華中科技大學

《软件工程》项目报告

题目： 学途神器小程序

课程名称： 软件工程

专业班级： **CS2107、CS2110**

组　　名： 实验做不队

同组成员： 学号：**U20211xxxx**

姓名：**cxk**

学号：**U20211xxxx**

姓名：**yy**

学号：**U20211xxxx**

姓名：**csr**

学号：**U20211xxxx**

姓名：**cxy**

指导教师： 瞿彬彬

报告日期： **2023 年 12 月 2 日**

计算机科学与技术学院

任 务 书

一 总体要求

1. 综合运用软件工程的思想，协同完成一个软件项目的开发，掌软件工程相关的技术和方法；
2. 组成小组进行选题，通过调研完成项目的需求分析，并详细说明小组成员的分工、项目的时间管理等方面。
3. 根据需求分析进行总体设计、详细设计、编码与测试等。

二 基本内容

根据给出的题目任选一题，自行组队，设计与开发中软件过程必须包括：

1. **问题概述、需求分析：**正确使用相关工具和方法说明所开发软件的问题定义和需求分析，比如 NABCD 模型，Microsoft Visio，StarUML 等工具 (20%)；
2. **原型系统设计、概要设计、详细设计：**主要说明所开发软件的架构、数据结构及主要算法设计，比如墨刀等工具（35%）；
3. **编码与测试：**编码规范，运用码云等平台进行版本管理，设计测试计划和测试用例（30%）；
4. **功能创新：**与众不同、特别吸引用户的创新（10%）；
5. **用户反馈：**包括用户的使用记录，照片，视频等（5%）。

目　录

1 问题定义

1.1 项目背景与意义

1.1.1 项目需求 Need

1、项目来源：儿童学习

小明的妈妈近期很郁闷，小明上小学后，老师要求每天完成 30 道口算题，每天出题改题对小明妈妈成了一个大负担，“有个可以出题的软件就好了”小明妈妈这么想。并且小明现在处于一年级，小明妈妈希望买一个软件可以用三年。

目前，许多家长都面临上述的问题：老师所布置的作业往往需要孩子和家长共同完成，然而许多家长由于工作繁忙或者专业知识有限并不能好好的完成该项任务。因此，孩子需要一个可以提供题目和批改功能的东西来帮助自己完成老师布置的作业，对于广大家长而言，迫切需要一个可以提供题库和自动批改题目的小程序来帮助自己更高效更准确的完成任务。

2、社会需求分析

当下社会，从小学到高中，乃至大学和工作的人群，都面临“做题”的处境。小学生、中学生需要刷题，为了更好的考试成绩；大学生需要刷题，为了更好的检验自己的学习成果，并提高学习效率；工作阶段也需要刷题，为了更好的通过面试或笔试。

首先，当下人们拥有的时间更多是“碎片化”的，以往通过纸质课本、辅导书来刷题的形式渐渐不能符合人们的需求，人们很难找到一个固定的或者较长的时间坐在“自习室”中刷题，因此需要一个可以随时随地拿出来做一道题的工具来帮助自己实现知识的测试与巩固。

其次，纸质版的练习册很多都有一个共同的问题，题目作答一次就无法重复练习，如果学生想要重新再做一次试题就会看到自己之前作答的答案，导致无法重新练习，加深记忆，而考试题库 APP 就解决了这个问题，试题可以随时随地重新练习，让学生温习知识更简单。

同时，刷题类 app 一般都拥有实时解答功能，而传统练习册，大部分都将答案统一放在最后几页，翻来覆去对答案使得做题者更难集中注意力。如将答案撕

下，又可能造成丢失。相对而言，app 让对答案变得更加方便。

1.1.2 实现方法 Approach

发布途径上，团队计划用微信小程序的形式向用户开放，使用户可以通过扫描二维码或微信搜索的形式查询到该小程序。

软件的展现形式方面，采用类似 MOOC 答题和批改的形式展现题目给用户，且提供错题集、收藏等功能，方便用户随时随地查找自己想看的题目。此外，我们借鉴了许多市面上已有的学习类 app 的思路，开发 AI 问答、智能翻译、文本时被等功能，力图在刷题之外，提供给用户更全面的功能。

在开发途径方面，我们将采用微信官方的小程序开发工具进行实现，利用 wxml、wxss 对小程序界面进行多层渲染，利用 json 对界面进行整体配置，利用 js 实现界面功能，使用开发工具自带的云端数据库实现题集与用户信息的存储，使用腾讯云的轻量化服务器，搭建接口以实现学习神器的三样功能，同时，我们重写了部分微信小程序开发工具自带的许多函数，以便更好地实现某些功能。

1.1.3 项目益处 Benefit

针对广大为了孩子地教育而劳累不断的家长，小程序为他们的孩子提供更有针对性地题目、更加准确的答案以及更加高效的批改效率。家长可以更轻松地检验孩子的学习情况，查看孩子收藏的题目与错题集，更有针对性地帮助孩子巩固学习，分析并解决孩子目前所遇到的问题，并进一步为孩子制定培养计划。小程序的学习报告功能（将涵盖学习时间、做题数量、正确率等要素），能让家长更好的判断孩子的学习状态。

针对需要刷题的用户，他们可以随时随地打开小程序进行刷题，同时我们的小程序提供丰富的题库，并能保存正在刷的题库，这样每次开始刷题时都能接着自己上一次的记录继续进行，同时也有更加丰富的题目选择权。收藏与错题集功能，可以让用户更加清楚自己的痛点，复习更方便，更有针对性，同时也更高效。

1.1.4 竞争对手 Competitors

市面上有众多类似功能的 app，刷题类如“刷题神器”、“我爱刷题”等，辅助学习类如“作业帮”、“小猿搜题”等，此外还有专门针对某一学科的 app，如“不背单词”、“百词斩”等。以上均属于我们产品的竞争对手。

相比竞争对手，我们具有以下几点优势：

1、简洁性：我们将用户所需要选择的题库入口、正在刷题的入口、直接放置在首页，收藏夹/错题集入口放置在“我的”界面，防止因功能过多或太注重分区而使界面变得复杂错乱，使用户找不到自己需要的东西。

2、错题集：用户做题时，提交答案系统便会自动批改，而与正确答案不符的题目将自动加入错题集，错题集将按照科目分类，用户可以随时随地查看错题集并进行重做、删除等操作。

3、更加方便的进行刷题：我们设置的“继续刷题”窗口，将自动从用户所选题库中，未曾刷过的题目中出题。该功能是出于用户再一段时间内想要专注刷一本题集的考虑，同时也避免了用户在正常刷题过程中重复刷题。

4、学习神器：小程序包含了智能翻译、文本识别、AI 问答三样功能，放置在“学习神器”页面下，智能翻译支持包含英语、日语、法语等近 20 种语言的互译；文本识别能够支持 5mb 大小以内的图片的文本识别；AI 问答支持 GBT3.5Turbo-0613-16k 模型，实现用户与 AI 在线交互，能够回答绝大多数学习问题。

5、健壮性：小程序在面对各种特殊情况时，都会有相应的提示，如未选择题库不能进入“继续刷题”、未登录不能进入“收藏夹”和“错题集”、“收藏夹”与“错题集”无内容时会有相应提示等。

6、适应性：小程序的各类窗口、界面显示等会根据文字量/机型等做出相应的调整，不会出现很奇怪的显示情况。

1.1.5 推广途径 Delivery

1、官方发布

在官方平台上线（微信小程序），生成二维码方便用户扫描使用，微信搜索框搜索产品名称（学途神器）即可看到我们的小程序。

2、媒体平台推广

通过 QQ、微信、B 站、小红书等平台对小程序进行推广，在 B 站、小红书等平台推广时，打上对应的标签，方便推送给更合适的人群。

3、用户反馈

在小程序中增加用户反馈功能，采用一定的奖励机制让用户对小程序进行评价和推广，达成一传十十传百的效果。

4、其他方式

如利用小程序做出有意思的动画、图片等等，这种形式相对而言传播更加容易且迅速，会在短时间内积累大量用户，但是需要注意网络舆论的风险。

1.2 项目基本目标

“学途神器”作为一款针对小学生的刷题小程序，意在使众多小学生能够自由地刷题、使众多家长与小学老师能够出题与监督学习。同时，为了不使用户在长期的使用过程中感到枯燥乏味或功能单一，还要添加一些特殊的功能模块。为了实现此目标，我们在功能上做了以下设计：

1、小程序整体风格简约清晰，页面精美，没有太过冗杂的内容，不易分散学生的注意力，使学生能够更加专注地学习；

2、题库丰富，包含小学一到六年级的语文、数学、英语三个科目的海量题目，后台管理人员还会定期更新题库，保证小程序的实时性与新颖性；

3、刷题功能齐全，包含查看答案与解析、收藏题目、记录错题等功能，可随时查看收藏夹与错题集，也可随时修改删除其中的题目；

4、设计合理，刷题、收藏、错题集有快捷入口，可以快速进入，方便省时。尤其是“继续刷题”这一功能的实现，随时记录用户刷题情况，只要点击“继续刷题”就一定是从用户当前所刷题中上一次所刷的题目开始继续进行，避免用户每次需要重新查找题库，或者重复做已经做过的题，此可谓小程序的一大亮点；

