式，包括内边距、背景颜色、圆角半径和阴影等 */
164 .list-item {
165   padding: 32rpx;
166   background-color: #1392fb;
167   border-radius: 16rpx;
168   box-shadow: 0px 0px 2rpx #0000000a, 0px 0px 4rpx #0000000f, 0px 8rpx
169   16rpx #0000000a;
170 }
171
172 /* 定义列表项2的样式，包括内边距、背景颜色、圆角半径和阴影等 */
173 .list-item_2 {
174   padding: 32rpx;
175   background-color: #ffffff;
176   border-radius: 16rpx;
177   box-shadow: 0px 0px 2rpx #0000000a, 0px 0px 4rpx #0000000f, 0px 8rpx
178   16rpx #0000000a;
179 }

```

4.1.4.2 pages/edit

编辑文本的输入框页面，编辑文本的最大长度限制为200个字符，页面数据中的 `edit_text` 字段存储输入的文本，`remain_length` 字段存储剩余可输入的字符数。

当用户在输入框中输入或删除文本时，页面会更新剩余可输入的字符数并显示给用户。当用户提交输入文本时，如果文本长度超过0且与旧文本不相同，页面会将新文本保存到上一页的对话列表中，并返回上一页；否则，如果用户提交的文本为空，页面会提供相应的处理。

```
1 //初始化底部高度
2 const initBottomHeight = 0
3
4 var app = getApp()
5
6 Page({
7     // 页面的初始数据
8     data: {
9         edit_text_max: 200,          //最大编辑文本长度
10        remain_length: 200,        //剩余可输入长度
11        edit_text: "",            //编辑文本内容
12        is_focus: false,          //是否聚焦
13        tips: "",                //提示信息
14        index: -1,                //索引
15        bottomHeight: initBottomHeight //底部高度
16    },
17
18    // 获取最大编辑文本长度
19    getEditTextMax: function () {
20        return this.data.edit_text_max
21    },
22
23    // 更新剩余可输入长度
24    updateRemainLength: function (now_content) {
25        this.data.remain_length = this.getEditTextMax() - now_content.length
26        this.data.tips = "还可以输入" + this.data.remain_length + "字..."
27        this.setData({ tips: this.data.tips })
28    },
29
30    // 设置编辑文本内容
31    setEditText: function (text) {
32        this.data.edit_text = text
33        this.setData({ edit_text: this.data.edit_text })
34        // 更新剩余长度显示
35        this.updateRemainLength(text)
36        this.setData({ is_focus: true })
37    },
38
39    // bindinput事件处理
40    editInput: function (event) {
41        console.log(event)
42        if (event.detail.value.length > this.getEditTextMax()) {
43            //处理输入内容超过最大长度的情况
44        } else {
45            this.data.edit_text = event.detail.value
46            this.updateRemainLength(this.data.edit_text)
47        }
48    },
49
50    // bindconfirm事件处理
51    editConfirm: function (event) {
52        if (this.data.edit_text.length > 0 && this.data.edit_text !=
```

```
this.data.oldText) {
```

```
53     // 获取页面栈
54     let pages = getCurrentPages();
55     let prevPage = pages[pages.length - 2]; //上一个页面
56     let dialogList = prevPage.data.dialogList.slice(0)
57     let editItem = dialogList[dialogList.length - 1]
58     editItem.text = this.data.edit_text
59
60     prevPage.setData({
61         dialogList: dialogList,
62         recordStatus: 2,
63     })
64     wx.navigateBack()
65 } else {
66     // 处理输入文本为空的情况
67 }
68 },
69
70 // 点击输入框时改变底部按钮的高度，使得提示和按钮始终在键盘上方
71 editFocus: function(e) {
72     let {value, height} = e.detail
73     console.log(value, height)
74
75     if(!isNaN(height)) {
76         this.setData({
77             bottomHeight: height + initBottomHeight
78         })
79     }
80 },
81
82 // 输入框失去焦点事件处理
83 editBlur: function() {
84     this.setData({
85         bottomHeight: initBottomHeight
86     })
87 },
88
89 // 清空内容事件处理
90 deleteContent: function () {
91     this.setEditText("")
92     this.setData({
93         is_focus: true
94     })
95 },
96
97 // 生命周期函数--监听页面加载
98 onLoad: function (options) {
99     this.setEditText(options.content)
100    let index = parseInt(options.index)
101    this.setData({
102        index: index,
103        oldText: options.content,
104    })
105 },
106 })
```

2 edit.json

```
1 | {}
```

3 edit.wxml

```
1 <!-- edit.wxml -->
2 <!-- 定义了一个编辑文本的页面 -->
3 <view class="container edit-container">
4
5   <!-- 编辑区域，使用textarea组件 -->
6   <textarea
7     maxlength="{{edit_text_max}}"          <!-- 设置最大输入长度 -->
8     class="edit_textarea"                  <!-- 指定样式 -->
9     auto-focus="{{true}}"
10    focus="{{is_focus}}"
11    bindinput="editInput"                 <!-- 绑定输入事件 -->
12    bindconfirm="editConfirm"           <!-- 绑定确定事件 -->
13    value="{{edit_text}}"
14    adjust-position="{{true}}"
15    bindfocus="editFocus"                <!-- 显示的文本 -->
16    bindblur="editBlur">
17   </textarea>
18
19   <!-- 底部区域，显示提示和删除按钮 -->
20   <view class="bottom-wrap" style="padding-bottom: {{bottomHeight}}px">
21     <view class="tips-wrapper">
22       <!-- 显示提示信息 -->
23       <text class="edit-tips">{{tips}}</text>
24       <!-- 删除按钮 -->
25       <view class="delete-content" capture-bind:tap="deleteContent">
26         <!-- 使用图片作为删除按钮的图标 -->
27         <image src="../../imgs/delete_all.png" class="img-delete-all">
28       </image>
29     </view>
30   </view>
31 </view>
32
```

3 edit.wxss

```
1 /* pages/edit/edit.wxss */
2
3 /* 容器样式 */
4 .edit-container {
5   position: relative;
6   padding: 20px 50rpx 20rpx;          /* 内边距设置 */
7   justify-content: flex-start;        /* 子元素沿主轴的对齐方式 */
8   -webkit-justify-content: flex-start; /* webKit内核的浏览器兼容设置 */
9   background-color: #FAFAFA;          /* 背景色 */
10 }
11
12 /* 文本输入框样式 */
```

```

13 .edit_textarea {
14   flex: 1;          /* 弹性布局比例设置 */
15   width: 100%;      /* 宽度100% */
16   box-sizing: border-box; /* 设置盒模型 */
17   font-size: 36rpx;    /* 字体大小 */
18   line-height: 60rpx;   /* 行高 */
19 }

20
21 /* 底部提示和删除按钮容器样式 */
22 .tips-wrapper {
23   width: 100%;      /* 宽度100% */
24   display: flex;     /* 使用弹性布局 */
25   display: -webkit-flex; /* webKit内核的浏览器兼容设置 */
26   justify-content: space-between; /* 子元素之间的间距均匀分布 */
27   -webkit-justify-content: space-between; /* webKit内核的浏览器兼容设置 */
28   padding: 0;         /* 内边距设置 */
29   box-sizing: border-box; /* 设置盒模型 */
30   align-items: center;  /* 子元素沿交叉轴的对齐方式 */
31   -webkit-align-items: center; /* webKit内核的浏览器兼容设置 */
32 }

33
34 /* 提示文字样式 */
35 .edit-tips {
36   font-size: 30rpx;    /* 字体大小 */
37   color: #B2B2B2;      /* 字体颜色 */
38   line-height: 50rpx;    /* 行高 */
39 }

40
41 /* 删除按钮图标样式 */
42 .img-delete-all {
43   height: 32rpx;        /* 高度 */
44   width: 28rpx;         /* 宽度 */
45 }

46
47 /* 删除按钮容器样式 */
48 .delete-content {
49   position: relative;   /* 定位方式 */
50   right: -20rpx;        /* 向右偏移 */
51   padding: 20rpx 20rpx;   /* 内边距设置 */
52 }

```

4.1.4.3 pages/getPic

1 getPic.js

这段代码的主要功能是：

1. 拍照：用户可以点击按钮触发 `takeshot` 方法，调用微信的摄像头接口进行拍照，拍照成功后，会将图片转换为base64格式，并保存到全局变量中，然后跳转到OCR页面。
2. 提示：用户可以点击按钮触发 `onTap` 方法，显示一个“只支持中译英”的提示。

```

1 // 引入api模块，该模块可能包含了一些工具函数
2 var api = require("../utils/api.js");
3
4 // 获取全局的app实例

```

```
5 const app = getApp();
6
7 // 定义一个页面
8 Page({
9     // 页面的初始数据
10    data: {
11        },
12    },
13
14    // 定义一个方法，用于拍照
15    takeShot: function () {
16        // 创建一个摄像头的上下文
17        const ctx = wx.createCameraContext()
18
19        // 使用上下文来拍照
20        ctx.takePhoto({
21            // 设置图片质量为高
22            quality: 'high', // 上传图片进行文字提取时，图片转换为base64后，大小不能超过
300k
23
24        // 拍照成功后的回调函数
25        success: (res) => {
26            // 打印图片的临时地址
27            console.log("图片的临时地址为：" + res.tempImagePath);
28
29            // 调用api模块的getPicBase64函数，将图片转换为base64格式
30            api.getPicBase64(res.tempImagePath).then(function (res) {
31                // 将转换后的base64图片保存到全局数据中
32                app.globalData.picBase64 = res.data;
33
34                // 导航到OCR页面
35                wx.navigateTo({
36                    url: '../OCR/OCR',
37                })
38            })
39        }
40    })
41    },
42
43    // 定义一个方法，用于显示提示
44    onTap: function () {
45        // 显示一个只支持中译英的提示
46        wx.showToast({
47            title: '只支持中译英',
48            icon: "error"
49        })
50    }
51 })
```

2 getPic.json

缺省代码。

```
1  {
2      "usingComponents": {}
3 }
```

3 getPic.wxml

这段代码的主要功能是：

1. 头部视图：包含了一个语言切换视图，用户可以点击中文或英文图标和文字来切换语言。
2. 摄像头视图：显示一个摄像头，用户可以通过这个摄像头来拍照。
3. 工具栏视图：包含了一个拍照图标，用户可以点击这个图标来拍照。

```
1  <!-- 头部视图 -->
2  <view class="head">
3      <!-- 语言切换视图，点击时触发onTap事件 -->
4      <view class="language" bindtap="onTap">
5          <!-- 中文图标，点击时触发onTap事件 -->
6          <image bindtap="onTap" class="language_pic" src="../../imgs/Chinese.png"
7      />
8          <!-- 中文文字，点击时触发onTap事件 -->
9          <text bindtap="onTap" class="language_text" decode>&ampnbsp中文</text>
10         <!-- 切换图标，点击时触发onTap事件 -->
11         <image bindtap="onTap" class="switch" src="../../imgs/switch.png">
12     </image>
13         <!-- 英文图标，点击时触发onTap事件 -->
14         <image bindtap="onTap" class="language_pic" src="../../imgs/English.png"
15     />
16         <!-- 英文文字，点击时触发onTap事件 -->
17         <text bindtap="onTap" class="language_text" decode>&ampnbsp英语</text>
18     </view>
19 </view>
20
21 <!-- 摄像头视图 -->
22 <camera style="margin-top:0rpx;width:100%;height:1100rpx"></camera>
23
24 <!-- 工具栏视图 -->
25 <view class="toolbar">
26     <!-- 拍照图标，点击时触发takeShot事件 -->
```

4 getPic.wxss

这段CSS样式表的主要功能是：

1. `.head`：设置头部视图的样式，包括位置、大小和背景颜色。
2. `.btn`：设置按钮的样式，包括背景颜色、大小、内边距、颜色、边框圆角、外边距和文本对齐方式。
3. `.toolbar`：设置工具栏的样式，使用Flex布局，设置宽度、高度、主轴对齐方式和交叉轴对齐方式。
4. `.shot`：设置拍照图标的样式，包括大小、文本对齐方式和上外边距。

5. `.image`: 设置图片的样式，包括大小和垂直对齐方式。
6. `.language`: 设置语言切换视图的样式，使用Flex布局，设置上内边距、宽度、左内边距和垂直对齐方式。
7. `.language_pic`: 设置语言图标的样式，包括上内边距和垂直对齐方式。
8. `.language_text`: 设置语言文字的样式，包括上内边距、字体大小、字体家族、行高、颜色、上下左外边距、字体家族和行高。
9. `.switch`: 设置切换图标的样式，使用Flex布局，设置对齐方式、方向、边框圆角、大小和自动外边距。

这些样式主要用于设置微信小程序中的元素样式，使得元素在页面上的布局和外观符合设计要求。

```
1  /* 头部视图样式 */
2  .head {
3      position: relative;
4      top: 0;
5      left: 0;
6      width: 100%;
7      height: 200rpx;
8      background-color: #1493FC;
9  }
10
11 /* 按钮样式 */
12 .btn {
13     background-color: #1493fc;
14     width: 200rpx;
15     padding: 15rpx 20rpx;
16     color: #fff;
17     border-radius: 20rpx;
18     margin: 40rpx;
19     text-align: center;
20 }
21
22 /* 工具栏样式 */
23 .toolbar {
24     display: flex;
25     width: auto;
26     height: 100rpx;
27     justify-content: center;
28     align-items: center;
29 }
30
31 /* 拍照图标样式 */
32 .shot {
33     width: 110rpx;
34     height: 110rpx;
35     text-align: center;
36     margin-top: 50rpx;
37 }
38
39 /* 图片样式 */
40 image {
41     width: 30rpx;
42     height: 30rpx;
```

```

43     vertical-align: bottom;
44 }
45
46 /* 语言切换视图样式 */
47 .language {
48     display: flex;
49     padding-top: 100rpx;
50     width: 300rpx;
51     padding-left: 80rpx;
52     vertical-align: bottom;
53 }
54
55 /* 语言图标样式 */
56 .language_pic {
57     padding-top: 25rpx;
58     vertical-align: text-bottom;
59 }
60
61 /* 语言文字样式 */
62 .language_text {
63     padding-top: 20rpx;
64     font-size: 30rpx;
65     font-family: Poppins;
66     line-height: 26rpx;
67     color: #ffffff;
68     margin-top: 10rpx;
69     margin-bottom: 10rpx;
70     margin-left: 10rpx;
71     font-family: Poppins;
72     line-height: 22rpx;
73 }
74
75 /* 切换图标样式 */
76 .switch {
77     display: flex;
78     align-items: center;
79     justify-content: flex-start;
80     flex-direction: column;
81     border-radius: 16rpx;
82     width: 72rpx;
83     height: 72rpx;
84     margin: auto;
85 }
86

```

4.1.4.4 pages/history

1 history.js

这段代码的主要功能是：

1. 显示历史记录：当页面显示时，会从本地存储中获取历史记录，并显示到页面上。
2. 点击历史记录项：用户可以点击历史记录项，页面会重新加载首页，并将点击的历史记录项的查询参数传递给首页。

3. 清除历史记录：用户可以点击按钮触发 `onClearHistory` 方法，清除页面数据中的历史记录数组，并清除本地存储中的历史记录。

```
1 // 引入全局的app实例
2 const app = getApp()
3
4 // 定义一个页面
5 Page({
6     // 页面的初始数据
7     data: {
8         history: [] // 历史记录数组
9     },
10
11     // 页面显示时的回调函数
12     onShow: function () {
13         // 从本地存储中获取历史记录，并设置到页面数据中
14         this.setData({
15             history: wx.getStorageSync('history')
16         })
17     },
18
19     // 点击历史记录项时的回调函数
20     onTapItem: function (e) {
21         // 重新加载首页，并传递点击的历史记录项的查询参数
22         wx.reLaunch({
23             url: `/pages/index/index?query=${e.currentTarget.dataset.query}`
24         })
25     },
26
27     // 清除历史记录的回调函数
28     onClearHistory: function () {
29         // 将页面数据中的历史记录数组设置为空
30         this.setData({
31             history: []
32         })
33
34         // 清除本地存储中的历史记录
35         wx.clearStorage('history')
36     },
37 }
38
```

2 history.json

缺省代码。

```
1 {
2     "navigationBarTitleText": ""
3 }
```

3 history.wxml

这段代码的主要功能是：

1. 显示翻译历史：页面上有一个滚动视图，里面包含了一个历史记录列表视图，列表中的每一项都是一个历史记录项，显示了查询的语言、查询的文本、结果的语言和结果的文本。
2. 清除历史记录：用户可以点击“清除历史记录”文本，触发 `onClearHistory` 事件，清除历史记录。
3. 查看历史记录：用户可以点击历史记录项，触发 `onTapItem` 事件，查看历史记录的详细信息。