5、学习神器模块，包含文本识别、智能翻译、AI 问答三样功能，文本识别支持 5mb 大小以内的图片中的印刷体文本识别、智能翻译支持近 20 种语言的互译、AI 问答实现用户与 AI 在线交互，能够回答绝大多数学习问题，此可谓小程序的又一大亮点。

1.3 可行性分析

为了方便用户的使用，我们将使用微信小程序的形式来实现学途神器这个软

件，接下来我们将从技术、工作量、市场、经济四个方面进行可行性分析。

从技术角度看，微信小程序开发有较为完善的流程与辅助工具，微信官方提供的技术文档能够满足我们绝大部分开发需求，如云端数据库用于存储用户信息、题目信息等，再如使用微信开发者工具可以较为方便地实现前后端交互。除此之外，为了实现我们设想的特色功能，我们选取了腾讯云的学生轻量化服务器，型号选取 WordPress，可以在我们的域名下轻松布置我们的展示页，也可以在后端实现对外的服务器接口搭建。我们选取轻量化 flask 框架，并把服务器的一个终端作为后端进行架设，现在市面上广泛可用的 API 接口也使得具体功能的实现成为可能。

从工作量角度看，我们使用 COCOMO 模型对工作量进行定量评估。记以人月（PM）为单位的工作量为 E，开发时间为 D 月，项目的代码行数估计值为 L（千行）。对于该项目的组织型软件，有模型公式如下：

$$\begin{cases} E = 2.4 * L^{1.05} \\ D = 2.5 * E^{0.38} \end{cases}$$

经小组的讨论与综合评估，取 L = 5，解得 $E \approx 13.01$，$D \approx 6.63$，所以具体参与软件开发的人数 $N = E/D \approx 2$。值得注意的是，上面计算出的 E 和 D 只作为中间结果，要对工作量进行更准确的评估需要采用中间 COCOMO 模型，加入 EAF 调节因子（使模型考虑到我们项目的特殊性和简易性）。基于这是一个课程项目而非商业项目的事实，对各个调节因子取值得到 $EAF = \prod_{i=1}^{15} Fi \approx 0.26$。这样可得 $E' = E * EAF \approx 1.95$，即需要接近两个月的开发时间，可以在软件工程结课验收前完成。所以，该项目满足工作量上的可行性。

从市场的角度来看，正如需求分析中所述，当今许多家长工作繁忙，没有足够的时间监督孩子做题，同时小孩也需要自主刷题以便巩固他们所学的知识，所以我们的小程序有着很大的市场空间。

从经济的角度来看，小组成员使用各自的计算机设备即可完成主要的开发工作，利用微信开发者工具和 gitee 线上平台，团队可以方便地以线上线下结合地方式开展交流讨论以协同工作进度。后端服务器方面，腾讯云具有专门针对学生用户的轻量化服务器，价格较为便宜，所以项目的初步开发不存在大额成本，满足经济上的可行性。

1.4 人员管理和项目进度管理

1.4.1 人员管理

我们团队的任务分配情况如下：

cxk：负责全过程文档整理、wiki 文档编写、分支项目集合、总体进度把控。负责软件功能设计、数据结构设计、云端数据库管理、云端数据库调用与可视化、应用整体逻辑优化等。

yy：负责服务器搭建与维护，参与全过程核心技术开发，包括服务器函数部署与测试、后端整体需求设计、后端函数测试更新、接口设计、后端数据库维护等，负责学习神器三样功能的实现。

csr：负责软件需求设计，基于墨刀平台进行 UI 设计，负责前端开发，包括大部分前端页面制作、全部页面的美化、图标设计等，参与小程序主体设计与软件测试，负责用户反馈的收集与整理。

cxy：负责 UML 相关需求分析，绘制 UML 相关图示。负责做题页面的制作与逻辑完善。负责软件测试，软件优缺点分析，软件的上传与发布。负责项目执行阶段成员的进度统计。

1.4.2 项目进度管理

总体上，通过甘特图完成项目初期具体任务的分派与规划；通过燃尽图在项目的全过程中对进度进行跟进和分析。

项目总体任务划分为：项目立项、UI 设计、软件设计、软件测试四个部分。将这 4 个阶段进一步细化成了一个个单独可执行的任务并进行了分配，项目甘特图如图 1-1 所示。

图 1-1 项目甘特图

小组各成员按照甘特图中所分派的具体任务完成项目，过程中定期进行进度统计，并通过燃尽图来判断当前进度是超前还是落后于预期，小组进度燃尽图如图 1-2 所示。由燃尽图可以看出，项目整体稳步推进，并在 10 月中下旬加快了进度。

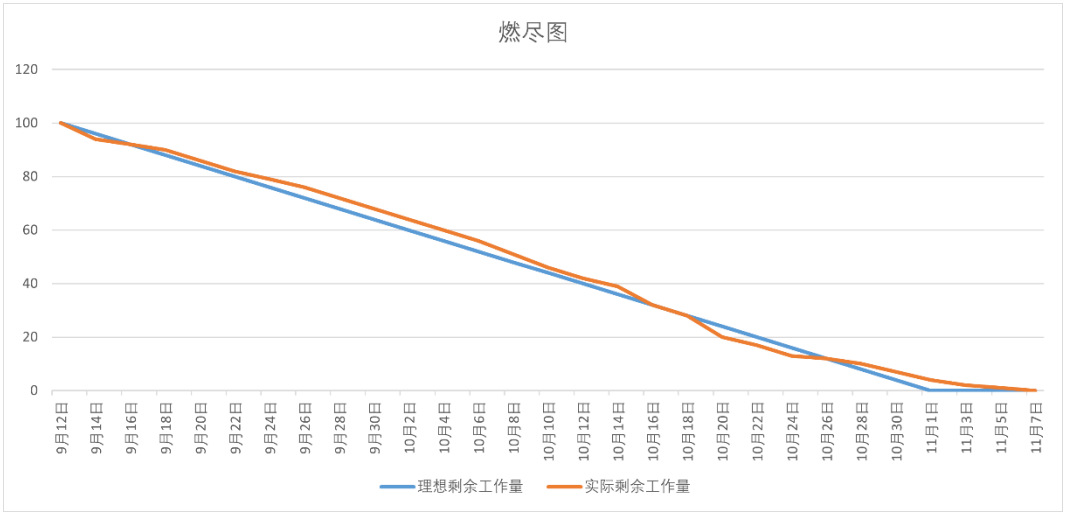

图 1-2 项目燃尽图

2 需求分析

2.1 需求分析概述

如 1.1 节中项目需求所述，我们小程序面向用户的需求主要为：解决当前教育环境下家长要与学生共同完成作业而家长力不从心的困境；小学生自主刷题的需求；便携式的刷题工具以便更好地利用“碎片化”时间；可重复练习的题集方便巩固知识；快速便捷的对答案方式；可更新的题集等。

综上，我们建立的需求分析表如表 2-1 所示。

表 2-1 需求分析表

社会需求	解决办法
家长/孩子/老师：刷题需求	做一个刷题类应用
便携的刷题工具	开发微信小程序
可重复练习的题目	收藏夹、错题集等
快速的对答案方式	做完题即刻出答案，可选择查看题解
多元化需求：不只是刷题工具	设置“学习神器”板块
不断能刷到新题目	开发者定期更新题库
更人性化的设置	继续刷题功能、反馈建议、自适应窗口、打卡功能、美化界面等

2.2 UML 相关需求分析图

学途神器小程序是一款面向所有小学生、小学教师、学生家长的刷题软件。用户登录进入软件，软件获取权限保存用户信息。小程序主要包含三个主页面：首页，学习神器以及我的。首页包含刷题，更换题集，打卡三个功能。学习神器包含文字识别，智能翻译，AI 问答三个功能。我的界面用于查看收藏与错题集，更改个人信息等。UML 缩略图如图 2-1 所示。

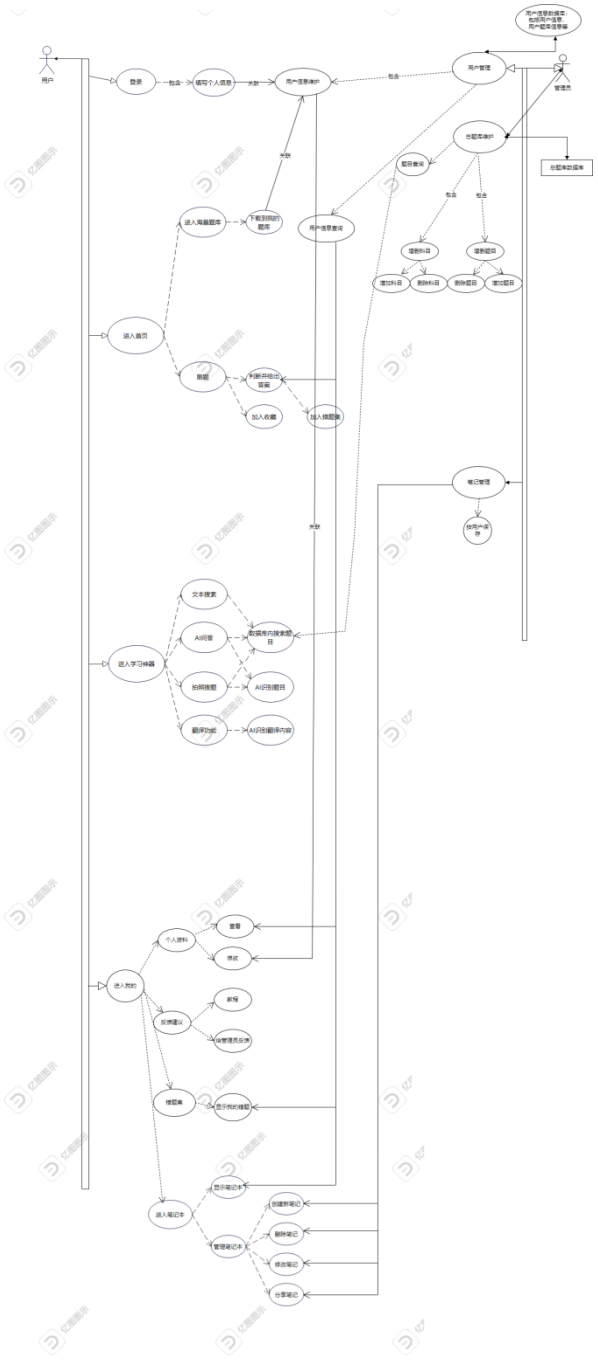

图 2-1 UML 缩略图

2.3 原型系统设计

在完成了功能设计和主要界面的构想后，我们使用墨刀进行了原型系统设计。虽然我们最终实现的小程序和原型系统设计有不小的出入，但是基本的思路其实是一致的。接下来，我们将具体来看登陆引导、首页、个人中心、题库、做题、我的题库、学习神器七个界面的原型设计。