```
1  <!-- 可滚动视图，设置为纵向滚动 -->
2  <scroll-view scroll-y class="container">
3      <!-- 历史记录列表视图 -->
4      <view class="history-list">
5          <!-- 头部视图 -->
6          <view class="header">
7              <!-- 标题文本 -->
8              <text class="title">翻译历史</text>
9              <!-- 清除历史记录文本，点击时触发onClearHistory事件 -->
10             <text bindtap='onClearHistory' class="iconfont icon-close">清除历史记录
11         </text>
12     </view>
13     <!-- 历史记录项视图，遍历历史记录数组，点击时触发onTapItem事件，传递查询参数和语言索引 -->
14     <view class="item" wx:for="{{history}}" wx:key="index"
15         bindtap='onTapItem' data-query="{{item.query}}" data-langId="
16             {{item.langIndex}}"
17         <!-- 查询视图 -->
18         <view class="query">
19             <!-- 语言视图 -->
20             <view class="language">
21                 <!-- 语言图标 -->
22                 <image src="../../imgs/Chinese.png" />
23                 <!-- 语言文本 -->
24                 中文
25             </view>
26             <!-- 查询文本 -->
27             {{item.query}}
28         </view>
29         <!-- 结果视图 -->
30         <view class="result">
31             <!-- 语言视图 -->
32             <view class="language">
33                 <!-- 语言图标 -->
34                 <image src="../../imgs/English.png" />
35                 <!-- 语言文本 -->
36                 英语
37             </view>
38             <!-- 结果文本 -->
39             {{item.result}}
40         </view>
41     </view>
42 </scroll-view>
```

4 history.wxss

这些样式主要用于设置微信小程序中的元素样式，使得元素在页面上的布局和外观符合设计要求。具体包括：

1. `.history-list`: 设置历史记录列表的样式，使用Flex布局，设置为列方向，设置内边距。
2. `.header`: 设置头部视图的样式，使用Flex布局，设置上外边距。
3. `.title`: 设置标题文本的样式，设置字体大小和颜色。
4. `.icon-close`: 设置清除历史记录图标的样式，设置左外边距为自动，设置字体大小和颜色。
5. `.item`: 设置历史记录项的样式，包括上外边距、内边距、背景颜色、边框圆角和阴影。
6. `.item .query`: 设置查询视图的样式，包括内边距、底部边框、字体大小、字体家族、行高和颜色。
7. `.item .query .language`: 设置查询语言的样式，包括字体大小、字体家族、行高、颜色、上下外边距。
8. `.item .result`: 设置结果视图的样式，包括上外边距、内边距、字体大小、字体家族、行高和颜色。
9. `.item .result .language`: 设置结果语言的样式，包括字体大小、字体家族、行高、颜色和下外边距。
10. `image`: 设置图片的样式，包括宽度、高度和垂直对齐方式。

这些样式主要用于设置微信小程序中的元素样式，使得元素在页面上的布局和外观符合设计要求。

```
1 /* 历史记录列表样式 */
2 .history-list {
3     display: flex;
4     flex-direction: column;
5     padding: 40rpx;
6 }
7
8 /* 头部视图样式 */
9 .header {
10    display: flex;
11    margin-top: 100rpx;
12 }
13
14 /* 标题文本样式 */
15 .title {
16    flex: 1;
17    font-size: 26rpx;
18    color: #8995a1;
19 }
20
21 /* 清除历史记录图标样式 */
22 .icon-close {
23    margin-left: auto;
24    color: #aaa;
25    font-size: 26rpx;
26 }
27
28 /* 历史记录项样式 */
29 .item {
```

```
30 margin-top: 40rpx;
31 padding: 0 32rpx;
32 background-color: #ffffff;
33 border-radius: 16rpx;
34 box-shadow: 0px 0px 2rpx #0000000a, 0px 0px 4rpx #0000000f, 0px 8rpx 16rpx
#0000000a;
35 }
36
37 /* 查询视图样式 */
38 .item .query {
39   padding: 20rpx;
40   border-bottom: solid 2rpx #e7e7e7;
41   font-size: 32rpx;
42   font-family: Poppins;
43   line-height: 50rpx;
44   color: #1e3163;
45 }
46
47 /* 查询语言样式 */
48 .item .query .language {
49   font-size: 28rpx;
50   font-family: Poppins;
51   line-height: 26rpx;
52   color: #a8abb0;
53   margin-top: 10rpx;
54   margin-bottom: 10rpx;
55 }
56
57 /* 结果视图样式 */
58 .item .result {
59   margin-top: 16rpx;
60   padding: 20rpx;
61   font-size: 32rpx;
62   font-family: Poppins;
63   line-height: 50rpx;
64   color: #1e3163;
65 }
66
67 /* 结果语言样式 */
68 .item .result .language {
69   font-size: 28rpx;
70   font-family: Poppins;
71   line-height: 26rpx;
72   color: #909dbd;
73   margin-bottom: 10rpx;
74 }
75
76 /* 图片样式 */
77 image {
78   width: 30rpx;
79   height: 30rpx;
80   vertical-align: bottom;
81 }
```

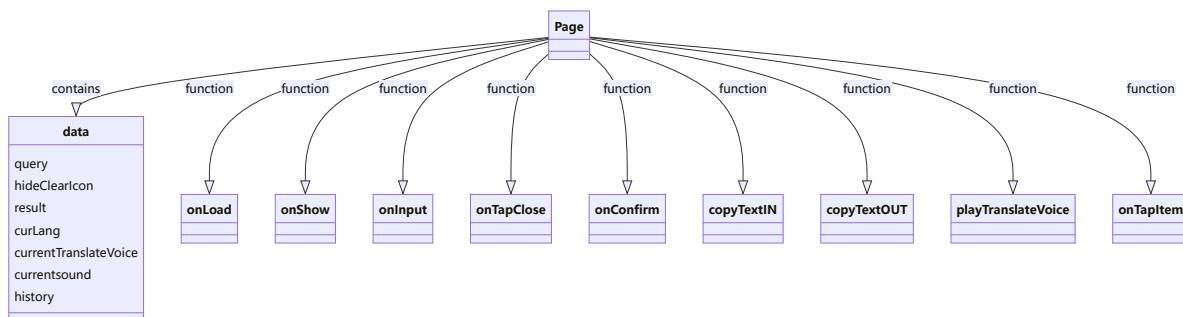
4.1.4.5 pages/index

1 index.js

这段代码的主要功能包括：

1. 用户输入文本进行翻译，翻译结果会保存在历史记录中。
2. 用户可以点击清除图标清除输入的文本和翻译结果。
3. 用户可以复制输入的文本和翻译结果。
4. 用户可以播放翻译的语言。
5. 用户可以查看翻译的历史记录。

以下是代码的流程图：



```
1 // 引入翻译工具和全局应用实例
2 import { translate } from '../../utils/api.js'
3 const app = getApp()
4 const plugin = requirePlugin("WechatsI")
5
6 Page({
7   data: {
8     query: '', // 用户输入的查询文本
9     hideClearIcon: false, // 控制清除图标的显示与隐藏
10    result: [], // 翻译结果
11    curLang: {}, // 当前选择的语言
12    fromLang: {},
13    currentTranslateVoice: '', // 当前播放的语音路径
14    currentsound: '', // 当前语音合成语言
15    history: [] // 翻译历史记录
16  },
17
18  // 页面加载时的处理函数
19  onLoad: function (options) {
20    if (options.query) {
21      this.setData({
22        query: options.query,
23        'hideClearIcon': false // 显示清除图标
24      })
25    }
26  },
27
28  // 页面显示时的处理函数
29  onShow: function () {
30    this.setData({
31      history: wx.getStorageSync('history') // 从本地存储获取历史记录
32    })
33  }
34})
```

```

32     })
33     if (this.data.curLang.lang !== app.globalData.curLang.lang) {
34       this.setData({
35         curLang: app.globalData.curLang
36       })
37     if (this.data.fromLang.lang !== app.globalData.fromLang.lang) {
38       this.setData({
39         fromLang: app.globalData.fromLang
40       })
41       this.onConfirm() // 执行翻译
42     }
43   },
44
45   // 处理用户输入的函数
46   onInput: function (e) {
47     this.setData({
48       'query': e.detail.value,
49       'hideClearIcon': this.data.query.length > 0 ? false : true // 根据输入
      内容是否为空来决定是否显示清除图标
50     })
51   },
52
53   // 处理用户点击清除图标的函数
54   onTapClose: function () {
55     this.setData({
56       query: '',
57       hideClearIcon: true,
58       result: '' // 清除翻译结果
59     })
60   },
61
62   // 执行翻译的函数
63   onConfirm: function () {
64     if (!this.data.query) return // 如果查询文本为空，则不执行翻译
65     translate(this.data.query, {
66       from: this.data.fromLang.lang || 'auto',
67       to: this.data.curLang.lang
68     }).then(res => {
69       this.setData({
70         'result': res.trans_result
71       })
72
73       // 更新历史记录
74       let history = wx.getStorageSync('history') || []
75       history.unshift({
76         query: this.data.query,
77         result: res.trans_result[0].dst,
78         from: res.from,
79         to: res.to
80       })
81       history.length = history.length > 10 ? 10 : history.length
82       wx.setStorageSync('history', history)
83     })
84   },
85
86   // 复制输入文本的函数

```

```

87  copyTextIN: function (e) {
88    wx.setClipboardData({
89      data: this.data.query,
90      success: function (res) {
91        wx.showToast({
92          title: '复制成功',
93        });
94      }
95    });
96  },
97
98 // 复制翻译结果的函数
99 copyTextOUT: function (e) {
100   wx.setClipboardData({
101     data: this.data.result[0].dst,
102     success: function (res) {
103       wx.showToast({
104         title: '复制成功',
105       });
106     }
107   });
108 },
109
110 // 播放翻译语音的函数
111 playTranslateVoice: function (e) {
112   let componentId = e.currentTarget.dataset.id;
113   this.setData({
114     currentsound: wx.getStorageSync('currentsound') || 'en_US'
115   })
116   let lto = this.data.currentsound
117   let content = (componentId === 'src') ? this.data.result[0].src :
118   this.data.result[0].dst
119   plugin.textToSpeech({
120     lang: lto,
121     content: content,
122     success: resTrans => {
123       if (resTrans.retcode == 0) {
124         this.setData({
125           currentTranslateVoice: resTrans.filename,
126         })
127         let play_path = this.data.currentTranslateVoice
128         if (!play_path) {
129           console.warn("no translate voice path")
130           return
131         }
132         let audio = wx.createInnerAudioContext()
133         audio.src = play_path // 设置音频的源
134         audio.play() // 播放音频
135         audio.onError((res) => {
136           console.log(reserrMsg)
137           console.log(res.errCode)
138         })
139       } else {
140         console.warn("语音合成失败", resTrans)
141       }

```

```

142     },
143     fail: function (resTrans) {
144         console.warn("语音合成失败", resTrans)
145     }
146 },
147 },
148
149 // 跳转到历史记录页面的函数
150 onTapItem: function (e) {
151     wx.reLaunch({
152         url: `/pages/history/history`
153     })
154 },
155 }

```

2 index.json

缺省代码。

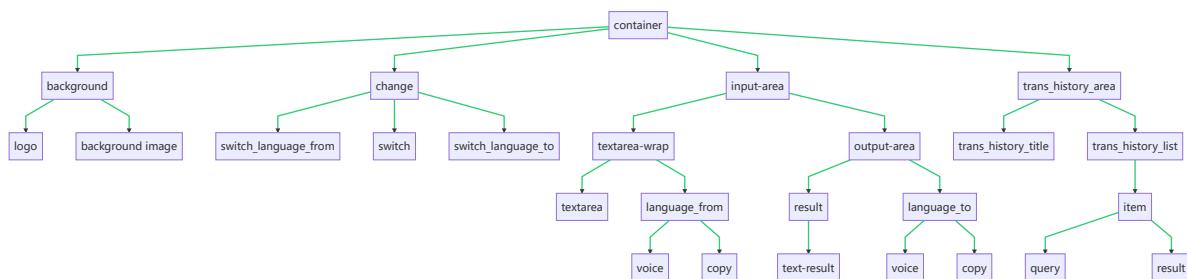
```
1 | {}
```

3 index.wxml

这段代码是微信小程序的WXML模板，用于构建用户界面。主要包括以下部分：

1. **container**: 这是页面的主容器。
2. **background**: 包含logo和背景图片。
3. **change**: 这部分包含语言切换部分，用户可以在这里选择翻译的源语言和目标语言。
4. **input-area**: 这是用户输入要翻译的文本的地方。输入区域包含一个文本框和一个语音按钮，用户可以通过输入或语音输入文本。
5. **output-area**: 这是显示翻译结果的地方。输出区域包含一个文本结果区域和一个语音按钮，用户可以听到翻译的结果。
6. **trans_history_area**: 这部分显示用户的翻译历史。用户可以看到他们过去翻译的文本和结果。

这是该代码的框架图：



```

1 <!--index.wxml-->
2 <view class="container">
3     <!-- 背景图片和logo -->
4     <view class="background">
5         <image class="logo" src="../../imgs/logo.png"></image>
6         <image class="background" src="../../imgs/background.png"></image>
7     </view>
8

```

```

9   <!-- 语言切换部分 -->
10  <view class="change">
11    <navigator url="/pages/choose_language/choose_language" hover-
12      class="navigator-hover">
13      <view class="switch_language_from">
14        <image class="switch_language_pic" src="{{fromLang.src}} />
15        <text class="switch_language_from_text" decode="true">&nbsp;&nbsp;
16          {{from.chs}}</text>
17          <text class="iconfont icon-down"></text>
18        </view>
19        <image class="switch" src="../../imgs/switch.png"></image>
20        <view class="switch_language_to">
21          <image class="switch_language_pic" src="{{curLang.src}} />
22          <text class="switch_language_to_text" decode="true">&nbsp;&nbsp;
23            {{curLang.chs}}</text>
24            <text class="iconfont icon-down"></text>
25          </view>
26        </navigator>
27      </view>
28
29      <!-- 输入区域 -->
30      <view class="input-area">
31        <view class="textarea-wrap">
32          <textarea placeholder='请输入要翻译的文本' placeholder-style=' color:
#1e3163;line-height: 34rpx;font-size: 36rpx;font-family: Poppins;' bindinput='onInput' bindconfirm='onConfirm' bindblur='onConfirm' value=
{{query}}></textarea>
33          <view class="language_from">
34            <image class="language_pic" src="{{fromLang.src}} />
35            <text class="language_from_text">{{from.chs}}</text>
36            <view data-id="src" catchtap="playTranslatevoice"
37              catchtouchstart="playTranslatevoice">
38              <image class="voice" src="../../imgs/voice.png" mode="widthFix" />
39            </view>
40            <view class="copy" bindtap="copyTextIN">
41              <image src="../../imgs/copy.png" mode="widthFix" />
42            </view>
43          </view>
44        </view>
45
46      <!-- 输出区域 -->
47      <view class="output-area">
48        <view class="result">
49          <view class="text-result" wx:for="{{result}}" wx:key="index">
50            <text selectable="true" decode="true">{{item.dst}}</text>
51          </view>
52        </view>
53        <view class="language_to">
54          <image class="language_pic" src="{{curLang.src}} />
55          <text class="language_to_text">{{curLang.chs}}</text>

```

```

56         <image src="../../imgs/copy_white.png" mode="widthFix" />
57     </view>
58 </view>
59 </view>
60
61     <!-- 翻译历史模块 -->
62 <view class="trans_history_area">
63     <view class="trans_history_title">
64         <view class="trans_history_title_chi">翻译历史</view>
65         <view class="trans_history_title_eng">Translation History</view>
66     </view>
67     <view class="trans_history_list">
68         <view class="item" wx:for="{{history}}" wx:key="index"
bindtap=' onTapItem' data-query="{{item.query}}" data-langId="{{item.langIndex}}"
69             <view class="query">
70                 <view class="language">
71                     <image src="../../imgs/Chinese.png" />
72                     中文
73                 </view>
74                 {{item.query}}
75             </view>
76             <view class="result">
77                 <view class="language">
78                     <image src="../../imgs/English.png" />
79                     英语
80                 </view>
81                 {{item.result}}
82             </view>
83         </view>
84     </view>
85 </view>
86 </view>
87 </view>
88

```

4 index.wxss

这段代码是微信小程序的WXSS样式表，用于设置页面的样式。主要包括以下部分：

1. **container**: 设置页面的主容器的位置、大小和背景颜色。
2. **logo和background**: 设置logo和背景图片的位置、大小和显示方式。
3. **change**: 设置语言切换部分的样式，包括颜色、字体大小、内外边距、显示方式、对齐方式等。
4. **input-area和textarea-wrap**: 设置用户输入要翻译的文本的区域的样式，包括位置、背景颜色、边距、边框、阴影等。
5. **output-area**: 设置显示翻译结果的区域的样式，包括显示方式、最小高度、边距、边框、背景颜色、阴影等。
6. **trans_history_area**: 设置显示用户的翻译历史的区域的样式，包括边距、标题的样式、列表项的样式等。

```

1 .container {
2     position: fixed;
3     top: 0;

```

```
4   left: 0;
5   width: 100%;
6   /* height: 100%; */
7   height: 500rpx;
8   /* background: linear-gradient(to bottom, #1493FC 40%, #FFFFFF 60%); */
9   background: #1493FC;
10 }
11
12 .container .logo {
13   position: fixed;
14   display: flex;
15   margin-top: 20rpx;
16   width: 387rpx;
17   height: 211rpx;
18   left: 20rpx;
19   top: 0rpx;
20 }
21
22 .container .background {
23   position: fixed;
24   display: flex;
25   width: 312rpx;
26
27   height: 288rpx;
28   right: 0;
29   top: 60rpx;
30 }
31
32 .change {
33   color: #8995a1;
34   font-size: 24 rpx;
35   /* padding: 20rpx 40rpx; */
36   display: block;
37   align-items: center;
38   justify-content: space-between;
39   margin: 30rpx 50rpx 0;
40   margin-top: 220rpx;
41   padding: 32rpx 18rpx;
42   background-color: #1265dd;
43   border-radius: 16rpx;
44 }
45
46 .change .icon-right {
47   color: #888;
48 }
49
50 .change .icon-down {
51   color: #8995a1;
52   font-size: 20rpx;
53 }
54
55 .change .switch_language_from {
56   position: fixed;
57   display: flex;
58   flex-direction: row;
59   margin-left: 6rpx;
```

```
60  top: 265rpx;
61  left: 100rpx;
62  align-items: center;
63  justify-content: center;
64 }
65
66 .change .switch {
67 /* display: flex; */
68 display: flex;
69 align-items: center;
70 justify-content: flex-start;
71 flex-direction: column;
72 border-radius: 16rpx;
73 width: 72rpx;
74 height: 72rpx;
75 margin: auto;
76 }
77
78 .change .switch_language_to {
79 position: fixed;
80 display: flex;
81 flex-direction: row;
82 top: 265rpx;
83 right: 100rpx;
84 align-items: center;
85 justify-content: center;
86 vertical-align: text-bottom;
87 }
88
89
90 .input-area {
91 position: relative;
92 }
93
94 .textarea-wrap {
95 background: #fff;
96 margin: 50rpx;
97 border-bottom: 1px solid #c7cee0;
98 padding: 0 32rpx;
99 background-color: #ffffff;
100 border-radius: 16rpx;
101 box-shadow: 0px 0px 2rpx #000000a, 0px 0px 4rpx #000000f, 0px 8rpx
16rpx #000000a;
102 height: 300rpx;
103 left: 6.13%;
104 right: 6.67%;
105 top: 53.72%;
106 bottom: 0.55%;
107 color: #1e3163;
108 line-height: 34rpx;
109 font-size: 36rpx;
110 font-family: Poppins;
111 }
112
113 .language_from {
114 bottom: auto;
```

```
115 font-size: 30rpx;
116 font-family: Poppins;
117 line-height: 26rpx;
118 color: #a8abb0;
119 margin-top: 10rpx;
120 margin-bottom: 10rpx;
121 font-family: Poppins;
122 line-height: 22rpx;
123 }
124
125 .language_pic {
126 margin-top: 20rpx;
127 width: 40rpx;
128 height: 40rpx;
129 vertical-align: text-bottom;
130 }
131
132 .language_from_text {
133 font-size: 30rpx;
134 font-family: Poppins;
135 line-height: 26rpx;
136 color: #a8abb0;
137 margin-top: 10rpx;
138 margin-bottom: 10rpx;
139 margin-left: 10rpx;
140 font-family: Poppins;
141 line-height: 22rpx;
142 }
143
144 .input-area textarea {
145 background-color: #fff;
146 padding: 30rpx 0 30rpx 10rpx;
147 /* width: calc(100% - 48rpx); */
148 width: auto;
149 margin: 0;
150 box-sizing: border-box;
151 height: 210rpx;
152 border-bottom: solid 2rpx #e7e7e7;
153 padding-right: 0;
154 }
155
156
157 .input-area .icon-close {
158 position: absolute;
159 right: 12rpx;
160 top: 20rpx;
161 z-index: 100;
162 font-size: 40rpx;
163 color: #888;
164 }
165
166 .voice {
167 position: fixed;
168 margin-top: -40rpx;
169 width: 40rpx;
170 height: 40rpx;
```

```
171 vertical-align: text-bottom;
172 text-align: end;
173 right: 150rpx;
174 }
175
176 .copy {
177 position: fixed;
178 margin-top: -45rpx;
179 width: 50rpx;
180 height: 50rpx;
181 right: 100rpx;
182 display: flex;
183 align-items: center;
184 justify-content: center;
185 }
186
187
188 .play-icon {
189 position: absolute;
190 right: 3rpx;
191 bottom: 7rpx;
192 padding: 0 8rpx;
193 display: flex;
194 align-items: center;
195 }
196
197 .edit-icon::before .play-icon::before {
198 content: "";
199 position: absolute;
200 top: -10rpx;
201 left: -10rpx;
202 bottom: -10rpx;
203 right: -10rpx;
204 }
205
206
207 .input-area .output-area {
208 display: flex;
209 flex-direction: column;
210 min-height: 150rpx;
211 margin: 50rpx;
212 border-bottom: 1px solid #c7cee0;
213 padding: 0 32rpx;
214 background-color: #1493FC;
215 border-radius: 16rpx;
216 box-shadow: 0px 0px 2rpx #000000a, 0px 0px 4rpx #000000f, 0px 8rpx
217 16rpx #000000a;
218 height: 300rpx;
219 left: 24px;
220 right: 24px;
221 top: 384px;
222 bottom: 262px;
223 }
224
225 .output-area .text-result {
226 min-height: 150rpx;
```

```
226 /* margin-top: 20rpx; */
227 /* padding: 20rpx 0; */
228 height: 150rpx;
229 /* border-bottom: solid 2rpx #e7e7e7; */
230 color: #ffffff;
231 line-height: 32rpx;
232 font-size: 36rpx;
233 font-family: Poppins;
234 line-height: 33rpx;
235 }
236
237 .output-area .result {
238 margin-bottom: 10px;
239 /* 这里的值可以根据你的需求来调整 */
240 min-height: 150rpx;
241 margin-top: 20rpx;
242 padding: 20rpx 0;
243 height: 150rpx;
244 border-bottom: solid 2rpx #e7e7e7;
245 color: #ffffff;
246 line-height: 32rpx;
247 font-size: 36rpx;
248 font-family: Poppins;
249 line-height: 33rpx;
250 }
251
252 .language_to {
253 /* bottom: auto; */
254 font-size: 30rpx;
255 font-family: Poppins;
256 line-height: 26rpx;
257 color: #a8abb0;
258 /* margin-top: 10rpx; */
259 margin-bottom: 10rpx;
260 font-family: Poppins;
261 line-height: 22rpx;
262 }
263
264 .language_to_text {
265 font-size: 30rpx;
266 font-family: Poppins;
267 line-height: 26rpx;
268 color: #ffffff;
269 margin-top: 10rpx;
270 margin-bottom: 10rpx;
271 margin-left: 10rpx;
272 font-family: Poppins;
273 line-height: 22rpx;
274 }
275
276 .trans_history_area {
277 margin: 55rpx;
278 }
279
280 .trans_history_area .trans_history_title {
281 align-items: center;
```

```
282 justify-content: space-between;
283 display: flex;
284 flex-direction: row;
285 }
286
287 .trans_history_area .trans_history_title .trans_history_title_chi {
288   color: transparent;
289   font-size: 32rpx;
290   font-family: Poppins;
291   font-weight: 700;
292   line-height: 30rpx;
293   background-image: linear-gradient(180deg, #0064e1 0%, #0845c2 100%);
294   -webkit-background-clip: text;
295 }
296
297 .trans_history_area .trans_history_title .trans_history_title_eng {
298   color: #949494;
299   line-height: 24rpx;
300   font-style: italic;
301   font-family: Poppins;
302   line-height: 22rpx;
303 }
304
305 .item {
306   margin-top: 40rpx;
307   padding: 0 10rpx;
308   background-color: #ffffff;
309   border-radius: 16rpx;
310   box-shadow: 0px 0px 2rpx #000000a, 0px 0px 4rpx #000000f, 0px 8rpx
311   16rpx #000000a;
312 }
313
314 .item .query {
315   padding: 20rpx;
316   border-bottom: solid 2rpx #e7e7e7;
317   font-size: 32rpx;
318   font-family: Poppins;
319   line-height: 50rpx;
320   color: #1e3163;
321 }
322
323 .item .query .language {
324   font-size: 28rpx;
325   font-family: Poppins;
326   line-height: 26rpx;
327   color: #a8abb0;
328   margin-top: 10rpx;
329   margin-bottom: 10rpx;
330 }
331
332 .item .result {
333   margin-top: 16rpx;
334   padding: 20rpx;
335   font-size: 32rpx;
336   font-family: Poppins;
337   line-height: 50rpx;
```

```

337     color: #1e3163;
338 }
339
340 .item .result .language {
341   font-size: 28rpx;
342   font-family: Poppins;
343   line-height: 26rpx;
344   color: #909dbd;
345   margin-bottom: 10rpx;
346 }
347
348 image {
349   width: 30rpx;
350   height: 30rpx;
351   vertical-align: bottom;
352 }

```

4.1.4.6 pages/OCR

1 OCR.js

这段代码的主要功能是：

1. 上传图片：用户可以通过点击按钮上传图片，然后跳转到另一个页面进行图片的选择和处理。
2. 获取OCR：通过调用API，将上传的图片进行OCR识别，提取出图片中的文字，并将识别结果进行翻译。
3. 返回按钮：用户可以通过点击返回按钮返回到上一个页面。

```

1 // 导入翻译工具
2 import { translate } from '../../utils/api.js'
3
4 // 获取全局应用程序实例对象
5 const app = getApp();
6
7 // 定义页面
8 Page({
9   /**
10    * 页面的初始数据
11    */
12   data: {
13     src: "", // 图片源
14     sourceText: [], // 原始文本
15     resultText: "", // 翻译结果
16     imgList: [], // 图片列表
17     filePath: '', // 文件路径
18     picBase64: '', // 图片的base64编码
19     textSrc: '', // 原文本
20     textDst: '' // 目标文本
21   },
22
23   // 页面显示时的回调函数
24   onShow: function () {
25     // 如果全局变量中的图片base64编码不为空
26     if (app.globalData.picBase64 != "") {
27       // 更新数据

```

```
28     this.setData({
29         src: app.globalData.picBase64,
30     })
31 }
32 },
33
34 // 上传图片的函数
35 uploadImg: function () {
36     // 导航到获取图片的页面
37     wx.navigateTo({
38         url: '../getPic/getPic',
39     })
40 },
41
42 // 获取OCR的函数
43 getOCR: function () {
44     // 提取图片里的文字
45     var that = this;
46     wx.request({
47         url: 'https://api.jisuapi.com/generalrecognition/recognize?
appkey=',
48         data: {
49             pic: this.data.src,
50             type: "cnen"
51         },
52         method: 'post',
53         header: {
54             'content-type': 'application/x-www-form-urlencoded' // 默认值
55         },
56         success: function (res) {
57             console.log(res);
58             that.setData({
59                 sourceText: res.data.result
60             })
61             let str = that.data.sourceText.join()
62             console.log(str)
63             translate(str, {
64                 from: 'zh',
65                 to: 'en'
66             }).then(res => {
67                 console.log(res)
68                 that.setData({
69                     resultText: res.trans_result[0].dst
70                 })
71             })
72         },
73         fail: function (res) {
74             console.log(res);
75         }
76     })
77 },
78
79 // 返回按钮的点击事件处理函数
80 onBackIconTap: function () {
81     // 导航回上一页
82     wx.navigateBack({
```

```
83     delta: 1, // 返回的页面层数
84   );
85 }
86 })
87
```

2 OCR.json

缺省代码。

```
1 {
2   "usingComponents": {}
3 }
```

3 OCR.wxml

这段代码的主要功能是：

1. 显示一个返回按钮，用户可以通过点击返回按钮返回到上一个页面。
2. 显示一个图片，图片的源数据是base64编码的。
3. 提供一个按钮，用户可以通过点击按钮触发OCR识别和翻译的功能。
4. 显示OCR识别和翻译的结果，包括源语言（中文）的文本和目标语言（英语）的文本。

```
1 <!-- 头部视图 -->
2 <view class="head">
3   <!-- 返回图标视图，点击时触发onBackIconTap事件 -->
4   <view id="back-icon" class="back-icon" bindtap="onBackIconTap" >
5     <!-- 返回箭头图标 -->
6     <image src="../../imgs/arrow-left.png" class="arrow-icon"></image>
7   </view>
8 </view>
9
10 <!-- 容器视图 -->
11 <view class="container">
12   <!-- 图片视图，图片源为base64编码的数据，宽度为100%，高度为1000rpx -->
13   <image src="data:image/png;base64,{{src}}"
14     style="width:100%;height:1000rpx"></image>
15
16   <!-- 翻译图片按钮，点击时触发getOCR事件 -->
17   <view bindtap="getOCR" class="btn">
18     翻译图片
19   </view>
20
21   <!-- 项目视图 -->
22   <view class="item">
23     <!-- 查询视图 -->
24     <view class="query">
25       <!-- 语言视图，包含一个图标和文本“中文” -->
26       <view class="language">
27         <image src="../../imgs/Chinese.png" />
28         中文
29       </view>
30       <!-- 显示源文本 -->
31       <text>{{sourceText}}</text>
```

```
31 </view>
32
33     <!-- 结果视图 -->
34     <view class="result">
35         <!-- 语言视图，包含一个图标和文本“英语” -->
36         <view class="language">
37             <image src="../../imgs/English.png" />
38             英语
39         </view>
40         <!-- 显示翻译结果文本 -->
41         <text>{{resultText}}</text>
42     </view>
43 </view>
44 </view>
```

4 OCR.wxss

这段代码定义了一些样式规则，用于美化小程序中的各个元素的外观和布局。

其中包括头部视图、返回图标、箭头图标、按钮、项目视图、查询视图、查询语言、结果视图、结果语言和图片的样式。

通过设置不同的样式属性，如位置、大小、颜色等，可以使页面元素呈现出不同的效果，增强用户体验。

```
1 /* 容器样式 */
2 .container {
3     /* 居中对齐 */
4     text-align: center;
5 }
6
7 /* 头部样式 */
8 .head {
9     /* 相对定位 */
10    position: relative;
11    /* 顶部和左侧边距为0，宽度为100%，高度为200rpx */
12    top: 0;
13    left: 0;
14    width: 100%;
15    height: 200rpx;
16    /* 背景颜色为#1493FC */
17    background-color: #1493FC;
18 }
19
20 /* 返回图标样式 */
21 .back-icon {
22     /* 绝对定位，位于头部视图的左上角 */
23     position: absolute;
24     top: 50px;
25     left: 20px;
26     /* z-index用于控制元素的层叠顺序 */
27     z-index: 1;
28     /* 背景颜色为#1493FC */
29     background-color: #1493fc;
30 }
31
```

```
32 /* 箭头图标样式 */
33 .arrow-icon {
34   width: 30px;
35   height: 30px;
36 }
37
38 /* 按钮样式 */
39 .btn {
40   /* 背景颜色为#1493FC */
41   background-color: #1493fc;
42   /* 上下内边距为15rpx, 左右内边距为20rpx */
43   padding: 15rpx 20rpx;
44   /* 文本颜色为白色 */
45   color: #fff;
46   /* 圆角半径为20rpx */
47   border-radius: 20rpx;
48   /* 外边距为40rpx */
49   margin: 40rpx;
50 }
51
52 /* 项目样式 */
53 .item {
54   /* 上外边距为40rpx, 左右内边距为10rpx */
55   margin-top: 40rpx;
56   padding: 0 10rpx;
57   /* 背景颜色为白色 */
58   background-color: #ffffff;
59   /* 圆角半径为16rpx */
60   border-radius: 16rpx;
61   /* 阴影效果 */
62   box-shadow: 0px 0px 2rpx #0000000a, 0px 0px 4rpx #0000000f, 0px 8rpx
63   16rpx #0000000a;
64 }
65
66 /* 查询样式 */
67 .item .query {
68   /* 内边距为20rpx */
69   padding: 20rpx;
70   /* 底部边框为实线, 宽度为2rpx, 颜色为#e7e7e7 */
71   border-bottom: solid 2rpx #e7e7e7;
72   /* 字体大小为32rpx */
73   font-size: 32rpx;
74   /* 字体家族为Poppins */
75   font-family: Poppins;
76   /* 行高为50rpx */
77   line-height: 50rpx;
78   /* 字体颜色为#1e3163 */
79   color: #1e3163;
80 }
81
82 /* 查询语言样式 */
83 .item .query .language {
84   /* 字体大小为28rpx */
85   font-size: 28rpx;
86   /* 字体家族为Poppins */
87   font-family: Poppins;
```