2.3.1 登陆引导页面

在用户一进入应用时，应用就会自动申请获得用户的微信账号信息。如图 2-2 所示。

图 2-1 登陆引导

用户允许登录后，如图 2-2 所示，首先让用户选择所在的年级，随后将展示应用使用引导，如图 2-3、图 2-4、图 2-5 所示。

图 2-2 选择年级

图 2-3、图 2-4、图 2-5 应用使用引导

在最后一个引导界面，用户点击“开启学习之旅”，即可进入首页。在最后的小程序实现中，考虑到引导页面的作用不大，我们没有保留这个页面。

2.3.2 首页

在首页，我们主要放置了这样几个元素：用户头像代表的“个人中心”，打卡按钮，接着刷题与海量题库。如图 2-6、图 2-7 所示。

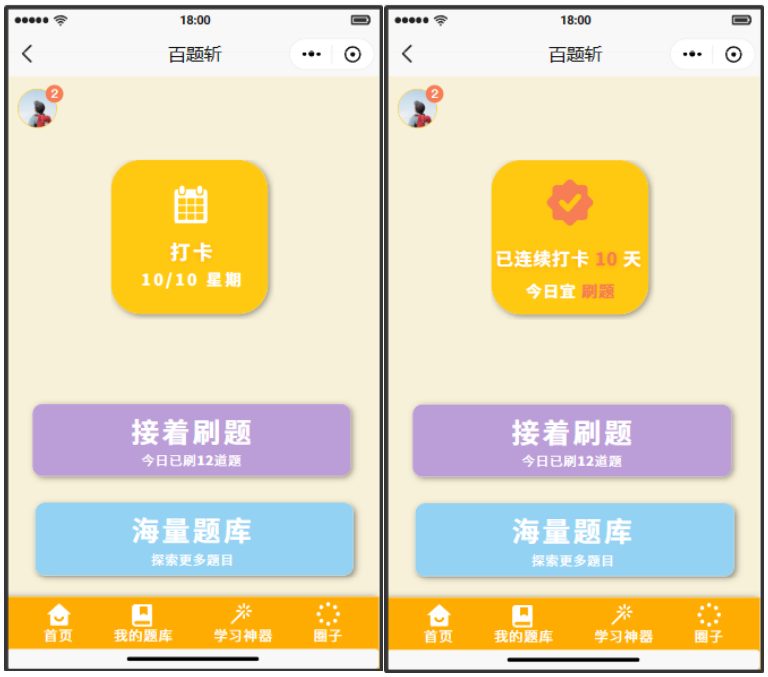

图 2-6、图 2-7 首页界面

在这里，我们着重介绍以下打卡的设计。在初始状态，打卡按钮会显示当日

的日期，在用户点击后，就会显示连续打卡的相关信息。

2.3.3 个人中心

在首页点击左上角的个人头像，就可以进入个人中心。我们在个人中心中设置了三个大选项，外观设置、学习设置、更多设置，并在右上角放置了消息按钮，这样可以充分满足用户的使用需求，如图 2-8 所示。

图 2-8 个人中心页面设计

考虑到外观设置与学习设置的相关内容会加重全局代码的负担，在最后我们没有按照原型设计的思路来设计小程序。

2.3.4 题库选择页面

在首页点击“海量题库”，就可以进入题库选择页面，如图 2-9 所示。在页面中，我们设计了科目和年级的选择，根据选择会匹配不同的题库。在题库条目中，会显示题库的相关信息，如是否已经练习过、答对题数、总体数等等。在题库条目的最右侧设置的“去练习”或“再练一次”按钮，直接链接到做题页面。

图 2-9 题库选择页面设计

在最后的小程序实现中，我们更换了更加便捷的条件选择方式，避免了因为前后端交互不同步而导致的筛选问题。

2.3.5 做题页面

在首页点击“接着刷题”或者在题库选择页面中点击练习按钮，就会进入做题页面。做题页面的主要元素有题目、选项交互、下一题、解析和收藏。根据用户选择的不同选项，页面会做出相应的响应，正确则选项变绿，错误则变红，如图 2-10、图 2-11 所示。

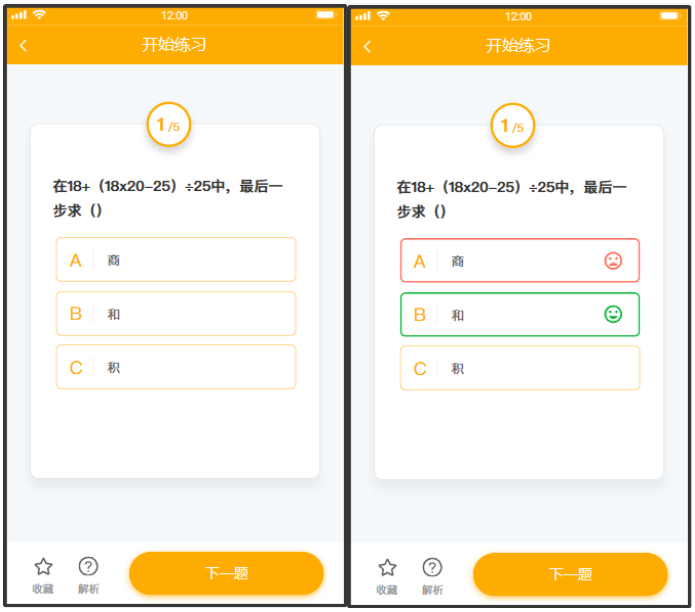

图 2-10、图 2-11 做题页面响应

用户提交答案后，还可以点击解析来查看解析，如图 2-12 所示。点击下一题就可以进入下一题的页面。

图 2-12 做题页面解析展示

在用户做完该题库的所有题目后，会有结算页面，如图 2-13 所示。

图 2-13 做题结算页面

2.3.6 我的题库

点击 Tabbar 中的“我的题库”按钮即可进入我的题库页面，如图 2-14 所示。

这个页面链接到了我的收藏、错题本、历史记录三个部分，这三个部分的页面与做题页面是类似的结构，因此在原型设计中我们并没有进行详细设计。

图 2-14 我的题库页面设计

在最后的小程序实现中，我们也将“我的题库”这一部分的功能转移到了“我的”即个人页面中。

2.3.7 学习神器

在原型设计时，我们计划的四个附加功能是 AI 问答、智能翻译、拍照搜题和文本搜题，如图 2-15 所示。

图 2-15 学习神器页面

点击页面中的功能按钮，就可以进入每个功能不同的页面。我们还单独设计

了这 4 个功能的展示页面，如图 2-16 至图 2-19 所示。

图 2-16 至图 2-19 功能页面展示

在最后的小程序中，考虑到本身题库的体量并不大，我们就删除了文字搜题和拍照搜题的功能，增加了文字识别 OCR 的功能。

3 概要设计和详细设计

3.1 系统结构

3.1.1 系统模块说明

整个小程序可以分为题库管理、学习神器、用户相关三大功能模块。在这三大模块中还有许多细分功能，下面进行详细介绍：

1、题库管理

面向用户来说，题库的相关功能有题库选择、接着刷题、错题集以及收藏夹 4 个细分功能。在后端数据库中，不同的题库有年级与科目的标签，用户可以通过选择年级以及科目来选择不同的题库。用户的刷题记录是随时在后台用户数据中更新的，每做完一道题，都会更新所在的题库 ID 和题目 ID，这就实现了用户随时登录都可以“接着刷题”的功能。错题集是在用户刷题的过程中自动记录的，只要做错，就会记录下题目 ID，加入该用户的错题集当中。而收藏夹功能是在题目页面加入了交互按钮，用户可以手动加入的。

对于管理者来说，题目在加上题目 ID、题库 ID 等相关信息后可以直接导入到后台数据库，也可以非常便捷地实现增删改等操作。

2、学习神器

学习神器模块包含了文字识别、智能翻译、AI 问答 3 个细分功能。在小程序中，用户发送的相关信息将被后台收集，通过调用不同的接口来实现具体功能。在文字识别模块，用户可以选择相册照片或使用相机拍照，后台将把识别出的文字内容反馈在文字框中。在智能翻译模块，用户输入的语言将由后台程序自动识别语言，而用户可以自由选择目标语言，并在文字框中得到相应反馈。在 AI 问答模块，用户和 AI 的对话形式是一问一答的，用户输入的内容将被后台接收，AI 生成相应问答后再发送给用户，整个问答流程表现为“对话”的模式，可以有效帮助用户解决学习上的难题。

3、用户相关

针对用户，我们实现的功能不仅仅是账户登录。用户的信息直接与它的刷题记录、错题本、收藏夹三个关键功能挂钩，AI 问答的记录也是直接按照用户信

息保存的。除此之外，我们在首页上加入了一个打卡的功能，用户可以进行每日打卡。在用户页面，我们也放置了一个意见反馈的部分，这样用户也可以比较快捷地反馈意见。

整个小程序的系统结构图如图 3-1 所示。

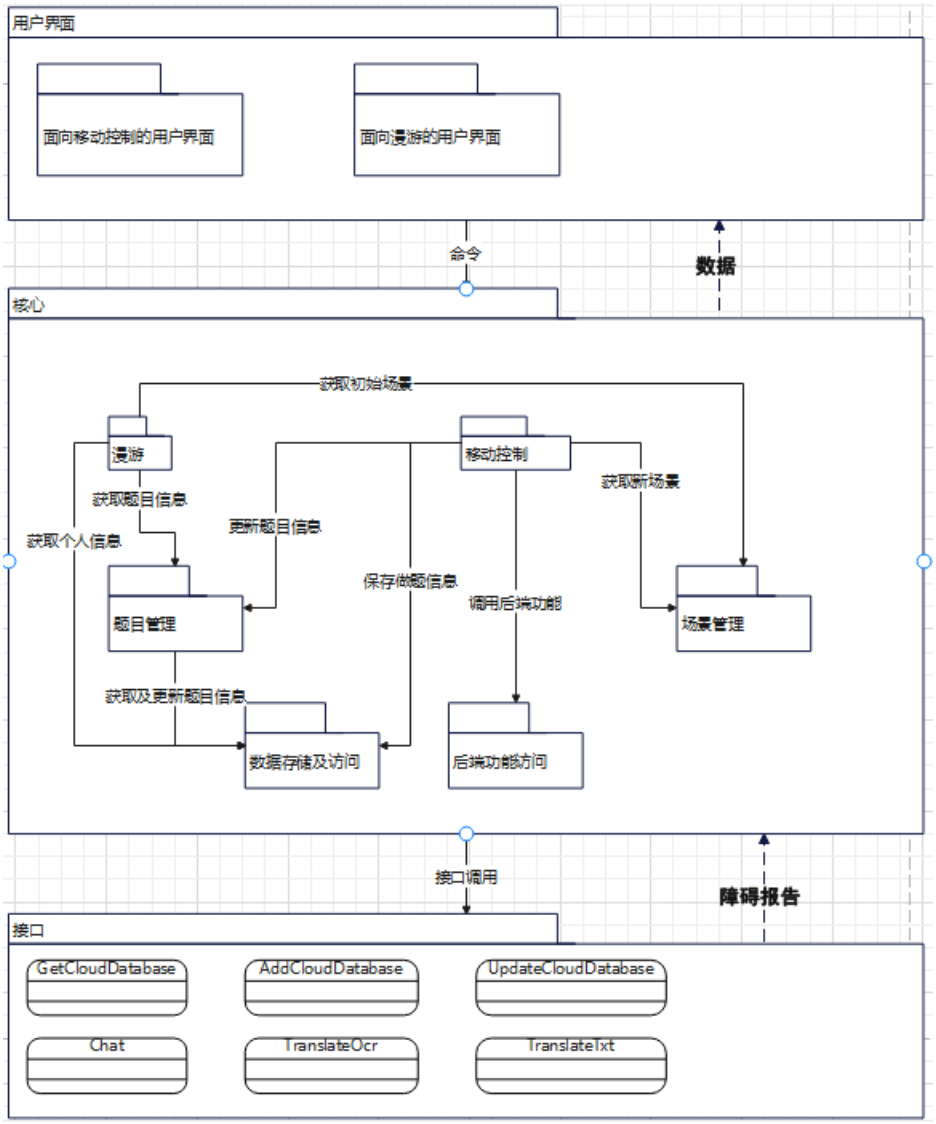

图 3-1 系统结构图

3.1.2 接口设计

1、云端数据库接口

在小程序的使用过程中，经常涉及到与云端数据库的交互，需要对数据库进行增删改查等一系列操作，我们直接使用微信小程序开发工具自带的函数作为接口来实现此项功能，如下所述。

（1）wx.cloud.database()

云数据库初始化，在对云数据库进行任何操作之前必须先进行此函数调用。以下用 db 代表此函数。

（2）db.collection().doc()

collection 获取某集合，doc 获取该集合中的某条（些）记录，可在括号内添加参数达到选取指定集合/记录的功能。在对数据库进行增删改查时，实际上是对数据库中指定集合的指定记录进行的，所以几乎所有调用数据库的场景均需使用此函数。以下用 record 代表此函数。

（3）db.collection().add()

功能为向指定集合中添加一条记录。在用户首次登录时，会自动创建用户信息、收藏夹、错题集等条目，此时会用到该函数。

（4）record.update()

功能为更新指定集合指定记录中的数据。在更新错题集、更新收藏夹、更新用户做题信息等数据库中的内容时，会用到此函数。

（5）record.get()

功能为获取指定集合指定记录中的数据。该函数可以说是本次项目中用到最多的函数，题库中题目的显示、收藏夹中题目的显示、错题集中题目的显示、用户信息的获取、AI 问答的内容等许多方面均需要通过此函数来实现。换句话说，只要需要用到数据库中的信息做判断，或者需要将内容显示给用户时，就需要使用此函数。

2、后端接口

鉴于我们实现了“学习神器”板块中的三样功能，本次项目对后端的需求量是较大的，如何沟通前端的请求函数与后端的路由端口也成为重点。通过对我们设计的分析可以得知需要一个和 GPT 语言模型沟通的接口，一个翻译文本的接口，和一个从图片中提取文字的 OCR 接口，分别如下所述。

（1）/chat：OpenAI交互接口

```python
@server.route('/chat', methods=["POST"])
def chat_with_AI():
    msg = request.get_json().get('word')
    uid = str(request.get_json().get('uid'))
    session = get_chat_session('P'+uid)
    try:
    # 设置本次对话内容
        session['msg'].append({"role": "user", "content": msg})
    # 设置时间
        session['msg'][1] = {"role": "system", "content": "current time is:" + get_bj_time()}
    # 检查是否超过tokens限制
        while num_tokens_from_messages(session['msg']) > config_data['chatgpt']['max_tokens']:
        # 当超过记忆保存最大量时，清理一条
            del session['msg'][2:3]
        # 与ChatGPT交互获得对话内容
        #message = "非常抱歉，IWS主人正在与错误代号： <502>  进行斗争，请等待服务恢复！\n"\
        #  "这段时间内，newbing服务仍将继续，请使用认真模式来操作。小iws依旧爱您！"
        message = chat_with_gpt(session['msg'])
    # 记录上下文
        session['msg'].append({"role": "assistant", "content": message})
        print("              : ")
        print(message)
        return message
    except Exception as error:
        traceback.print_exc()
        return str('异常: ' + str(error))
```

图 3-2 /chat 接口

如图 3-2 所示，可以看到在设计中将此接口路由在了“/chat”上并指定了请求方法为“POST”。在传参 data 中指定接收的参数为“word”（发给 AI 助手的话）以及“uid”（用户的唯一区分码）。获取到这两个传参后，利用 get_chat_session()函数来获取 uid 所对应的相应的 Session。通过在 Session 中新加入记录的方式来更新对话，通过 chat_with_gpt()函数来实现远程 Session 的更新，从而得到回参（AI 的回答）。唯一值得注意的地方就是每一个 Session 储存对话的记忆量（Tokens）是有限的，所以需要特殊判定来维护其健康。

此外，uid 也可以通过路由的方式来定义带变量的 API 接口，如/chat/<uid>但是考虑到前端实现的问题还是简单为好。

（2）/translate_txt：有道智云文本翻译接口

```python
@server.route('/translate_txt', methods=["POST"])
def translate_Youdao():
    msg = request.get_json().get('word')
    Trans_to = request.get_json().get('trans_to')
    TR.set_to(Trans_to)
    text = TR.connect(msg)[0]
    return text;
```

图 3-3 /translate_txt 接口

如图 3-3 所示，本接口和上一个接口的基础实现形式是大体一致的，请求方式当然也是 POST。 在传参 data 中指定接收的参数为“word”（需要翻译的文本）还有“trans_to”（翻译的目标语言）。这里看起来代码不多是因为绝大多数

代码依旧通过 TR 这个类封装起来了（TR 在后端的其他文件中，此处不做赘述），然后更新 TR 中参数后通过 TR.connect（）函数来获得翻译结果。

（3）/translate_pic：有道智云OCR接口

```
@server.route('/translate_pic/<openid>', methods=["POST"])
def upload_trans(openid):
    try:
        openid = str(openid)
        # 获取上传的文件数据
        print("1")
        image_data = request.get_data()
        #print(image_data)
        msg = TRP.connect(q = image_data);
        #msg = 'success'
        # 解码Base64字符串为字节数据
        #image_bytes = base64.b64decode(image_data)
        # 将字节数据转换为图像
        #image = Image.open(BytesIO(image_bytes))
        # 保存图像文件
        #image.save('uploads/'+ openid + 'image.png')  # 保存路径可以根据需求自定义
        return str({'error': 0, "data": {'msg': msg}})
    except Exception as e:
        print(e)
        return str({'error': 1, "data": {'msg': e}})
```