```

87  /* 行高为26rpx */
88  line-height: 26rpx;
89  /* 字体颜色为#a8abb0 */
90  color: #a8abb0;
91  /* 上外边距为10rpx, 下外边距为10rpx */
92  margin-top: 10rpx;
93  margin-bottom: 10rpx;
94 }
95
96 /* 结果样式 */
97 .item .result {
98  /* 上外边距为16rpx */
99  margin-top: 16rpx;
100 /* 内边距为20rpx */
101 padding: 20rpx;
102 /* 字体大小为32rpx */
103 font-size: 32rpx;
104 /* 字体家族为Poppins */
105 font-family: Poppins;
106 /* 行高为50rpx */
107 line-height: 50rpx;
108 /* 字体颜色为#1e3163 */
109 color: #1e3163;
110 }
111
112 /* 结果语言样式 */
113 .item .result .language {
114  /* 字体大小为28rpx */
115  font-size: 28rpx;
116  /* 字体家族为Poppins */
117  font-family: Poppins;
118  /* 行高为26rpx */
119  line-height: 26rpx;
120  /* 字体颜色为#909dbd */
121  color: #909dbd;
122  /* 下外边距为10rpx */
123  margin-bottom: 10rpx;
124 }
125
126 /* 图片样式 */
127 image {
128  width: 30rpx;
129  height: 30rpx;
130  /* 垂直对齐方式为底部对齐 */
131  vertical-align: bottom;
132 }
133

```

4.1.4.7 pages/voice_translation

语音翻译页面，主要功能是通过微信的语音插件实现语音的录制、识别和翻译。

以下代码的主要功能：

1. 语音录制：用户按下按钮时，开始进行语音录制，录制过程中会实时显示识别结果。当用户松开按钮时，结束录制，并进行语音识别。

- 2. 语音识别：语音识别的结果会添加到对话列表中，显示在页面上。如果识别结果为空，则会显示提示信息。
- 3. 语音翻译：识别后的文字会被发送到插件进行翻译，翻译的结果会被更新到对话列表中的相应位置。
- 4. 语音播放：当翻译完成后，页面会自动播放翻译后的语音。如果语音文件过期，会重新进行语音合成。
- 5. 语言切换：用户可以点击按钮切换输入语言，支持中英文切换。
- 6. 历史记录：用户的历史录音和翻译记录会被保存，在用户再次进入页面时可以查看。
- 7. 滚动显示：当识别或翻译的内容添加到对话列表时，页面会自动滚动到最新的内容。

1 voice_translation.js

页面数据的配置

```

1 // 获取应用实例
2 const app = getApp()
3
4 // 引入工具库
5 const util = require('../utils/util.js')
6
7 // 引入微信语音插件
8 const plugin = requirePlugin("wechartsI")
9
10 // 引入语言配置文件
11 import { language } from '../utils/conf.js'
12
13 // 获取全局唯一的语音识别管理器
14 const manager = plugin.getRecordRecognitionManager()
15
16 Page({
17   // 页面的初始数据
18   data: {
19     dialogList: [], // 对话列表，初始为空
20     lan_type: true, // 语言类型
21     scroll_top: 10000, // 竖向滚动条位置
22     bottomButtonDisabled: false, // 底部按钮是否禁用
23     tips_language: language[0], // 提示语言，初始为中文
24     // 初始时的翻译卡片
25     initTranslate: {
26       create: '04/27 15:37',
27       text: '等待说话',
28     },
29     // 当前的翻译卡片
30     currentTranslate: {
31       create: '04/27 15:37',
32       text: '等待说话',
33     },
34     recording: false, // 是否正在录音
35     recordStatus: 0, // 录音状态: 0 - 录音中 1- 翻译中 2 - 翻译完成/二次翻译
36     toView: 'fake', // 滚动位置
37     lastId: -1, // dialogList 最后一个item的 id
38     currentTranslateVoice: '', // 当前播放语音路径
39     // 图片路径

```

```

40     image_c:'https://codefun-proj-user-res-1256085488.cos.ap-
41     guangzhou.myqcloud.com/644bb0005a7e3f03102917b5/644bb06fb98f5d0011665f39/153
42     cd789341a2b9a6a2d1ac163978ba0.png',
43     image_e:'https://codefun-proj-user-res-1256085488.cos.ap-
44     guangzhou.myqcloud.com/644bb0005a7e3f03102917b5/644bb06fb98f5d0011665f39/e9f
45     c70c625980d75443bf2ae1516d24f.png'
46   },

```

录音按钮函数streamRecord, streamRecordEnd, 当用户松开按钮后, 结束语音识别。函数首先检查是否已经在录音, 如果没有, 或者已经在录音但录音状态不为0 (这可能意味着录音已经被停止) , 则返回。否则, 它会停止录音, 并禁用底部按钮以防止重复停止。

接着是一个函数changelanguage, 该函数用于切换语音识别的语言。它首先切换lan_type的值 (这可能是一个布尔值, 表示使用的是哪种语言) 。然后, 根据新的lan_type的值, 设置适当的初始化翻译文本。

```

1 // 按住按钮开始语音识别
2 streamRecord: function(e) {
3   // 开始语音识别
4   manager.start({
5     lang: e.detail.buttonItem.lang,
6   })
7
8   // 根据语言类型设置翻译卡片内容
9   let lan_type = this.data.lan_type
10  this.setData({
11    recordStatus: 0,
12    recording: true,
13    currentTranslate: {
14      create: util.recordTime(new Date()),
15      text: lan_type ? '正在聆听中':'listening',
16      lfrom: lan_type ? e.detail.buttonItem.lang :
e.detail.buttonItem.lto,
17      lto: lan_type ? e.detail.buttonItem.lto : e.detail.buttonItem.lang,
18    },
19  })
20  this.scrollToNew();
21
22 },
23
24 /**
25 * 松开按钮结束语音识别
26 */
27 streamRecordEnd: function(e) {
28   let detail = e.detail || {} // 自定义组件触发事件时提供的detail对象
29   let buttonItem = detail.buttonItem || {}
30
31   // 防止重复触发stop函数
32   if(!this.data.recording || this.data.recordStatus != 0) {
33     console.warn("has finished!")
34     return
35   }
36
37   // 停止录音识别
38   manager.stop()

```

```

39     // 禁用底部按钮
40     this.setData({
41         bottomButtonDisabled: true,
42     })
43 },
44
45
46 // 切换语言
47 changeLanguage: function(){
48     this.setData({
49         lan_type: !this.data.lan_type,
50     })
51
52     if(this.data.lan_type) {
53         this.setData({
54             initTranslate: {
55                 create: util.recordTime(new Date()),
56                 text: '等待说话',
57             },
58         })
59     } else {
60         this.setData({
61             initTranslate: {
62                 create: util.recordTime(new Date()),
63                 text: 'Please Speaking',
64             },
65         })
66     }
67 }

```

以下代码定义了函数 translateText，该函数用于翻译文本。它接受两个参数：一个是要翻译的文本项，另一个是该项在对话列表中的索引。

这个函数首先确定源语言和目标语言，默认为从中文到英文。然后，它调用翻译插件，传入要翻译的文本，并启用文本到语音功能。

如果翻译失败，fail 回调函数会被调用，并记录一个错误。无论成功或失败，complete 回调函数都会被调用，用来更新录音状态，并隐藏加载提示。

```

1 /**
2 * 翻译
3 */
4 translateText: function(item, index) {
5     let lfrom = item.lfrom || 'zh_CN' // 原语言，默认为中文
6     let lto = item.lto || 'en_US' // 目标语言，默认为英文
7
8     // 调用翻译插件
9     plugin.translate({
10         lfrom: lfrom,
11         lto: lto,
12         content: item.text, // 要翻译的文本
13         tts: true, // 启用文本到语音
14         success: (resTrans)=>{ // 翻译成功的回调函数
15
16             let passRetcode = [
17                 0, // 翻译合成成功

```

```

18         -10006, // 翻译成功, 合成失败
19         -10007, // 翻译成功, 传入了不支持的语音合成语言
20         -10008 // 翻译成功, 语音合成达到频率限制
21     ]
22
23     // 如果返回的结果是可接受的
24     if(passRetcode.indexOf(resTrans.retcode) >= 0 ) {
25         let tmpDialogList = this.data.dialogList.slice(0)
26
27         // 如果索引有效
28         if(!isNaN(index)) {
29             // 更新当前条目的翻译结果
30             let tmpTranslate = Object.assign({}, item, {
31                 autoplay: true, // 自动播放背景音乐
32                 translateText: resTrans.result, // 翻译结果
33                 translatevoicePath: resTrans.filename || "", // 语音文件路径
34                 translateVoiceExpiredTime: resTrans.expired_time || 0 // 语音
35             })
36
37             tmpDialogList[index] = tmpTranslate
38
39             // 更新对话列表和底部按钮状态
40             this.setData({
41                 dialogList: tmpDialogList,
42                 bottomButtonDisabled: false,
43                 recording: false,
44             })
45
46             // 滚动到新的位置
47             this.scrollToNew();
48
49         } else {
50             console.error("index error", resTrans, item)
51         }
52     } else {
53         console.warn("翻译失败", resTrans, item)
54     }
55
56 },
57 fail: function(resTrans) { // 翻译失败的回调函数
58     console.error("调用失败", resTrans, item)
59     this.setData({
60         bottomButtonDisabled: false,
61         recording: false,
62     })
63 },
64 complete: resTrans => { // 翻译完成的回调函数, 无论成功或失败
65     this.setData({
66         recordStatus: 1,
67     })
68     wx.hideLoading() // 隐藏加载提示
69 }
70 }
71
72 },

```

在下面的代码段中，定义了函数translateTextAction，这个函数会在修改文本信息后触发，它调用translateText函数进行翻译。

定义了expiredAction函数，这个函数用于处理语音文件过期的情况，它会调用插件的文本到语音功能重新生成语音文件。

```

1  /**
2  * 修改文本信息后触发翻译操作
3  */
4  translateTextAction: function(e) {
5      // 获取由自定义组件触发事件提供的detail对象
6      let detail = e.detail
7      let item = detail.item
8      let index = detail.index
9
10     // 调用翻译函数
11     this.translateText(item, index)
12 },
13
14 /**
15 * 语音文件过期，重新合成语音文件
16 */
17 expiredAction: function(e) {
18     // 获取由自定义组件触发事件提供的detail对象
19     let detail = e.detail || {}
20     let item = detail.item || {}
21     let index = detail.index
22
23     // 检查索引是否有效
24     if(isNaN(index) || index < 0) {
25         return
26     }
27
28     // 设定目标语言，默认为英语
29     let lto = item.lto || 'en_us'
30
31     // 调用插件的文本到语音功能
32     plugin.textToSpeech({
33         lang: lto,
34         content: item.translateText,
35         success: resTrans => {
36             if(resTrans.retcode == 0) {
37                 let tmpDialogList = this.data.dialogList.slice(0)
38
39                 // 用新的属性更新对应的条目
40                 let tmpTranslate = Object.assign({}, item, {
41                     autoplay: true, // 自动播放背景音乐
42                     translateVoicePath: resTrans.filename, // 语音文件路径
43                     translateVoiceExpiredTime: resTrans.expired_time || 0 // 语音文件
44                     过期时间
45                 })
46
47                 tmpDialogList[index] = tmpTranslate

```

```

48         // 更新对话列表
49         this.setData({
50             dialogList: tmpDialogList,
51         })
52
53     } else {
54         console.warn("语音合成失败", resTrans, item)
55     }
56 },
57 fail: function(resTrans) {
58     console.warn("语音合成失败", resTrans, item)
59 }
60 })
61 ,

```

定义了initCard函数，这个函数用于初始化一张空白的卡片。

然后定义了deleteItem函数，用于删除某一条目。

当列表为空时，deleteItem函数将会调用initCard函数创建一张空白的卡片。

```

1 /**
2 * 初始化为空时的卡片
3 */
4 initCard: function () {
5     // 创建新的初始化翻译对象，并添加当前时间
6     let initTranslateNew = Object.assign({}, this.data.initTranslate, {
7         create: util.recordTime(new Date()),
8     })
9
10    // 更新数据
11    this.setData({
12        initTranslate: initTranslateNew
13    })
14 },
15
16 /**
17 * 删除卡片
18 */
19 deleteItem: function(e) {
20     // 获取由自定义组件触发事件提供的detail对象
21     let detail = e.detail
22     let item = detail.item
23
24     // 创建一个新的对话列表副本
25     let tmpDialogList = this.data.dialogList.slice(0)
26     let arrIndex = detail.index
27
28     // 删除对应索引的元素
29     tmpDialogList.splice(arrIndex, 1)
30
31     // 使用setTimeout来避免可能的错误：Expect END descriptor with depth 0 but
32     // get another
33     setTimeout( ()=>{
34         this.setData({
35             dialogList: tmpDialogList
36         })
37     })
38 }
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135

```

```

36     // 如果列表为空，则初始化卡片
37     if(tmpDialogList.length == 0) {
38         this.initCard()
39     }
40     }, 0)
41 },
42
43 /**
44 * 识别内容为空时的反馈
45 */
46 showRecordEmptyTip: function() {
47     // 更新数据
48     this.setData({
49         recording: false,
50         bottomButtonDisabled: false,
51     })
52
53     // 显示提示
54     wx.showToast({
55         title: this.data.tips_language.recognize_nothing,
56         icon: 'success',
57         image: '/image/no_voice.png',
58         duration: 1000,
59         success: function (res) {
60
61     },
62     fail: function (res) {
63         console.log(res);
64     }
65 });
66 },
67
68
69 /**
70 * 初始化语音识别回调
71 * 绑定语音播放开始事件
72 */
73 initRecord: function() {
74     // 有新的识别内容返回，则会调用此事件
75     manager.onRecognize = (res) => {
76         let currentData = Object.assign({}, this.data.currentTranslate, {
77             text: res.result,
78         })
79         // 更新当前翻译内容
80         this.setData({
81             currentTranslate: currentData,
82         })
83         // 滚动到新的内容
84         this.scrollToNew();
85     }
86
87     // 识别结束事件
88     manager.onStop = (res) => {
89         let text = res.result
90
91         // 如果结果为空，显示提示信息

```

```

92     if(text == '') {
93         this.showRecordEmptyTip()
94         return
95     }
96
97     let lastId = this.data.lastId + 1
98
99     let currentData = Object.assign({}, this.data.currentTranslate, {
100             text: res.result,
101             translateText: '正在翻译中',
102             id: lastId,
103             voicePath: res.tempFilePath
104         })
105
106     // 更新当前翻译和记录状态
107     this.setData({
108         currentTranslate: currentData,
109         recordStatus: 1,
110         lastId: lastId,
111     })
112
113     // 滚动到新的内容
114     this.scrollToNew();
115
116     // 开始翻译
117     this.translateText(currentData, this.data.dialogList.length)
118 }
119
120 // 识别错误事件
121 manager.onError = (res) => {
122     // 如果发生错误，停止录音并启用底部按钮
123     this.setData({
124         recording: false,
125         bottomButtonDisabled: false,
126     })
127 }
128
129 // 语音播放开始事件
130 wx.getBackgroundAudioManager (res=>{
131     const backgroundAudioManager = wx.getBackgroundAudioManager()
132     let src = backgroundAudioManager.src
133
134     // 更新当前播放的音频源
135     this.setData({
136         currentTranslateVoice: src
137     })
138 }
139 },
140

```

以下代码包含了对历史记录的获取和设置。

```

1 /**
2 * 设置语音识别历史记录
3 */

```

```

4  setHistory: function() {
5    try {
6      let dialogList = this.data.dialogList
7      dialogList.forEach(item => {
8        item.autoPlay = false
9      })
10     // 尝试将对话列表存储在本地
11     wx.setStorageSync('history', dialogList)
12   } catch (e) {
13     console.error("setStorageSync setHistory failed")
14   }
15 },
16 /**
17 * 得到历史记录
18 */
19 getHistory: function() {
20   try {
21     // 尝试从本地获取历史记录
22     let history = wx.getStorageSync('history')
23     if (history) {
24       let len = history.length;
25       let lastId = this.data.lastId
26       if(len > 0) {
27         lastId = history[len-1].id || -1;
28       }
29       // 如果历史记录存在，将其设置到dialogList中
30       this.setData({
31         dialogList: history,
32         toView: this.data.toView,
33         lastId: lastId,
34       })
35     }
36   } catch (e) {
37     // 如果出现错误，将dialogList设置为空列表
38     this.setData({
39       dialogList: []
40     })
41   }
42 },
43 },
44 /**
45 * 重新滚动到底部
46 */
47 scrollToNew: function() {
48   // 更新视图到最新的位置
49   this.setData({
50     toView: this.data.toView
51   })
52 },
53 },
54 onShow: function() {
55   // 当页面显示时，滚动到最新的位置，并初始化卡片
56   this.scrollToNew();
57   this.initCard()
58 },
59

```

```

60     if(this.data.recordStatus == 2) {
61         wx.showLoading({
62             // title: '',
63             mask: true,
64         })
65     }
66 },
67
68 onLoad: function () {
69     // 页面加载时，获取历史记录并初始化语音识别
70     this.getHistory()
71     this.initRecord()
72     this.setData({toView: this.data.toView})
73     app.getRecordAuth()
74 },
75
76 onHide: function() {
77     // 页面隐藏时，设置历史记录
78     this.setHistory()
79 },
80
81 /**
82 * 定义一个点击事件处理函数
83 */
84 onBackIconTap: function() {
85     // 返回上一页
86     wx.navigateBack({
87         delta: 1, // 返回的页面层数
88     });
89 },
90
91 })
92
93
94

```

2 voice_translation.json

```

1 {
2     "usingComponents": {
3         "bottom-button": "/components/bottom-button/index",
4         "result-bubble": "/components/result-bubble/index"
5     }
6 }
7

```

3 voice_translation.wxml

```

1 <!--index.wxml-->
2
3 <!-- 页面背景部分 -->
4 <view class="page-background">
5     <image class="image self-center" src="../../imgs/voice_background.png">
6     </image>
7 </view>

```

```

7      <!-- 主体部分，包含对话框和底部按钮 -->
8      <view class="container">
9
10     <!-- 对话框部分 -->
11     <scroll-view id="scroll-content"
12       scroll-top="{{scroll_top}}"
13       scroll-y="true"
14       class="dialog-part"
15       scroll-into-view="translate-{{toView}}"
16       enable-back-to-top="true"
17       scroll-with-animation="true">
18
19     <!-- 空白间隔部分 -->
20     <view class="spacer"></view>
21
22
23     <!-- 返回图标，点击触发 onBackIconTap 函数 -->
24     <view id="back-icon" class="back-icon" bindtap="onBackIconTap" >
25       <image src="../../imgs/arrow-left.png" class="arrow-icon"></image>
26     </view>
27
28     <!-- 语言切换部分 -->
29     <view class="flex-row justify-between items-center relative group_2">
30       <view class="flex-row items-center self-stretch group_3 space-x-8">
31         <!-- 根据 lan_type 的值，显示不同的图片和文本 -->
32         <image wx:if="{{lan_type}}" class="shrink-0 image_1" src=
33           "{{image_c}} />
34         <image wx:else class="shrink-0 image_1" src="{{image_e}} />
35         <text class="text" wx:if="{{lan_type}}>中文</text>
36         <text class="text" wx:else>英文</text>
37
38         <!-- 语言切换按钮，点击触发 changelanguage 函数 -->
39         <view bindtap="changelanguage" style="display: inline-block">
40           <image class="image_1 image_2" src="../../imgs/Vector.png"/>
41         </view>
42       </view>
43     </view>
44
45     <!-- 列表部分 -->
46     <!-- 列表为空时显示的部分 -->
47     <view class="dialog-wrap" id="translate-empty" wx:if="{{!recording &&
48       dialogList.length == 0}}>
49       <result-bubble item="{{initTranslate}}" record-status="0"></result-
50       bubble>
51     </view>
52
53     <!-- 列表部分 -->
54     <view wx:for="{{dialogList}}" wx:key="id" class="dialog-wrap" data-
55       index="{{index}}" catchmodaldelete="deleteItem">
56       <result-bubble item="{{item}}"
57         edit-show="{{index==dialogList.length-1}}"
58         index="{{index}}"
59         current-translate-voice="{{currentTranslateVoice}}"
60         bindtranslate="translateTextAction"
61         bindexpired="expiredAction"></result-bubble>
62     </view>

```

```

59      <!-- 正在录音时显示的部分 -->
60      <view class="dialog-wrap" id="translate-recording" wx:if="{{recording}}>
61          <result-bubble item="{{currentTranslate}}" record-status="{{recordStatus}}></result-bubble>
62      </view>
63
64      <view id="translate-fake"></view>
65
66  </scroll-view>
67
68
69      <!-- 底部按钮部分 -->
70      <view class="foot-group" catchlongpress="catchTapEvent">
71          <bottom-button button-disabled="{{bottomButtonDisabled}}"
72              bindrecordstart="streamRecord"
73              bindrecordend="streamRecordEnd"></bottom-button>
74      </view>
75  </view>
76

```

3 voice_translation.wxss

```

1  /* 主容器样式 */
2  .container {
3      height: 100%;
4      display: flex;
5      flex-direction: column;
6      justify-content: space-between;
7      box-sizing: border-box;
8      position: relative;
9      font-family: "PingFang-SC-Regular", "SimSun", "Microsoft Yahei";
10     background-color: #1494fc;
11 }
12
13 /* 页面样式 */
14 page {
15     height: 100%;
16     width: 100%;
17     background: #FAFAFA;
18 }
19
20 /* 输入框样式 */
21 input {
22     font-family: "PingFang-SC-Regular", "SimSun", "Microsoft Yahei";
23 }
24
25 /* flex布局，列方向 */
26 .flex-column {
27     display: flex;
28     flex-direction: column;
29     align-items: center;
30     justify-content: space-between;
31 }
32

```

```
33 /* spacer元素样式 */
34 .spacer {
35   height: 246rpx;
36 }
37
38 /* 页面背景样式 */
39 .page-background {
40   position: fixed;
41   top: 345px;
42   left: 0;
43   width: 100%;
44   height: 100%;
45   background-position: center;
46   background-size: cover;
47   z-index: 1;
48 }
49
50 /* 图片样式 */
51 .image {
52   width: 750rpx;
53   height: 348rpx;
54 }
55
56 /* 设置元素相对于其父元素的位置为中心 */
57 .self-center {
58   align-self: center;
59 }
60
61 /* 对话框包裹样式 */
62 .dialog-wrap {
63   position: relative;
64   padding: 20rpx 40rpx 50rpx 40rpx;
65   box-sizing: border-box;
66   display: flex;
67   width: 100%;
68   flex-direction: column;
69 }
70
71 /* 消息详情文字样式 */
72 .send-message .text-detail {
73   color: #9B9B9B;
74 }
75
76 /* 对话部分样式 */
77 .dialog-part {
78   position: absolute;
79   left: 0;
80   top: 0;
81   bottom: 257rpx;
82   right: 0;
83   z-index: 1;
84 }
85
86 /* 用户输入样式 */
87 .user-input {
88   flex: 1;
```

```
89 height: 60rpx;
90 box-sizing: border-box;
91 margin: 0 10px;
92 border-radius: 10rpx;
93 }
94
95 /* 文本内容样式 */
96 .text-content {
97   margin: 0 48px 0 0;
98   box-sizing: border-box;
99 }
100
101 /* 编辑图标样式 */
102 .edit-icon {
103   position: absolute;
104   right: 10rpx;
105   bottom: 0;
106   padding: 0 8rpx;
107 }
108
109 /* 播放图标样式 */
110 .play-icon {
111   position: absolute;
112   right: 10rpx;
113   bottom: 14rpx;
114   padding: 0 8rpx;
115   display: flex;
116   align-items: center;
117 }
118
119 /* 声音图标样式 */
120 .play-loud-icon {
121   position: absolute;
122   right: 0;
123   bottom: 14rpx;
124   padding: 0 8rpx;
125   display: flex;
126   align-items: center;
127 }
128
129 /* 文字详情样式 */
130 .text-detail {
131   font-size: 18px;
132   line-height: 24px;
133   font-family: "PingFang-SC-Regular", "SimSun", "Microsoft Yahei";
134 }
135
136 /* 返回图标样式 */
137 .back-icon {
138   position: absolute;
139   top: 50px;
140   left: 20px;
141   z-index: 1;
142 }
143
144 /* 箭头图标样式 */
```

```
145 .arrow-icon {  
146   width: 30px;  
147   height: 30px;  
148 }  
149  
150 /* 翻译消息样式 */  
151 .translate-message {  
152   position: relative;  
153 }  
154  
155 /* 发送消息样式 */  
156 .send-message {  
157   position: relative;  
158 }  
159  
160 /* 创建时间样式 */  
161 .create-time {  
162   font-size: 14px;  
163   color: #B2B2B2;  
164   margin-bottom: 5px;  
165   display: flex;  
166   justify-content: center;  
167 }  
168  
169 /* 模糊滤镜样式 */  
170 .filter-blur {  
171   filter: blur(5px);  
172 }  
173  
174 /* 空提示样式 */  
175 .empty-tip {  
176   position: absolute;  
177   margin: auto;  
178   top: 0;  
179   left: 0;  
180   bottom: 0;  
181   right: 0;  
182   height: 24px;  
183   width: 100px;  
184   font-size: 24px;  
185   color: #000000;  
186   opacity: 0.1  
187 }  
188  
189 /* 伪翻译样式 */  
190 .translate-fake {  
191   width:100%;  
192   height:1px;  
193 }  
194  
195 /* 底部分组样式 */  
196 .foot-group {  
197   position: fixed;  
198   left: 0;  
199   bottom: 0;  
200   z-index: 40;
```

```

201     width: 100%;  

202 }  

203  

204 /* 分组样式 */  

205 .group_2 {  

206   margin-top: -76rpx;  

207 }  

208  

209 .group_3 {  

210   margin-left: 220rpx;  

211 }  

212  

213 /* 空间间隔样式 */  

214 .space-x-8 > view:not(:first-child),  

215 .space-x-8 > text:not(:first-child),  

216 .space-x-8 > image:not(:first-child) {  

217   margin-left: 16rpx;  

218 }  

219  

220 /* 图片样式 */  

221 .image_1 {  

222   width: 32rpx;  

223   height: 32rpx;  

224 }  

225  

226 /* 文本样式 */  

227 .text {  

228   color: #ffffff;  

229   font-size: 32rpx;  

230   font-family: Poppins;  

231   line-height: 29rpx;  

232   margin-right: 50rpx;  

233 }  

234  

235 /* 第二张图片样式 */  

236 .image_2 {  

237   margin-right: 230rpx;  

238 }  

239

```

4.1.5 TDD_test_cdt/

```

1  translateText: function(item, index) {  

2    let lfrom = item.lfrom || 'zh_CN'  

3    let lto = item.lto || 'en_US'  

4  

5    plugin.translate({  

6      lfrom: lfrom,  

7      lto: lto,  

8      content: item.text,  

9      tts: true,  

10     success: (resTrans)=>{  

11  

12       let passRetcode = [  

13         0, // 翻译合成成功

```

```
14         -10006, // 翻译成功, 合成失败
15         -10007, // 翻译成功, 传入了不支持的语音合成语言
16         -10008, // 翻译成功, 语音合成达到频率限制
17     ]
18
19     if(passRetcode.indexOf(resTrans.retcode) >= 0 ) {
20         let tmpDialogList = this.data.dialogList.slice(0)
21
22         if(!isNaN(index)) {
23
24             let tmpTranslate = Object.assign({}, item, {
25                 autoPlay: true, // 自动播放背景音乐
26                 translateText: resTrans.result,
27                 translateVoicePath: resTrans.filename || "",
28                 translatevoiceExpiredTime: resTrans.expired_time || 0
29             })
30
31             tmpDialogList[index] = tmpTranslate
32
33
34             this.setData({
35                 dialogList: tmpDialogList,
36                 bottomButtonDisabled: false,
37                 recording: false,
38             })
39
40             this.scrollToNew();
41
42         } else {
43             console.error("index error", resTrans, item)
44         }
45     } else {
46         console.warn("翻译失败", resTrans, item)
47     }
48
49 },
50 fail: function(resTrans) {
51     console.error("调用失败", resTrans, item)
52     this.setData({
53         bottomButtonDisabled: false,
54         recording: false,
55     })
56 },
57 complete: resTrans => {
58     this.setData({
59         recordStatus: 1,
60     })
61     wx.hideLoading()
62     }
63 }
64
65 },
```

4.1.6 TDD_test_zxk/

4.1.6.1 node_modules

Mocha 测试框架调用的npm组件

4.1.6.2 translate.js

基于TDD测试开发的获取文本翻译内容的JavaScript代码。

```
1 // translate.js
2 const axios = require('axios');
3 const md5 = require('md5');
4
5 async function translate(inputText, sourceLang, targetLang) {
6     if (!inputText) {
7         throw new Error('Invalid input');
8     }
9
10    const appid = '';
11    const secretKey = '';
12    const salt = (new Date()).getTime();
13    const text_encode = encodeURIComponent(inputText);
14    const str1 = appid + text_encode + salt + secretKey;
15    const sign = md5(str1);
16
17    const response = await axios.post('https://fanyi-
api.baidu.com/api/trans/vip/translate', {
18        q: text_encode,
19        from: sourceLang,
20        to: targetLang,
21        appid: appid,
22        salt: salt,
23        sign: sign,
24    }, {
25        headers: {
26            'Content-Type': 'application/x-www-form-urlencoded'
27        }
28    });
29
30    if (response.data.error_code) {
31        throw new Error(`Translation failed: ${response.data.error_msg}`);
32    }
33
34    return response.data.trans_result[0].dst;
35}
36
37 module.exports = translate;
```

4.1.6.3 ranslate.test.js

基于Mocha框架编写的测试文本翻译Javascript的测试程序。

```
1 // 引入测试库
2 const assert = require('assert');
```

```

3 const translate = require('./translate');
4
5 describe('translate function', function() {
6     it('should translate "software" to Chinese correctly', async function()
7     {
8         const result = await translate('software', 'en', 'zh');
9         assert.strictEqual(result, '软件');
10    });
11
12    it('should translate "software@" to Chinese correctly', async function()
13    {
14        const result = await translate('software', 'en', 'zh');
15        assert.strictEqual(result, '软件@');
16    });
17
18    it('should handle space correctly', async function() {
19        const result = await translate('hello world', 'en', 'zh');
20        assert.strictEqual(result, '你好世界');
21    });
22
23    it('should handle error correctly', async function() {
24        try {
25            await translate('', 'en', 'zh');
26        } catch (e) {
27            assert.strictEqual(e.message, 'Invalid input');
28        }
29    });
30 });

```

4.1.7 utils/

4.1.7.1 utils/api.js

这段代码在utils/目录内继承了所需要用到的api服务，在小程序内需要用到时可以直接调用，提高了代码的可读性与可维护性。

```

1 import md5 from './md5.min.js'
2
3 const appid = '' //注册百度翻译api
4 const key = '' //注册百度翻译api
5
6 function translate(q, { from = 'auto', to = 'auto' } = { from: 'auto', to:
7 'auto' }) {
8     //表示默认传递参数传递的值
9     return new Promise((resolve, reject) => {
10         let salt = Date.now() //随机数
11         let sign = md5(`$${appid}${q}${salt}${key}`) //拼接 MD5进行加密
12         wx.request({
13             url: 'https://fanyi-api.baidu.com/api/trans/vip/translate',
14             data: {
15                 q, //待翻译文本
16                 from, //待翻译的原始语言
17                 to, //待翻译成的目标语言
18                 appid,
19                 salt, //随机数
20             }
21         })
22     })
23 }

```

```

19         sign //拼接 MD5进行加密
20     },
21     success(res) {
22       if (res.data && res.data.trans_result) {
23         resolve(res.data)
24       } else {
25         reject({ status: 'error', msg: '翻译失败' })
26         wx.showToast({
27           title: '翻译失败',
28           icon: 'none',
29           duration: 3000
30         })
31       }
32     },
33     fail() {
34       reject({ status: 'error', msg: '翻译失败' })
35       wx.showToast({
36         title: '网络异常',
37         icon: 'none',
38         duration: 3000
39       })
40     }
41   })
42 }
43 }

44 function getPicBase64(tempFilePath) {
45   return new Promise(function(resolve, reject) {
46     wx.getFileSystemManager().readFile({
47       filePath: tempFilePath,
48       encoding: "base64",
49       success: function(data) {
50         console.log(data); //返回base64编码结果，但是图片的话没有
51         data:image/png
52         resolve(data);
53       }
54     })
55   })
56 }

57 function getPicToWord(src) {
58   return new Promise(function(resole, reject) {
59     wx.request({
60       url: 'https://api.jisuapi.com/generalrecognition/recognize?
61 appkey=',
62       data: {
63         pic: src,
64         type: "cnen"
65       },
66       success: function(res) {
67         console.log(res);
68         resole(res);
69       },
70       fail: function(res) {
71         console.log(res);
72         reject(res);
73       }
74     })
75   })
76 }

77 
```

```
73         }
74     })
75   })
76 }
77
78 module.exports = {
79   translate: translate,
80   getPicBase64: getPicBase64,
81   getPicToword: getPicToword
82 }
```

4.1.7.2 utils/conf.js

语音翻译的语言配置

```
1 let language = [
2   {
3     id: 0,
4     lang_name: "中文",
5     lang_content: "zh_CN",
6     lang_to: ["en_US"],
7     max_length: 300,
8     source_language: "输入文字",
9     target_language: "输出文字",
10    hold_talk: "长按说话",
11    keyboard_input: "键盘输入",
12    type_here: "输入文字",
13    bg_content: "请输入翻译内容",
14    record_failed: "录制失败",
15    recognize_nothing: "请说话",
16    time_left: "录音输入倒数",
17    text_left: "剩余文本长度",
18    prompt_time: "提示秒数",
19    upload_failed: "上传失败",
20    translating: "翻译中",
21    text_limit: "限制长度",
22    input_tip: "请输入有效文字",
23    request_failed: "请求失败",
24    delete_tip: "删除该项",
25    cancel: "取消",
26    bubble_tip: "请输入文本",
27    bg_bubble: "正在听你说话",
28    copy_source_text: "复制原文",
29    copy_target_text: "复制译文",
30    delete_item: "删除",
31    exceed_network: "网络请求失败",
32    retry_network: "尝试重新连接",
33    wait_last_record: "请等待翻译结束",
34    access_auth: "请检查权限",
35    access_network: "网络错误",
36    login: "登录",
37  },
38];
39
40 module.exports = {
```