图 3-4 /translate_pic 接口

如图 3-4 所示，此 OCR 接口被路由到了“/translate_pic/ <openid>”，采用了“POST”的请求方法，然后运用到了我们前文中提及的定义变量的接口 API。在原则上，如果需要在后端保存图片的话，需要随机生成图片的编号 ID 和原名字结合，然后统一路径保存，在上图中也可以看到被注释的代码即是保存方法，但是本次课设中需要没有到这种查询 OCR 记录的地步，所以直接传参（我们设置了传参 data 一整个为 base64 编码的字符串）给封装好的 TRP 类，然后通过 TRP.connect()函数得到期望的回参。

3.2 类图等

运用类图等方法说明系统的设计，自顶向下地讲解程序主体部分的设计与原理。主页对信息和功能进行综合，自顶向下包含题集，错题集，收藏夹，题目，选项等结构，小程序主体 UML 类图如图 3-5 所示。

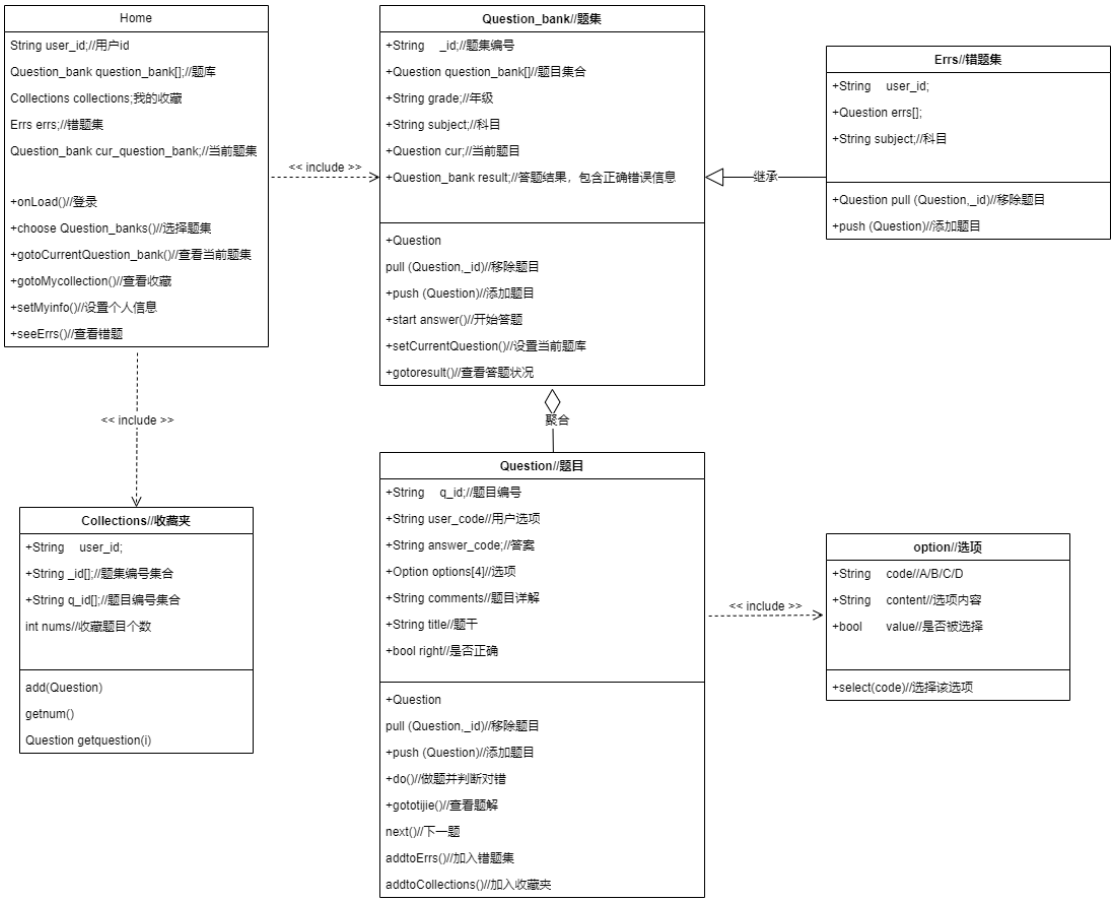

图 3-5 小程序主题 UML 类图

下面逐一介绍这些类的功能及关系：

主页（Home)：

用户选择登录，在未登录状态时，可以选择题库进行答题，但软件不会保存用户身份信息和学习记录。登录时需要获取用户相关权限。登录后，用户可以进入题库选择题集进行答题，直接答题，查看收藏夹以及错题集。

题集(Question_bank)：

题集保存在云数据库，采用微信云开发 CMS 架构，根据不同学科和年级分配题集编号。管理员具有增删题库的权限，定期更新题库数据库，包括增加热点题目和删除冷门题，以保持小程序的高效性和实用性。用户只有对题集的读取权限，以防止恶意删改。

错题集(Errs)：

错题集继承自题集，用户答题后会生成临时错题集，系统判断对错并核分以便用户迅速了解当前答题情况。永久性错题集保存用户在登录状态下每次答错的

题目，方便用户直接进入错题集开始答题，查漏补缺。用户也可以随时将已经巩固完成的题目从错题集中移除，以保持错题集的简洁高效。

收藏夹(Collections)：

收藏夹用于保存用户在答题时收藏的单个题目，用户具有查询和删改权限。

题目(Question)：

为方便管理与判断，题目通常是四个选项的选择题。每道题目有一个唯一的 q_id 题目编号（用于在收藏夹和错题集中加以区分），包括题干（title）、四个选项（options）以及题目详解（comments）。

选项(Option)：

选项包含在题目中，每个选项包括 code（A/B/C/D）、content（选项具体内容）以及 value（是否被选择）等信息。

3.3 关键数据结构定义

3.3.1 云端数据库数据

为更好地实现小程序目标功能，云端数据库需要保存题库、题目、用户信息、收藏夹、错题集、AI 对话记录这六大部分的数据，其中，题库、题目为开发者导入，用户不可修改，其余四部分在用户首次登录时自动创建，之后每次登录调用对应的记录，并随着用户的使用进行更新操作，数据变化流程图如图 3-6 所示。

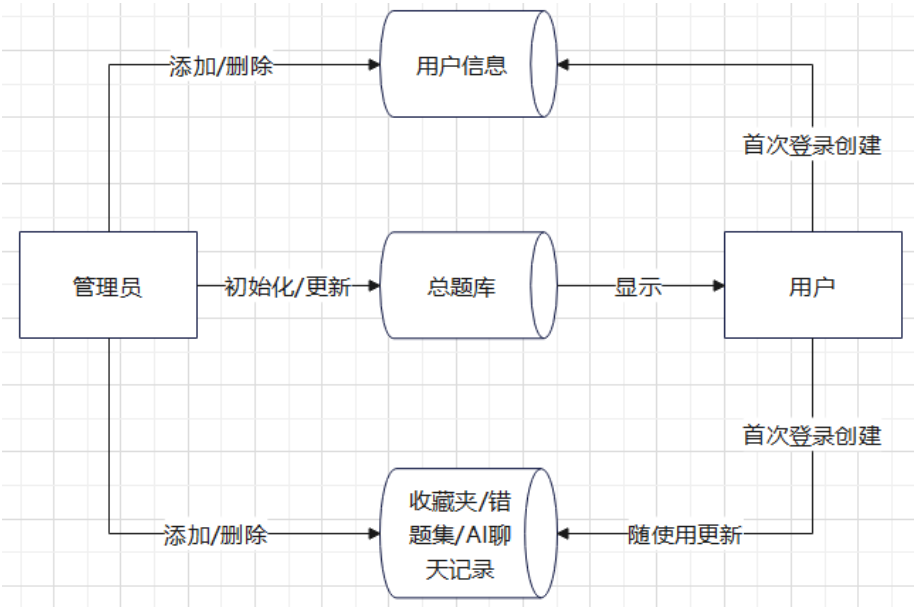

图 3-6 数据变化流程图

1、题库

题库集合命名为 topic_warehouse，保存了多个题集的具体数据，用来为用户提供题目。在小程序中，题集按照阶段和科目进行区分，因此该数据集合中的一条记录就是一本题集，需要保存的信息有：①id：唯一标识符，作为题集的查找依据；②阶段/科目：方便检索；③包含的题目：保存该题集中所有具体题目的 id；④数量：该题集包含题目的数量。

2、题目

题库中的一条记录为一份题集，一份题集由数道题目组成，题目统一保存在 topic 集合中，需要保存的信息有：①id：唯一标识符，作为题目的查找依据；②题干；③解析；④选项：包含选项的 code（比如 A）和 text（内容）；⑤答案；⑥所属题库：只做科目的区分，方便在用户做错题目时自动加入到错题集。