```
41     language: language,
42 };
43
```

4.1.7.3 md5.min.js

这段代码用于获取字符串的十六进制32位下的MD5。

```
1 ! function(n) {
2     "use strict";
3
4     function t(n, t) { var r = (65535 & n) + (65535 & t); return (n >> 16) +
5         (t >> 16) + (r >> 16) << 16 | 65535 & r }
6
7     function r(n, t) { return n << t | n >>> 32 - t }
8
9     function e(n, e, o, u, c, f) { return t(r(t(t(e, n), t(u, f)), c), o) }
10
11    function o(n, t, r, o, u, c, f) { return e(t & r | ~t & o, n, t, u, c,
12        f) }
13
14    function u(n, t, r, o, u, c, f) { return e(t & o | r & ~o, n, t, u, c,
15        f) }
16
17    function c(n, t, r, o, u, c, f) { return e(t ^ r ^ o, n, t, u, c, f) }
18
19    function f(n, t, r, o, u, c, f) { return e(r ^ (t | ~o), n, t, u, c, f)
20
21    function i(n, r) {
22        n[r >> 5] |= 128 << r % 32, n[14 + (r + 64 >>> 9 << 4)] = r;
23        var e, i, a, d, h, l = 1732584193,
24            g = -271733879,
25            v = -1732584194,
26            m = 271733878;
```

```

24         for (e = 0; e < n.length; e += 16) i = 1, a = g, d = v, h = m, g =
f(g = f(g = f(g = f(g = c(g = c(g = c(g = c(g = u(g = u(g = u(g = u(g = o(g
= o(g = o(g = o(g, v = o(v, m = o(m, l = o(l, g, v, m, n[e], 7, -680876936),
g, v, n[e + 1], 12, -389564586), l, g, n[e + 2], 17, 606105819), m, l, n[e +
3], 22, -1044525330), v = o(v, m = o(m, l = o(l, g, v, m, n[e + 4], 7,
-176418897), g, v, n[e + 5], 12, 1200080426), l, g, n[e + 6], 17,
-1473231341), m, l, n[e + 7], 22, -45705983), v = o(v, m = o(m, l = o(l, g,
v, m, n[e + 8], 7, 1770035416), g, v, n[e + 9], 12, -1958414417), l, g, n[e +
10], 17, -42063), m, l, n[e + 11], 22, -1990404162), v = o(v, m = o(m, l =
o(l, g, v, m, n[e + 12], 7, 1804603682), g, v, n[e + 13], 12, -40341101), l,
g, n[e + 14], 17, -1502002290), m, l, n[e + 15], 22, 1236535329), v = u(v, m
= u(m, l = u(l, g, v, m, n[e + 1], 5, -165796510), g, v, n[e + 6], 9,
-1069501632), l, g, n[e + 11], 14, 643717713), m, l, n[e], 20, -373897302),
v = u(v, m = u(m, l = u(l, g, v, m, n[e + 5], 5, -701558691), g, v, n[e +
10], 9, 38016083), l, g, n[e + 15], 14, -660478335), m, l, n[e + 4], 20,
-405537848), v = u(v, m = u(m, l = u(l, g, v, m, n[e + 9], 5, 568446438), g,
v, n[e + 14], 9, -1019803690), l, g, n[e + 3], 14, -187363961), m, l, n[e +
8], 20, 1163531501), v = u(v, m = u(m, l = u(l, g, v, m, n[e + 13], 5,
-1444681467), g, v, n[e + 2], 9, -51403784), l, g, n[e + 7], 14,
1735328473), m, l, n[e + 12], 20, -1926607734), v = c(v, m = c(m, l = c(l,
g, v, m, n[e + 5], 4, -378558), g, v, n[e + 8], 11, -2022574463), l, g, n[e +
11], 16, 1839030562), m, l, n[e + 14], 23, -35309556), v = c(v, m = c(m, l
= c(l, g, v, m, n[e + 1], 4, -1530992060), g, v, n[e + 4], 11, 1272893353),
l, g, n[e + 7], 16, -155497632), m, l, n[e + 10], 23, -1094730640), v = c(v,
m = c(m, l = c(l, g, v, m, n[e + 13], 4, 681279174), g, v, n[e], 11,
-358537222), l, g, n[e + 3], 16, -722521979), m, l, n[e + 6], 23, 76029189),
v = c(v, m = c(m, l = c(l, g, v, m, n[e + 9], 4, -640364487), g, v, n[e +
12], 11, -421815835), l, g, n[e + 15], 16, 530742520), m, l, n[e + 2], 23,
-995338651), v = f(v, m = f(m, l = f(l, g, v, m, n[e], 6, -198630844), g, v,
n[e + 7], 10, 1126891415), l, g, n[e + 14], 15, -1416354905), m, l, n[e +
5], 21, -57434055), v = f(v, m = f(m, l = f(l, g, v, m, n[e + 12], 6,
1700485571), g, v, n[e + 3], 10, -1894986606), l, g, n[e + 10], 15,
-1051523), m, l, n[e + 1], 21, -2054922799), v = f(v, m = f(m, l = f(l, g,
v, m, n[e + 8], 6, 1873313359), g, v, n[e + 15], 10, -30611744), l, g, n[e +
6], 15, -1560198380), m, l, n[e + 13], 21, 1309151649), v = f(v, m = f(m, l
= f(l, g, v, m, n[e + 4], 6, -145523070), g, v, n[e + 11], 10, -1120210379),
l, g, n[e + 2], 15, 718787259), m, l, n[e + 9], 21, -343485551), l = t(l,
i), g = t(g, a), v = t(v, d), m = t(m, h);
        return [l, g, v, m]
    }
}

27
28     function a(n) {
29         var t, r = "",
30             e = 32 * n.length;
31         for (t = 0; t < e; t += 8) r += String.fromCharCode(n[t >> 5] >>> t
% 32 & 255);
32         return r
33     }

34
35     function d(n) { var t, r = []; for (r[(n.length >> 2) - 1] = void 0, t =
0; t < r.length; t += 1) r[t] = 0; var e = 8 * n.length; for (t = 0; t < e;
t += 8) r[t >> 5] |= (255 & n.charCodeAt(t / 8)) << t % 32; return r }
36
37     function h(n) { return a(i(d(n), 8 * n.length)) }
38
39     function l(n, t) {

```

```

40         var r, e, o = d(n),
41             u = [],
42             c = [];
43         for (u[15] = c[15] = void 0, o.length > 16 && (o = i(o, 8 *
n.length)), r = 0; r < 16; r += 1) u[r] = 909522486 ^ o[r], c[r] =
1549556828 ^ o[r];
44         return e = i(u.concat(d(t)), 512 + 8 * t.length), a(i(c.concat(e),
640))
45     }
46
47     function g(n) { var t, r, e = ""; for (r = 0; r < n.length; r += 1) t =
n.charCodeAt(r), e += "0123456789abcdef".charAt(t >>> 4 & 15) +
"0123456789abcdef".charAt(15 & t); return e }
48
49     function v(n) { return unescape(encodeURIComponent(n)) }
50
51     function m(n) { return h(v(n)) }
52
53     function p(n) { return g(m(n)) }
54
55     function s(n, t) { return l(v(n), v(t)) }
56
57     function c(n, t) { return g(s(n, t)) }
58
59     function A(n, t, r) { return t ? r ? s(t, n) : c(t, n) : r ? m(n) : p(n)
}
60     "function" == typeof define && define.amd ? define(function() { return A
}) : "object" == typeof module && module.exports ? module.exports = A :
n.md5 = A
61 }(this);

```

4.1.7.4 util.js

这段代码中定义了三个函数: formatTime, recordTime, 和formatNumber。

- formatTime函数接收一个date对象作为参数，获取date对象的年、月、日、时、分、秒，并将它们格式化为字符串。
- recordTime只获取date对象的月、日、时、分，并将它们格式化为字符串。
- formatNumber函数接收一个数值n作为参数，将其转化为字符串并检查是否需要在前面补0。
- 通过module.exports将formatTime和recordTime两个函数导出，使得它们可以在其他文件中被引用。

```

1 const formatTime = date => {
2     const year = date.getFullYear();
3     const month = date.getMonth() + 1;
4     const day = date.getDate();
5     const hour = date.getHours();
6     const minute = date.getMinutes();
7     const second = date.getSeconds();
8
9     return [year, month, day].map(formatNumber).join('/') + ' ' + [hour,
minute, second].map(formatNumber).join(':');
10 }
11

```

```

12 function recordTime(date) {
13     var month = date.getMonth() + 1;
14     var day = date.getDate();
15     var hour = date.getHours();
16     var minute = date.getMinutes();
17
18     return [month, day].map(formatNumber).join('/') + ' ' + [hour,
19     minute].map(formatNumber).join(':');
20 }
21
22 const formatNumber = n => {
23     n = n.toString();
24     return n[1] ? n : '0' + n;
25 }
26
27 module.exports = {
28     formatTime: formatTime,
29     recordTime
}

```

4.1.8 ./

4.8.8.1 ./app.js

在全局的 App 对象中，`onLaunch` 函数定义了当小程序启动时的操作，例如从本地获取当前语言和历史记录。`getRecordAuth` 函数则用于获取录音权限。`onHide` 函数定义了当小程序隐藏时的操作，例如停止后台音频。`globalData` 对象则定义了全局的数据，包括历史记录、当前语言、按钮列表、图片的 Base64 编码、词语列表和语言列表等。

在语言列表 `langList` 中，每个元素都是一个对象，包含了语言的中文名（`chs`）、语言代码（`lang`）、索引（`index`）和图标源（`src`，对于部分语言）。其中，语言代码是用于识别和设置语言的标识，索引是语言在列表中的位置，图标源是显示语言图标的 URL 地址。每一个对象代表一种语言。在这个应用中，`langList` 定义了全局支持的语言列表，包括英语、中文、日语等。每种语言的定义都包括中文名（`chs`）、语言代码（`lang`）和语言在列表中的索引（`index`）。部分语言定义还包括了图标的 URL 地址（`src`），用于在界面上显示对应的图标。

```

1 // 导入工具模块
2 const utils = require('./utils/util.js')
3
4 App({
5     onLaunch: function () {
6         // 展示本地存储能力
7         // 在全局数据中设置当前语言，如果本地没有存储过当前语言，那么就使用语言列表的第一个语
8         // 言
9         this.globalData.curLang = wx.getStorageSync('curLang') ||
10        this.globalData.langList[0];
11        this.globalData.fromLang = wx.getStorageSync('fromLang') ||
12        this.globalData.langList[1];
13
14        // 从本地获取历史记录，如果获取失败，则将全局的历史记录设置为空数组
15        wx.getStorage({
16            key: 'history',
17            success: (res) => {
18                this.globalData.history = res.data
19            }
20        })
21    }
})

```

```
16      },
17      fail: (res) => {
18          console.log("get storage failed")
19          console.log(res)
20          this.globalData.history = []
21      }
22  })
23 },
24 // 权限询问
25 // 获取录音权限
26 getRecordAuth: function () {
27     wx.getSetting({
28         success(res) {
29             console.log("succ")
30             console.log(res)
31             if (!res.authSetting['scope.record']) {
32                 wx.authorize({
33                     scope: 'scope.record',
34                     success() {
35                         // 用户已经同意小程序使用录音功能
36                         console.log("succ auth")
37                     },
38                     fail() {
39                         console.log("fail auth")
40                     }
41                 })
42             } else {
43                 console.log("record has been authed")
44             }
45         },
46         fail(res) {
47             console.log("fail")
48             console.log(res)
49         }
50     })
51 },
52
53 onHide: function () {
54     // 当小程序隐藏时，停止后台音频
55     wx.stopBackgroundAudio()
56 },
57
58 // 定义全局的数据
59 globalData: {
60     history: [],
61     curLang: {},
62     formLang:{},
63     buttons:[],
64     picBase64: "",
65     word: [],
66     // langList定义了支持的语言列表
67     langList: [
68         'chs': '英文',    // 语言的中文名
69         "lang": 'en',    // 语言的代码
70         "index": 0,      // 语言在列表中的索引
71         // 语言的图标URL，如果没有指定，那么将不显示图标
```

```
72     "src": "https://codefun-proj-user-res-1256085488.cos.ap-
73     guangzhou.myqcloud.com/644bb0005a7e3f03102917b5/644bb06fb98f5d0011665f39/11
74     17504288074e5a51c1cc92bf0eedb.png"
75     },
76     {
77         'chs': '中文',
78         'lang': 'zh',
79         "index": 1,
80         "src": "https://codefun-proj-user-res-1256085488.cos.ap-
81     guangzhou.myqcloud.com/644bb0005a7e3f03102917b5/644bb06fb98f5d0011665f39/ac
82     7f93cc16a6602c18750922cf92014c.png"
83     },
84     {
85         'chs': '日语',
86         'lang': 'jp',
87         "index": 2
88     },
89     {
90         'chs': '韩语',
91         'lang': 'kor',
92         "index": 3,
93         "src": "https://codefun-proj-user-res-1256085488.cos.ap-
94     guangzhou.myqcloud.com/644bb0005a7e3f03102917b5/644bb06fb98f5d0011665f39/16
95     826823185920520575.png"
96     },
97     {
98         'chs': '法语',
99         'lang': 'fra',
100        "index": 4
101    },
102    {
103        'chs': '德语',
104        'lang': 'de',
105        "index": 5,
106        "src": "https://codefun-proj-user-res-1256085488.cos.ap-
107     guangzhou.myqcloud.com/644bb0005a7e3f03102917b5/644bb06fb98f5d0011665f39/16
108     826823185577922182.png"
109    },
110    {
111        'chs': '俄语',
112        'lang': 'ru',
113        "index": 6
114    },
115    {
116        'chs': '泰语',
117        'lang': 'th',
118        "index": 7
119    },
119    {
120        'chs': '西班牙语',
121        'lang': 'spa',
122        "index": 8
123    },
124    {
125        'chs': '阿拉伯语',
126        'lang': 'ara',
127        "index": 9
128    }
```

```
120     "index": 9
121   },
122   {
123     'chs': '意大利语',
124     'lang': 'it',
125     "index": 10
126   },
127   {
128     'chs': '葡萄牙语',
129     'lang': 'pt',
130     "index": 11
131   }
132 ]
133 }
134 })
135
```

4.8.8.2 ./app.json

这段代码是微信小程序的配置文件 `app.json` 中的一部分。`app.json` 是小程序的全局配置，包括了小程序的所有页面路径、界面表现、网络超时时间、多 tab 等等。简单解释一下，这段代码配置了小程序的页面、窗口表现、使用的插件、sitemap 文件位置以及底部 tab 栏的表现。其中，`pages` 数组包含了所有页面的路径，`window` 对象设置了全局窗口的外观，`plugins` 对象声明了小程序使用的插件，`sitemapLocation` 字符串指明了sitemap 文件的位置，`tabBar` 对象则描述了底部 tab 栏的表现。

```
1  {
2    // 小程序所有页面的路径数组
3    "pages": [
4      "pages/index/index",
5      "pages/history/history",
6      "pages/OCR/OCR",
7      "pages/getPic/getPic",
8      "pages/voice_translation/voice_translation",
9      "pages/edit/edit",
10     "pages/choose_language/choose_language"
11   ],
12
13   // 全局的默认窗口表现
14   "window": {
15     "backgroundTextStyle": "light", // 下拉背景字体、Loading 图的样式
16     "navigationBarBackgroundColor": "#1493FC", // 导航栏背景颜色
17     "navigationBarTitleText": "TransWe", // 导航栏标题文字内容
18     "navigationBarTextStyle": "black", // 导航栏标题颜色
19     "backgroundColor": "#4b3c96", // 窗口的背景颜色
20     "navigationStyle": "custom" // 导航栏样式
21   },
22
23   // 插件的声明
24   "plugins": {
25     "wechatsI": {
26       "version": "0.3.3", // 插件的版本
27       "provider": "wx069ba97219f66d99" // 插件的提供者
28     }
29   },
30 }
```

```

30
31     "sitemapLocation": "sitemap.json", // 小程序的站点地图文件位置
32
33     // 底部 tab 栏的表现
34     "tabBar": {
35         "borderStyle": "white", // tab 的边框颜色
36         "position": "bottom", // tab 的位置
37         "color": "#bfbfbf", // tab 的默认颜色
38         "selectedColor": "#1c1b21", // tab 被选中时的颜色
39         "list": [ // tab 的列表, 顺序与显示的顺序一致
40             {
41                 "pagePath": "pages/index/index",
42                 "text": "翻译",
43                 "iconPath": "imgs/icon-1.png", // 未选中时的图标路径
44                 "selectedIconPath": "imgssel-icon-1.png" // 选中时的图标路径
45             },
46             {
47                 "pagePath": "pages/getPic/getPic",
48                 "text": "拍照翻译",
49                 "iconPath": "imgs/paizhao-xianxing.png",
50                 "selectedIconPath": "imgs/paizhao.png"
51             },
52             {
53                 "pagePath": "pages/voice_translation/voice_translation",
54                 "text": "语音翻译",
55                 "iconPath": "imgs/maikefeng-xianxing.png",
56                 "selectedIconPath": "imgs/maikefeng.png"
57             },
58             {
59                 "pagePath": "pages/history/history",
60                 "text": "历史",
61                 "iconPath": "imgs/icon-2.png",
62                 "selectedIconPath": "imgssel-icon-2.png"
63             }
64         ]
65     }
66 }
67

```

4.8.8.3 ./app.wxss

这段代码是微信小程序的全局样式表 `app.wxss` 中的一部分。`.wxss` 是微信小程序的样式语言，类似于 Web 的 CSS，可以设置小程序中的组件样式。这里定义了全局的样式规则，将在整个小程序范围内生效。

这段代码的主要作用是：

- 定义了全局容器的样式，例如背景色、字体颜色和大小、布局方式等。
- 定义了版权信息的样式，例如字体颜色和大小、在容器中的位置等。
- 定义了视图在悬停时的背景颜色。

```

1  /* 引入 iconfont 的样式文件 */
2  @import "./assets/iconfont/iconfont.wxss";
3
4  /* 容器样式 */

```

```

5 .container {
6   padding: 0; /* 容器内边距设置为 0 */
7   background-color:#f7f8f9; /* 容器背景色 */
8   height: 100vh; /* 容器高度设置为视口高度的100% */
9   display: flex; /* 设置为弹性布局 */
10  flex-direction:column; /* 设置主轴方向为垂直方向 */
11  box-sizing: border-box; /* 盒模型设置为 border-box, 即元素的 padding 和
12  border 在元素宽高内 */
13  font-size: 30rpx; /* 字体大小设置为 30rpx */
14  color: #333; /* 字体颜色 */
15 }
16 /* 版权信息样式 */
17 .copyright {
18   align-self: center; /* 版权信息在交叉轴方向居中对齐 */
19   flex: 1; /* flex 值为 1, 使版权信息元素占据剩余的空间 */
20   display: flex; /* 设置为弹性布局 */
21   align-items: flex-end; /* 在交叉轴方向上, 版权信息位于容器的底部 */
22   padding-bottom: 20rpx; /* 底部内边距设置为 20rpx */
23   font-size: 28rpx; /* 字体大小设置为 28rpx */
24   color:#999; /* 字体颜色 */
25 }
26 /* 视图悬停样式 */
27 .view-hover {
28   background-color: #f3f3f3!important; /* 视图的背景色设置为#f3f3f3, !important 表示优先级最高, 如果有其他样式影响, 该规则将覆盖其他规则 */
29 }
30 }
31 
```

4.8.8.4 ./project.config.json

这段代码是微信小程序的项目配置文件 `project.config.json` 的一部分。`project.config.json` 是微信小程序的项目配置文件，用于配置项目的相关信息，包括项目名，`appid`，编译设置等。

这个文件对于小程序项目的运行具有重要影响。例如，`appid` 字段设置了小程序的唯一标识，`es6` 字段决定了是否需要将 ES6 代码转为 ES5 代码以适应更多环境，`minified` 字段决定了是否需要压缩代码等。

```

1 {
2   "description": "项目配置文件。", // 项目描述
3   "packOptions": { // 打包配置项
4     "ignore": [], // 打包时需要忽略的文件列表
5     "include": [] // 打包时需要包含的文件列表
6   },
7   "setting": { // 设置项
8     "urlCheck": false, // 是否开启url合法性检查
9     "es6": true, // 是否启用ES6转ES5功能
10    "postcss": true, // 是否启用PostCSS
11    "minified": true, // 是否压缩代码
12    "ignoreDevUnusedFiles": false, // 是否忽略开发模式下未引用的资源文件
13    "ignoreUploadUnusedFiles": false, // 是否忽略上传模式下未引用的资源文件
14    "newFeature": true, // 是否启用新的编译功能
15    "babelSetting": { // Babel 编译配置
16      "ignore": [] // 需要忽略编译的文件列表
17    }
18  }
19 }
```

```

17     "disablePlugins": [], // 需要禁用的插件列表
18     "outputPath": "" // 编译输出路径
19   },
20   "condition": false, // 条件编译
21   "skylineRenderEnable": false // 是否开启3D渲染模式
22 },
23   "compileType": "miniprogram", // 编译类型为小程序
24   "libVersion": "2.32.0", // 使用的基础库版本
25   "appid": "wx365790a8c04e3a3b", // 小程序的appid
26   "projectname": "TransWe", // 项目名称
27   "simulatorType": "wechat", // 模拟器类型为微信
28   "simulatorPluginLibVersion": {}, // 插件库版本（用于模拟器）
29   "condition": {}, // 调试配置
30   "editorSetting": { // 编辑器设置
31     "tabIndent": "insertSpaces", // 设置缩进为插入空格
32     "tabsize": 2 // 设置缩进大小为2个空格
33   }
34 }
35

```

4.8.8.5 ./project.private.config.json

这段代码是微信小程序的 `project.private.config.json` 文件的一部分。这个文件是一个私有配置文件，用于覆盖 `project.config.json` 中的同名字段，存储开发者在开发工具中改动的项目配置。当 `project.private.config.json` 与 `project.config.json` 中的字段冲突时，`project.private.config.json` 的配置项会优先被应用。

这个配置文件有助于个别开发者或者团队在本地进行特定的设置，比如禁止URL检查或者开启热重载等，并且不会影响到其他的开发人员。

```

1  {
2   // 项目描述，指出这是一个私有配置文件
3   "description": "项目私有配置文件。此文件中的内容将覆盖 project.config.json 中的相
同字段。项目的改动优先同步到此文件中。详见文档：
4   // https://developers.weixin.qq.com/miniprogram/dev/devtools/projectconfig.html
5   // 项目名称
6   "projectname": "TransWe",
7   // 设置项
8   "setting": {
9     // 是否启用编译热更新
10    "compileHotReload": true,
11    // 是否开启URL合法性检查
12    "urlCheck": false
13  },
14  // 调试配置
15  "condition": {}
16

```

4.2 系统测试

4.2.1 TDD测试

4.2.1.1 TDD_test_cdt/

语音翻译测试

1. 测试用例（中翻英）

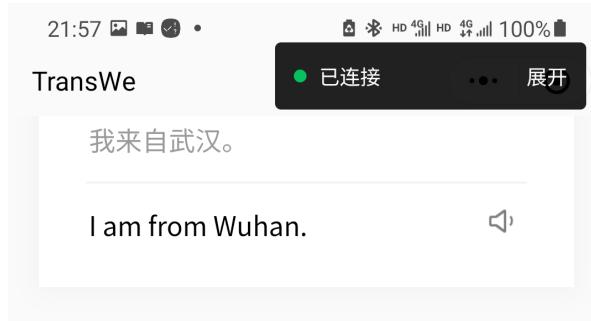
输入	期望输出
华中科技大学	Huazhong University of Science and Technology
我来自武汉	I am from Wuhan
如何进入校园	How to enter the campus
用户的要求是绝对的	User requirements are absolute

2. 测试结果

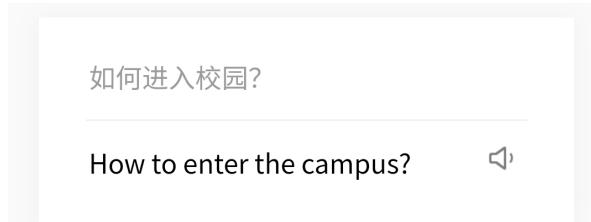
- 测试点1



- 测试点2



- 测试点3



- 测试点4

用户的要求是绝对的。

User requirements are
absolute.



测试通过数 (4/4)

4.2.1.2 TDD_test_zxk/

基于Mocha框架编写的测试文本翻译Javascript的测试程序。

```
1 // 引入测试库
2 const assert = require('assert');
3 const translate = require('./translate');
4
5 describe('translate function', function() {
6     it('should translate "software" to Chinese correctly', async function()
7     {
8         const result = await translate('software', 'en', 'zh');
9         assert.strictEqual(result, '软件');
10    });
11
12     it('should translate "software@" to Chinese correctly', async function()
13     {
14         const result = await translate('software', 'en', 'zh');
15         assert.strictEqual(result, '软件@');
16    });
17
18     it('should handle space correctly', async function() {
19         const result = await translate('hello world', 'en', 'zh');
20         assert.strictEqual(result, '你好世界');
21    });
22
23     it('should handle error correctly', async function() {
24         try {
25             await translate('', 'en', 'zh');
26         } catch (e) {
27             assert.strictEqual(e.message, 'Invalid input');
28         }
29    });
30});
```

若要进行测试，请在正确安装相关框架的情况下运行：

```
1 | mocha .\translate.test.js
```

测试成功会显示下列截图：

● PS D:\Documents\my documents\folder\study\软件工程\TransWe\code\TDD test_zxk> mocha .\translate.test.js