3、用户信息

用户信息对于差异化不同用户的收藏夹、错题集等功能起着至关重要的作用，保存用户信息的集合命名为 user_information，需要保存的信息有：①id：唯一标识符，作为用户信息的查找依据；②年级：标识用户的年级，可在“我的”界面进行修改；③头像：保存用户头像，使用开发工具自带的函数可以获取用户的微信头像；④用户名：保存用户昵称，使用开发工具自带的函数可以获取用户的微信昵称；⑤当前题库：保存用户当前正在刷的题库；⑥当前题目：保存用户上一次做到的题目（一定是“当前题库”这个题库中的题目）

4、收藏夹

收藏夹保存了用户主动收藏的题目的合集，用户在做题时可以随时将题目取消收藏。保存收藏夹的集合命名为 collection，需要保存的信息有：①id：唯一标识符，作为收藏夹的查找依据，直接以用户信息 id 作为此项；②收藏数量：收藏题目的数量；③收藏题集：保存了收藏的每一道题目的 id，方便做收藏夹中的题目时对数据库的查找。

5、错题集

错题集保存了用户刷题过程中（包括刷题库的题目和刷收藏的题目）做错的那些题，用户在从“我的错题集”进入重做错题时，可以随时将题目移出错题集。

为了方便用户使用，错题集按照科目进行了分类，语数外三科的错题集集合名称分别为 mis_chinese、mis_math、mis_english，需要保存的信息一致，即：①id：唯一标识符，作为错题集的查找依据，直接以用户信息 id 作为此项；②错题数量：（该科目）做错的题目的数量；③错题集合：保存了（该科目）做错的每一道题目的 id，方便做错题集中的题目时对数据库的查找。

6、AI对话记录

此项专门为 AI 问答功能创建，用于保存用户与 AI 对话的每一条记录，保存此项的集合命名为 chat_record，需要保存的信息有：①id：唯一标识符，作为 AI 对话记录的查找依据，直接以用户信息 id 作为此项；②对话记录（record 数组）：数组中的每一个元素都是一个 object，保存四项属性：发言者头像、发言者名称、发言文本、发言者 ID（用户的发言者 ID 并非用户 ID，这一项这是为了区分发言的是用户还是 AI），每一个 object 就是用户界面下一次性发送或接收到的内容。

AI 问答功能的实现思路是：每次用户输入完内容点击发送之后，将用户这一次发言对应的 object 存入云端数据库，调用后端 AI 的返回结果也更改为对应的 object 存入云端数据库。在进入页面之初，我们定义了一个 chatList 变量，让其时刻等于最新的 chat_record 集合中的那个 record 数组（即无论是用户点击发送消息后还是后端调用 AI 返回结果后，都用 this.setData 更新一下 chatList），在前端，使用 wx:if="{{chatList}}"对显示的内容进行循环渲染，若某条记录的发言者 ID 代表 AI，则此记录显示在用户界面的左侧，头像和名字都显示成 AI 的，反之，则此记录显示在用户界面的右侧，头像和名字都显示成用户的。

正因上述用户界面的实现逻辑，才特此设置了“AI 对话记录”这一集合。

3.3.2 后端数据

后端数据主要保存在 session 数据结构中，如图 3-7 所示。

```
session_config = {
    'msg': [
        {"role": "system", "content": config_data['chatgpt']['preset'][0]}
    ],
    'send_voice': False,
    'new_bing': False,
    'send_voice_private': False,  ##
    'send_emoticon' : True,   ##
    'send_answer' : True,
    'prompt_index' : 0,
    'models_index' : 0,
    'group_reply' :False,
    'message_history': {

        }
}
```

图 3-7 session 数据结构

session 是用户和智能语言模型之间的重要交互途径，以下将解释每一条参数的作用：

【‘msg’ ---- 保存所有已有的消息记录（主记忆）

‘send_voice’ ---- 是否允许向用户发送语音

‘new_bing’ ---- 是否切换模型到 newbing（通过 cookies 实现 newbing 访问）

‘send_voice_private’ ---- 是否读取用户的语音设置需求

‘send_emoticon’ ---- 是否允许向用户发送表情包

‘send_answer’ ---- 是否允许直接回复用户题目答案

后四条服务于尚未实装的群聊功能（后端已经实现）】

3.4 关键算法设计

3.4.1 前后端交互连接框架设计

前后端交互的主要需求就是学习神器三大件的各种交互请求在微信小程序和腾讯云服务器后台直接通讯的实现。由于微信的自带的 wx.request 以及 wx.upload 函数对服务器域名有极其高的要求（支持 ssl 安全证书的经过我国 icp 备案的域名），这对我们的短期工程开发非常不友好，所以我选择去重载这些函数。其中运用到的核心技术就是微信可选择的云函数功能，它提供给了我一个搭建外层封装的可能性，也就是在微信小程序和服务器直接隔了云开发这样一层，绕过了对微信小程序严格域名审核。

前端云函数框架设置如图 3-8 所示。

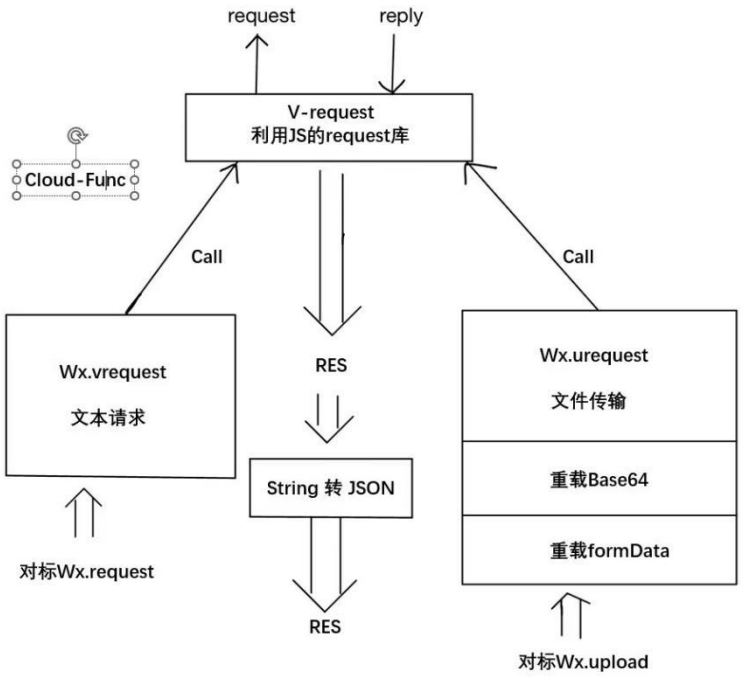

图 3-8 前端云函数框架

可见，我们设计了 wx.vrequest 函数来对标微信自带的 wx.request 函数，还有 wx.urequest 函数来对标微信自带的 wx.upload 函数。对于这两个新函数，实质上就是对不同数据进行了不同形式的处理在传 data 数据给 v-request 这个云函数内核（用 js 的 request 库编写），指的特别注意的是，微信的 formData 以及 base64 库不完全和 js 的库等价，所以如果我们需要和我们内核函数兼容，需要重写新的 formData 结构已经 base64 转码库。

当云函数 v-request 返回值得到正确提示时，开始判断返回的是 string 还是 json，并进行不同的处理，再转交给前端页面。

3.4.2 后端算法设计

1、语言模型交互流程设计

我们在后端设计了许多封装类，参考了官方文档来实现和 API 远程仓库的交互，后端接入了 OpenAI 的语言模型，有道智云的学习服务，Replicate 的 Diffusion 模型等等，这里单取 GPT3.5 模型的交互逻辑框架来展开讲述，如图 3-9 所示。

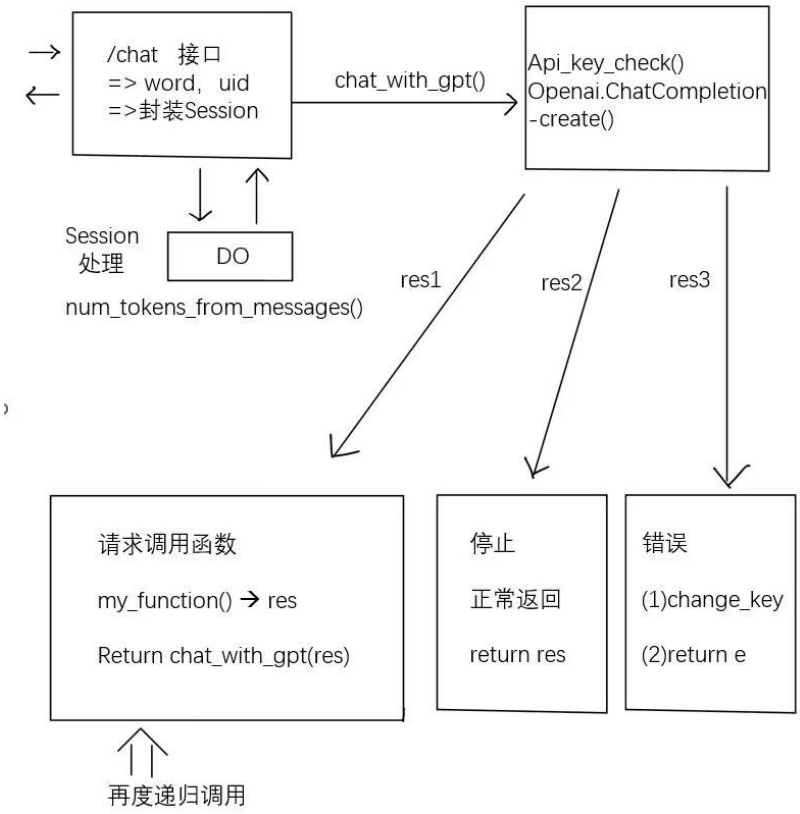

图 3-9 后端 GPT 交互框架

可以清晰看到当 flask 的路由接口“/chat”收到交互请求的时候，端口绑定函数会从请求数据中提取出 word 和 uid 来构建新的 session（同时由 num_tokens_from_messages() 函数来判断记忆量是否健康），它会调用 chat_with_gpt()函数来传输 session，利用 Api_key_check()函数来检查 apikey 的合法和未被占用，然后调用远程库的核心函数 openai.chatCompletion. create()函数来实现对 OpenAI 的远程模型的请求。最关键的地方是三种对回参的处理模式，这里特别提一下 openai 强大的自定义函数功能。当 gpt 模型得到本地函数列表，而且判断你的问题可以被这些函数解决时，它会返回一个调用函数请求，同时把自己压入更深的一层递归之中。