```
translate function
  ✓ should translate "software" to Chinese correctly (361ms)
  ✓ should translate "software@" to Chinese correctly
  ✓ should handle space correctly
  ✓ should handle error correctly
```

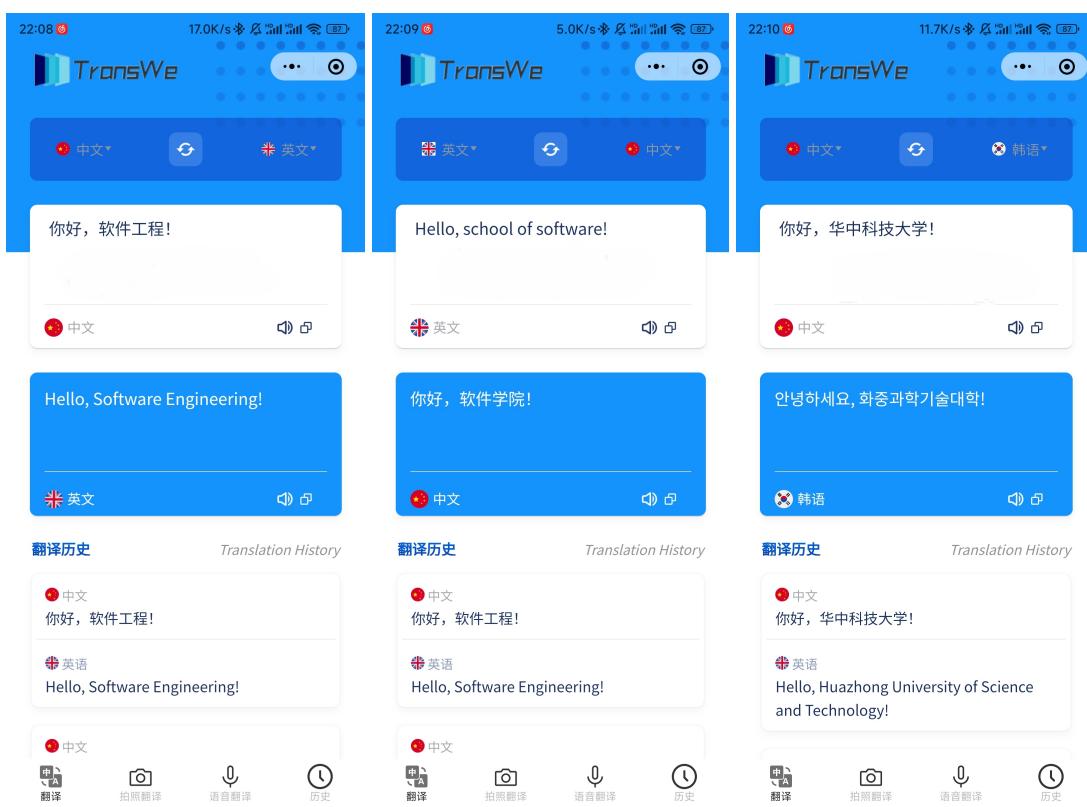
4 passing (367ms)

5. 系统界面展示

5.1 index | 主页

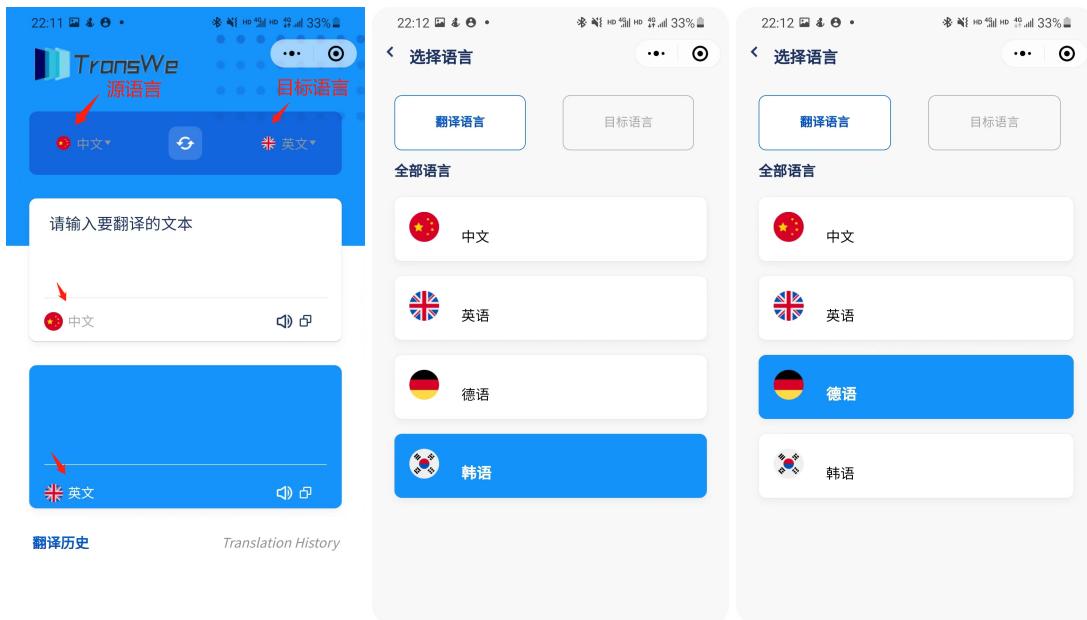
在主页中：

- 用户在输入框中输入待翻译的文本，**程序会自动检测输入语言**，用户在下拉菜单中选择相应的目标语言。
- 用户也可以**手动选择输入语言**。输入完成后，程序将进行翻译并在输出框中显示翻译结果。
- 用户可以点击小喇叭按钮，调用语音合成功能，听到翻译结果。
- 用户可以点击剪贴板按钮，待翻译文本或翻译结果会自动复制到用户的剪贴板。
- 用户可以点击下方的翻译历史板块，跳转到翻译历史页面。
- 用户可以点击最下方的导航栏跳转到对应的界面。

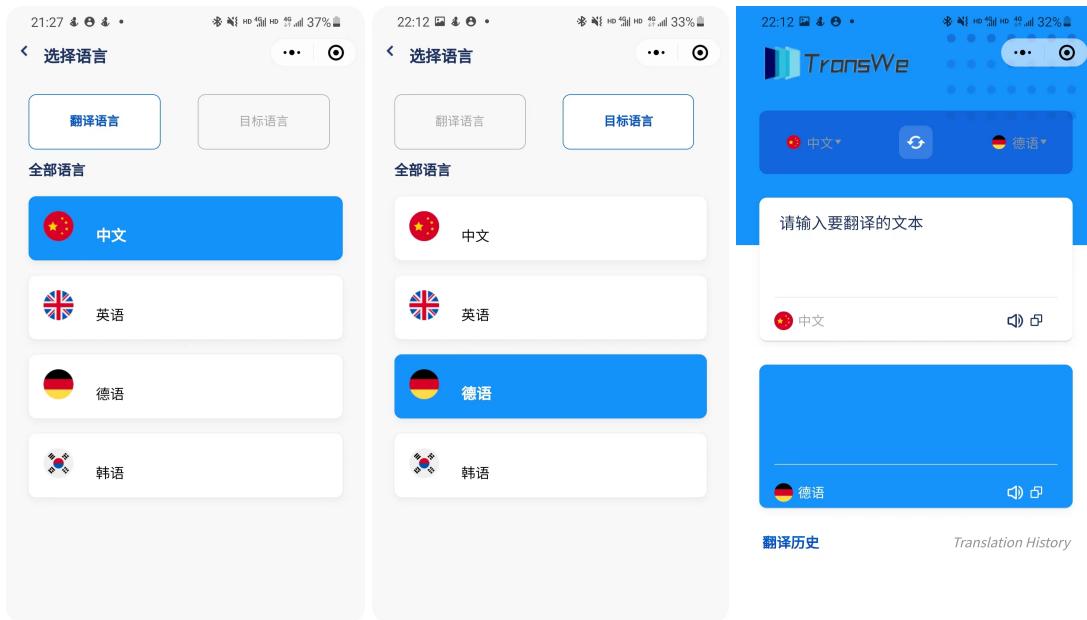


5.2 choose_language | 选择语言界面

在选择语言界面选择翻译语言和目标语言

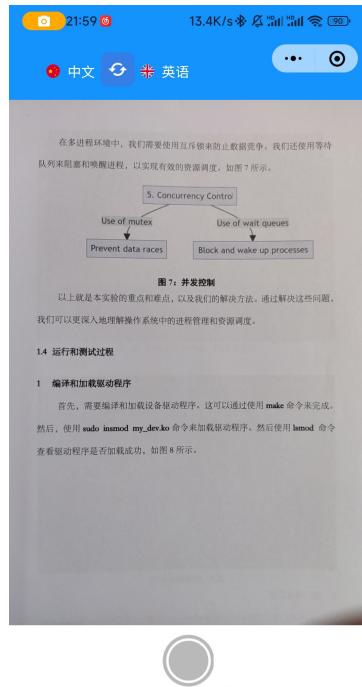


可以看到，翻译页面根据语言选择进行了及时的更新



5.3 getPic | 拍照界面

在这个界面，用户点击拍照按钮可以进行拍照，并进入拍照翻译的结果页面。



翻译
拍照翻译
语音翻译
历史

5.4 OCR | 拍照翻译结果

在这个界面，点击翻译图片，小程序会进行OCR识别并进行翻译，将结果显示在屏幕上。



中文字幕

在多进程环境中，我们需要使用互斥锁来防止数据竞争。我们还使用等待队列来阻塞和唤醒进程，以实现有效的资源调度。如图7所示。

As shown in Figure 7, Full/empty, 5. Concurrency Control, es, Use of mutex, Use of wait queues, Prevent data races, Block and wake up processes, ,互, 实, 图7: 并发控制, ,以上就是本实验的重点和难点, 以及我们的解决方法。通过解决这些问题, ,我们可以更深入地理解操作系统中的进程管理和资源调度。,1.4运行和测试过程,ions,1,编译和加载驱动程序,首先, 需要编译和加载设备驱动程序。这可以通过使用make命令来完成。,然后, 使用sudo insmod my_dew.ko命令来加载驱动程序。然后使用lsmod命令,查看驱动程序是否加载成功, 如图8所示。,用

英文字幕

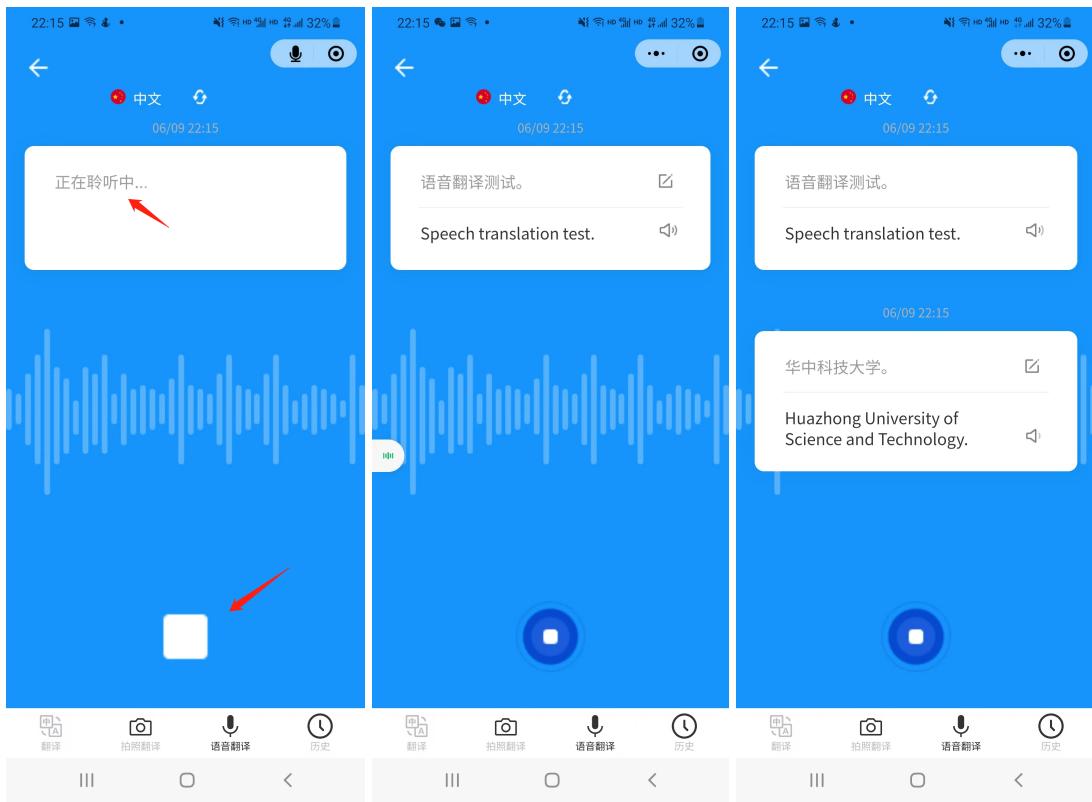
In a multi process environment, we need to use Mutual exclusion to prevent data contention. We also use waiting and queues to block and wake up processes for effective resource scheduling.

As shown in Figure 7., Full/empty, 5. Concurrency Control, es, Use of mutex, Use of wait queues, Prevent data races, Block and wake up processes mutual real Figure 7:

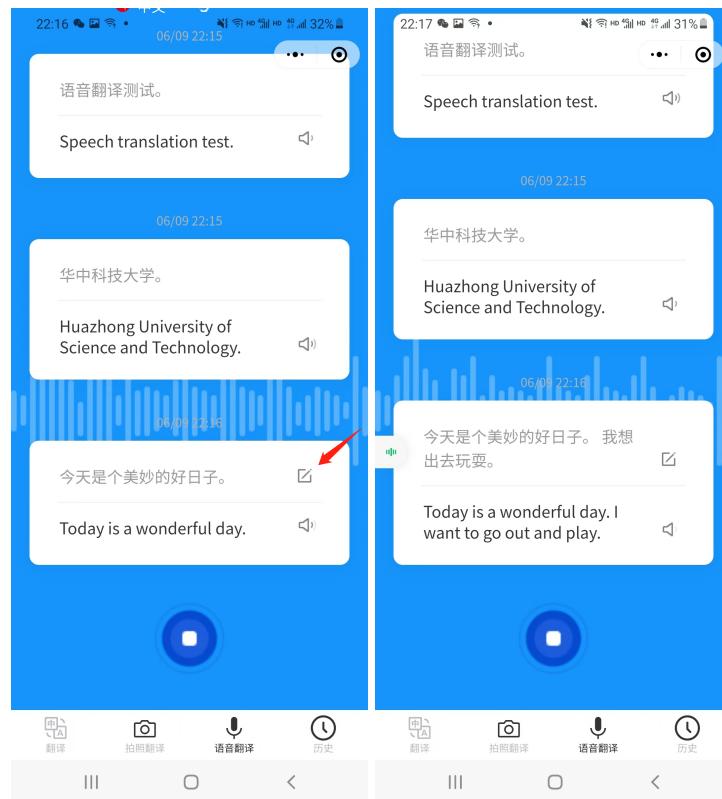
5.5 voice_translation | 语音翻译页面

语音翻译页面，点击切换按钮改变录音语言。

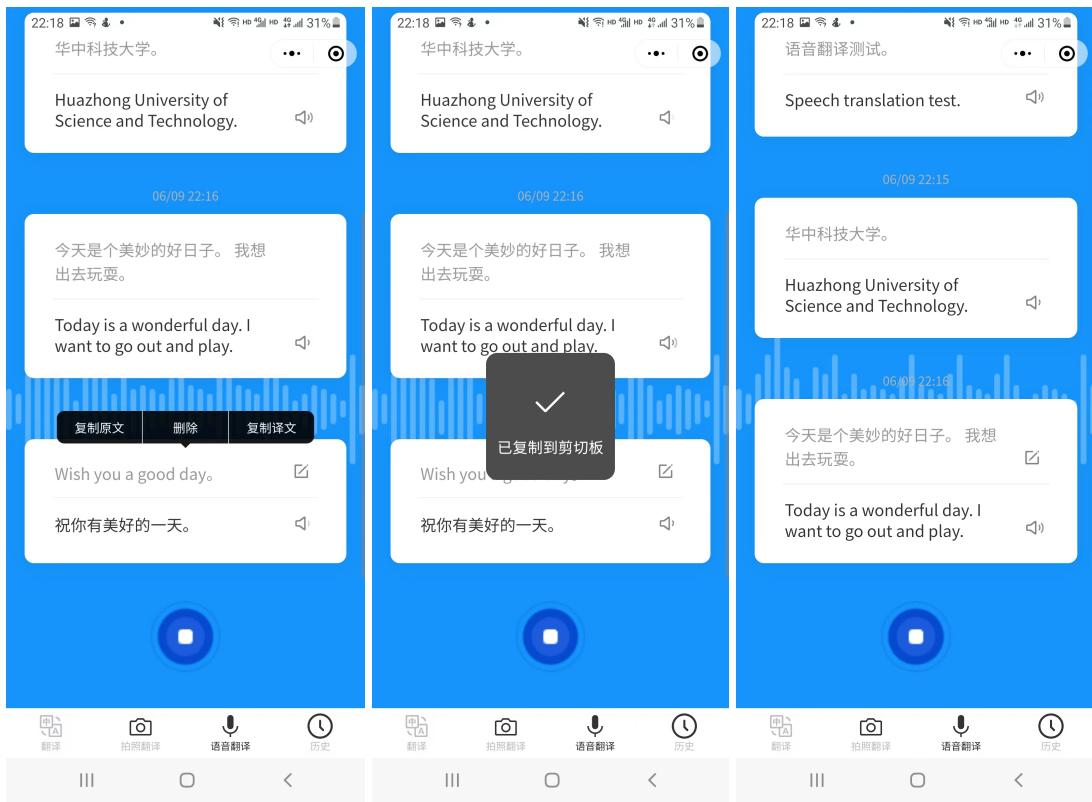
长按录音按钮，按钮样式改变，出现文字提示正在录音，翻译的结果会以卡片的形式保存在本地。



在卡片右侧对应两个组件，分别代表编辑文本和语音合成。

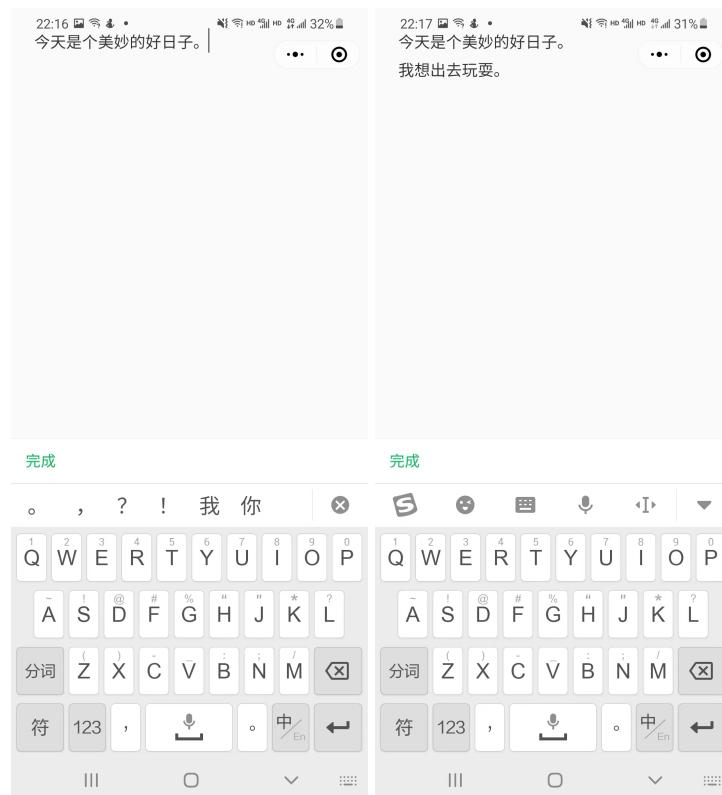


长按卡片出现弹窗，功能包括复制文本以及删除不用的卡片。

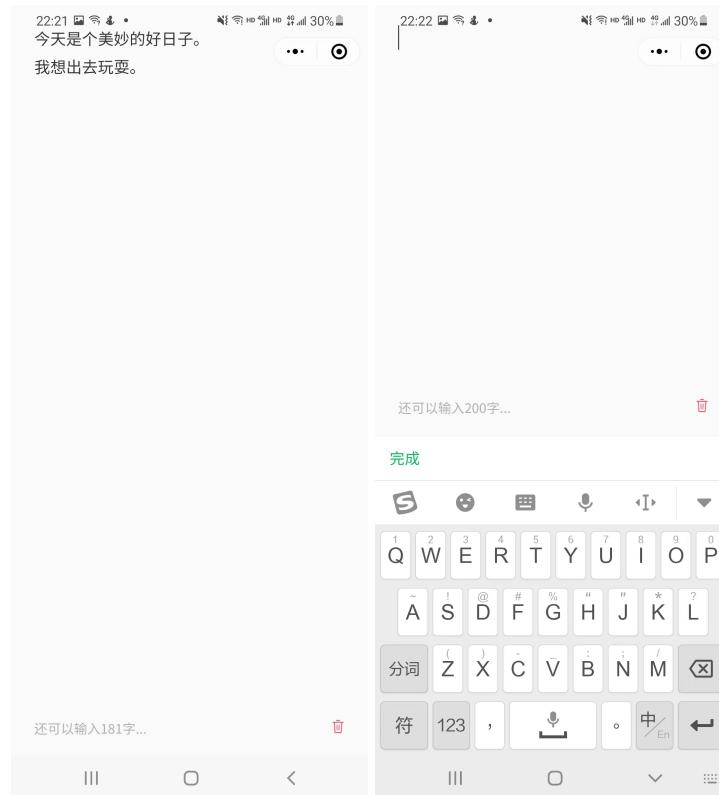


5.6 edit | 文本编辑界面

考虑到用户录音时可能会因为录音失误导致录入的文本有误，为避免用户重新录音的麻烦，允许编辑录音文本，为用户带来更好的体验。

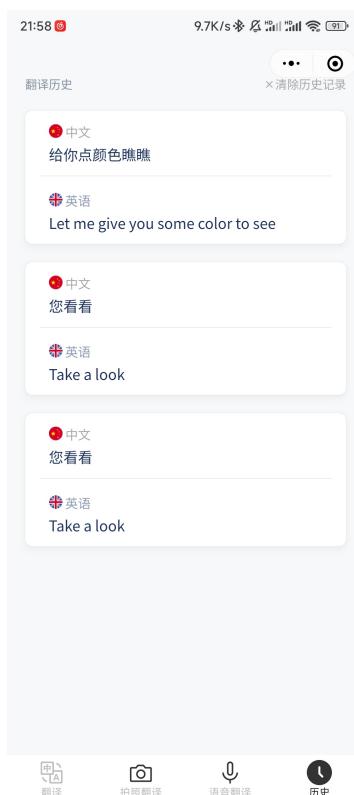


设置了最大输入文本限制，用户可以看到剩余可输入文字，点击清空按钮可以快速清除文字



5.7 history | 翻译历史

在这个界面，用户可以查看使用小程序的翻译历史，点击对应的翻译历史可以跳转到主页查看翻译结果。



6. 总结

在这个课程设计项目中，我们的团队开发了一个名为TransWe的微信小程序。这是一个强大的机器翻译工具，它的核心功能是能够快速准确地翻译各种语言。但是，我们并没有止步于此，我们还集成了第三方OCR、语音识别和语音合成服务，这些功能的加入使得TransWe不仅仅是一个翻译工具，更是一个全方位的语言服务平台，为用户提供了更便捷、高效的翻译服务。

在这个过程中，我们深入了解了微信小程序的开发流程和特性。微信小程序的特性如页面跳转、图片上传、音频播放等，都被我们充分利用，以实现TransWe的各项功能。同时，我们也学习了如何利用第三方服务来增强小程序的功能性和用户体验。例如，我们利用OCR服务实现了图片中文本的识别，利用语音识别和语音合成服务实现了语音翻译功能，这些都大大提高了TransWe的用户体验。

在代码实现方面，我们遵循了良好的编程习惯。我们的代码结构清晰，每个函数、每个模块都有其明确的职责；我们的命名规范，变量名、函数名都能准确地反映其功能；我们的注释详细，每一段重要的代码都有相应的注释，方便后续的维护和修改。这些都是我们在软件工程课程中学到的重要知识，也是我们在实际开发过程中得到应用的地方。

然而，这个项目的开发过程并非一帆风顺。我们遇到了一些挑战，如API调用的问题、图片大小限制的问题等。但是，我们并没有因此而退缩，我们通过查阅文档、搜索解决方案、反复测试等方式，最终都成功地解决了这些问题。这个过程不仅锻炼了我们的问题解决能力，也让我们更加深入地理解了软件开发的实际过程。

总的来说，这个课程设计项目是一次非常宝贵实践经验。它不仅提升了我们的编程技能，也锻炼了我们的问题解决能力。同时，看到自己的作品能够真正地帮助到用户，也是一种非常满足的感觉。我们深感，软件开发不仅仅是编写代码，更是解决问题，满足用户需求的过程。**我们将带着这次的经验和教训，继续在软件工程的道路上探索和前进。**