以我们设计的搜索函数为例，如图 3-10 所示。

```python
#函数说明
my_functions = [
    {
        "name": "search_internet",
        "description": "通过互联网搜索信息的方法",
        "parameters": {
            "type": "object",
            "properties": {
                "query": {
                    "type": "string",
                    "description": "搜索关键词",
                }
            },
            "required": ["query"],
        },
    }
]
# 搜索互联网
def search_internet(query):
    res = requests.post(url="https://duck.lucent.blog/search", json=query).json()
    return res
```

图 3-10 关于 GPT 模型自定义函数设计

在同 GPT 远程模型交互过程中，函数表（包含各项描述）将被传给它，然后它会根据对话交互内容的具体需求来判断是否需要调用函数表中的具体函数。如上图的“search_internet”函数的描述就是“通过互联网搜索信息的方法”，它可以弥补 AI 的训练数据不紧跟时事的缺点，当 AI 判断它无法回答用户的“未来式问题”时（AI 的训练滞后于时代），它就会返回请求来调用此函数。而我们可以看到此函数的实现其实就是通过我自己挂载的一个搜索引擎来大批量爬取网上的各种新闻热点（此引擎的实现不做赘述，涉及另一个后端），而爬取到的内容会在一度返回给 GPT 模型，它再分析这些数据来得到用户真正需要的答案。

2、翻译/图文识别的对象类设计

自然语言翻译对象类和 OCR 光学识别对象类代码如图 3-11 和图 3-12 所示。

```python
#有道自然语言翻译
class Trans_Fuc():
    def __init__(self, APP_KEY = config_data['youdao']['APP_KEY'], APP_SECRET = confi
        reload(sys)
        self.YOUDAO_URL = 'https://openapi.youdao.com/api'
        self.APP_KEY = APP_KEY
        self.APP_SECRET = APP_SECRET
        os.environ['NO_PROXY'] = "https://openapi.youdao.com/api"
        self._to = 'zh-CHS'
    def set_to(self, T_to):
        self._to = T_to
    def encrypt(self, signStr):
        hash_algorithm = hashlib.sha256()
        hash_algorithm.update(signStr.encode('utf-8'))
        return hash_algorithm.hexdigest()
    def truncate(self, q):
        if q is None:
            return None
        size = len(q)
        return q if size <= 20 else q[0:10] + str(size) + q[size - 10:size]
    def do_request(self, data):
        headers = {'Content-Type': 'application/x-www-form-urlencoded'}
        os.environ['NO_PROXY'] = self.YOUDAO_URL
        return requests.post(self.YOUDAO_URL, data=data, headers=headers)
    def connect(self, input_str, _from = 'auto'):
        q = input_str
        data = {}
        data['from'] = _from
        data['to'] = self._to
        data['signType'] = 'v3'
        curtime = str(int(time.time()))
        data['curtime'] = curtime
        salt = str(uuid.uuid1())
        signStr = self.APP_KEY + self.truncate(q) + salt + curtime + self.APP_SECRET
        sign = self.encrypt(signStr)
        data['appKey'] = self.APP_KEY
        data['q'] = q
        data['salt'] = salt
        data['sign'] = sign
        data['vocabId'] = "您的用户词表ID"
        response = self.do_request(data)
        contentType = response.headers['Content-Type']
```

图 3-11 自然语言翻译对象类

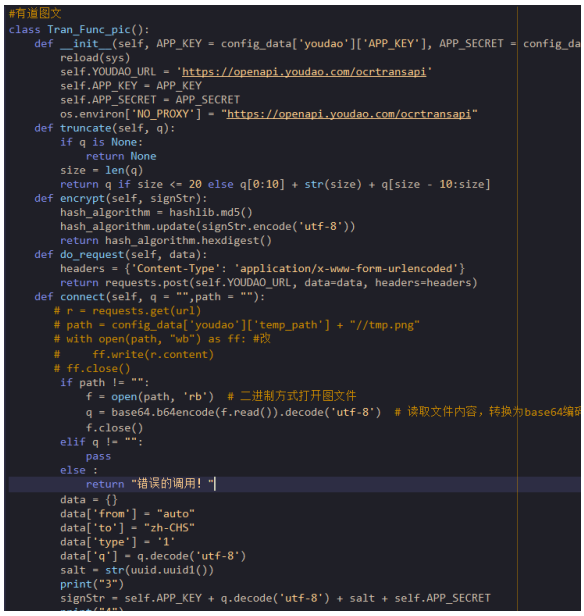

图 3-12 OCR 光学识别对象类

我们参考阅读了许多有道智云的官方 API 调用文档，在取得开发者申请后我设计了以上两个调用 API_KEY 的对象类（可以理解为翻译器），封装了很多子函数，使得我们能在不触及后端详细代码编写的情况下设计前后端连接的路由接口，整体代码框架显得更加清晰，同时能够让不了解具体函数实现的团队成员可以轻松黑盒调用此函数。主要需要调用的就是 connect（）函数，其中 connect 函数的传参也会调用类中的其他函数来对类的私有属性进行修改（比如自然语言翻译的设置翻译目标语言），更好地服务于我们的前端需求。

3.5 数据管理说明

小程序所涉及到的几乎全部数据都在云端数据库中进行了保存，包括用户个人信息、题库、题目、错题集、收藏夹等等，这里将进一步对云端数据库的管理及调用进行讲述；后端虽存储有少量数据，但与用户管理不大，存储形式及调用方法分别在 3.3 关键数据结构定义和 3.4 主要算法这两节中有过讲述，故这里不再进行叙述。

3.5.1 数据导入

微信小程序自带的云端数据库支持一键导入，如图 3-13 所示。

图 3-13 云端数据库一键导入

这里我们选取 json 文件进行导入，以题目（topic）集合为例，导入形式如图 3-14 所示。每一道题目必须用大括号括起来，两道题目之间不能有其他符号。

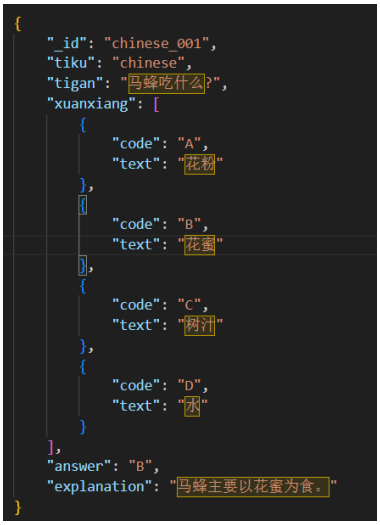

图 3-14 导入的 json 文件的数据格式

3.5.2 数据访问

1、增加记录

以用户登录时调用的代码为例进行说明，如图 3-15 所示。

```
db.collection('user_information').add({
  data:{
    _id:this.data.userOpenId,
    grade:null,
    image:this.data.userInfo.avatarUrl,
    name:this.data.userInfo.nickName,
    topic_id:null,
    topic_warehouse_id:null
  },success(res){
    console.log('添加数据成功',res)
  },fail(res){
    console.log('添加失败',res)
  }})
```

图 3-15 用户登录时创建用户信息的代码

使用 db.collection('user_information').add() 函数往云端数据库中的 user_information 集合增添一条记录，data 后面的内容即为增添的数据。在此之前已经用 wx.getUserProfile()函数进行了用户信息的读取，userInfo 为函数返回值，用户名、头像等信息都存在其中。success 函数规定添加记录成功时在控制台输出“添加数据成功”，fail 函数规定添加记录失败时在控制台输出“添加失败”。

2、修改记录

以往错题集添加题目调用的代码为例进行说明，如图 3-16 所示。

```
db.collection('mis_chinese').doc(app.globalData.globalUserid).update({
  data:{
    err_topic: _.push(this.data.subject._id),//当前错题添加到错题本
    err_num: _.inc(1) //错题本数据更新
  },
  success(modify){
  console.log('加入错题集成功',modify)
  },
  fail(modify){
  console.log('加入错题集失败',modify)
  }
})
```

图 3-16 往错题集添加题目的代码

使用 db.collection('mis_chinese').doc().update 更新云端数据库中 mis_chinese 这个集合中的指定记录，查找依据就是 doc 括号里面的 app.globalData.globalUserid，这是在用户登录时就已经被赋值的用户 id（具有唯一性），data 后面的内容即为更新内容，图中的_.push()函数实际上是 db.command.push()函数，_.inc 函数同理，由于 err_topic 是一个数组，所以需要用 push 函数往里新增记录，inc 函数用于使记录中的数值型数据增加或减少指定常量。success 函数规定修改记录成功后输出“加入错题集成功”，fail 函数规定修改记录失败后输出“加入错题集失败”。

3、读取记录

以做题界面读取题库中的题目所调用的代码为例进行说明，如图 3-17 所示。

```
db.collection('topic').doc(app.globalData.globaltopic_id).get().then(res=>{
  this.setData({
    subject:res.data,
    real_answer:res.data.answer
  })
})
```

图 3-17 读取题库中题目所调用的代码

读取数据库内容的代码比较简单，但用处极为广泛，可以看到，使用db.collection('topic').doc().get()函数进行对云端数据库中 topic 集合中指定记录的信息读取，app.globalData.globaltopic_id 是在用户登录时即被赋值，在用户做题中随时更新的用户做题记录，then 函数表示需要先执行 get 函数才能进行后续代码，并且 get 函数的结果将作为 res 在 then 函数中使用。后面在 this.setData 函数中给相应变量赋值即可。需要注意的是，this.setData 函数用于将数据从逻辑层发送至视图层，也就是说，只要想让用户看到界面上某处内容的变化，就必须在 this.setData 函数中给对应变量赋值，而那些只起后端逻辑关系判断等作用的变量，不必须在此函数内赋值。

3.6 界面设计

将原型设计转化为小程序的实际页面，我们主要要编辑 wxml 文件、wxss 文件、json 文件和 js 文件。微信的 wxml 文件其实和 html 语言有着基本一样的结构，wxss 则是基本继承了 css。在这份报告里，我们不就如何调整样式等内容详细展开，而是选择了美观性、适应性、便捷性、健壮性、协同性五个方面来展示小程序的特点。

3.6.1 美观性

为了让小程序更加美观，我们主要注重了两个方面，一个是风格的统一，一个是多样化的图标设计。

1、风格统一

小程序主要采用了如图 3-18 所示的配色，在所有页面中我们都运用了统一的配色，保证了用户的视觉体验。值得一提的是，我们并没有选择纯黑色我作为

文字的主体颜色，而是采用了饱和度更低的黑色来防止视觉疲劳。

图 3-18 小程序配色

此外，小程序中有相似元素的部分，我们也采取了统一的设计。如首页的选项就和选择科目的选项以及选择学习神器的选项采用了相同的设计，如图 3-19 所示。

图 3-19 相同设计的运用

2、图标设计

在小程序中，我们加入了多样的、贯彻主题色的图标，这些图标可以打破文字的沉闷，也增强了用户的体验。图 3-20 是我们小程序中的一些图标。

图 3-20 小程序图标

3.6.2 适应性

1、元素单位选择

在代码实现的过程中，我们主要采用了相对单位“vh”和“vw”，即当前元素与屏幕高度/宽度的相对长度，完美保证了小程序在不同的机型中都可以正常显示。如图 3-21 所示，左边机型为 iPhone6，右边机型为 iPhone12，可以看出都可以非常正常地显示。

图 3-21 不同机型显示对比

2、Sroll-view 的应用

在小程序中，很多时候我们都会遇到长文本的情况。微信开发工具自带了一种属性 scroll-view，这种属性也在我们的小程序中得到了非常广泛的应用。无论是题目过长、翻译内容过长，还是反馈意见过长，都可以用 scroll-view 进行滑动浏览，如图 3-22、图 3-33、图 3-34 所示。在 AI 对话功能中，用户与 AI 的对话内容也是放在一个 scroll-view 框架中的。

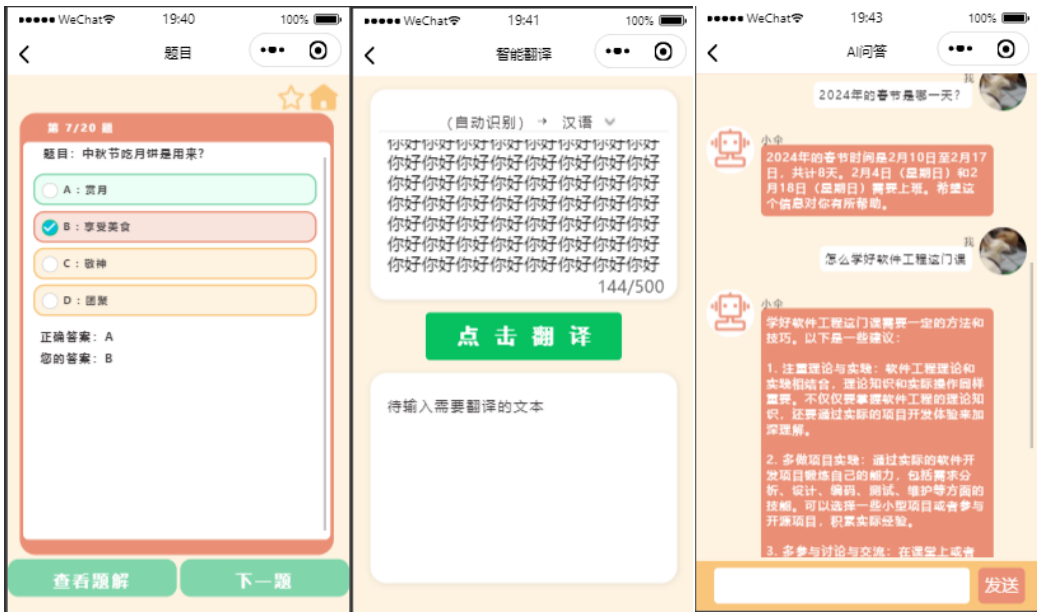

图 3-22、图 3-23、图 3-24 scroll-view 在小程序中的运用示例

在我们的小程序中，我们可以通过如图 3-25 所示的方式对 scroll-view 的属性进行设定，因为我们需要实现的是上下滑动，所以将 scroll-y 设为了“true”。在实际操作中，我们发现如果同时设定“scroll-x=’false’”，那么将会变为左右滑动，因此我们最终还是选择了值设定 scroll-y 属性。

```
<scroll-view style="height: 90vh;" scroll-y="true" >
<block wx:for="{{chatList}}">···
</block>
</scroll-view>
```

图 3-25 scroll-view 的设定

3、对话窗口的实现

在小程序中，AI 对话的前端实现是一个难点。在这里，我们着重介绍我们是如何正常显示对话的文字的。要求既要能正常换行，又要能保证背景底色不仅仅只在有文字的地方出现，还要整体呈现出一个矩形的样式。但是相关的内容在网络上很难找到，经过我们的不断探索，我们最终发现，我们需要在 wxss 中将文字的 display 属性调整为 inline-block，并对于溢出文字和换行进行设定，如图 3-26 所示。

```
line-height: 3vh;
display:inline-block;
text-overflow: ellipsis;
word-wrap: break-word;
```

图 3-26 对话框属性设定（节选）

经过这样的设置，即便我们要展示的文本很长，也可以以非常合理地形式出现。如图 3-27 所示。在实现的过程中，我们还遇到了诸如无法向右对齐、人机交互区域紊乱等问题，在此就不再进行详细讲解。

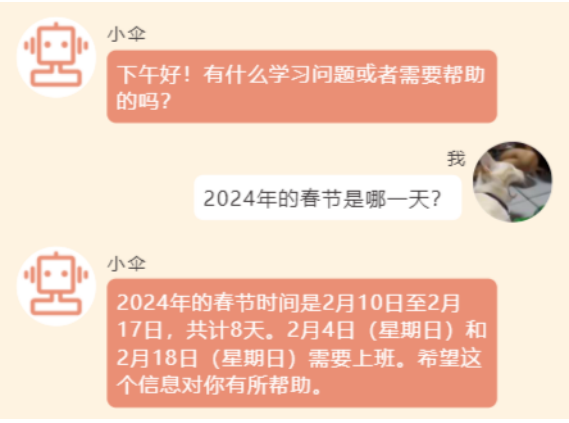

图 3-27 AI 对话界面截图

3.6.3 便捷性

为了减轻后端的代码负担，在程序设计时我们合理利用了非常多微信开发工具自带的属性。

1、radio 的使用

radio 在微信开发者工具中的意思“单选框”。在 wxml 文件中，我们的选项展示逻辑如图 3-28 所示。

```
<!-- 这里是题干 -->
<view class="question">题目: {{subject.tigan}}</view>
<!-- 这里是选项 -->
<radio-group bindchange="radioChange">
  <view wx:for="{{subject.xuanxiang}}" class="option">
    <radio value="{{item.code}}"  checked="{{isSelect==item.code}}"/>
    <text>{{item.code}} : {{item.text}}</text>
  </view>
</radio-group>
```

图 3-28 选项展示逻辑

为了展示所有选项，我们需要 wx:for 完成数组中选项的遍历，而 radio 则与每一个选项的码绑定，在选项最外层绑定一个 bindchange，那么在用户提交后，后端就可以接收到用户选择了哪一个选项。Radio 的优势是，我们不需要增加别的逻辑来判断用户选择了哪个选项，radio 本身就可以返回用户选择了第几个选项。这大大减轻了我们后端代码的负担。

为了保证美观性，我们还需要在选项外套用 view 来进行修饰，如图 3-29 所示。当然，radio 有一个很大的缺点，就是用户在选择时必须点击选项最左侧的小圆圈才能选中，这对用户的操作其实并不友好，但是为了简化代码，我们不得不选择用 radio。

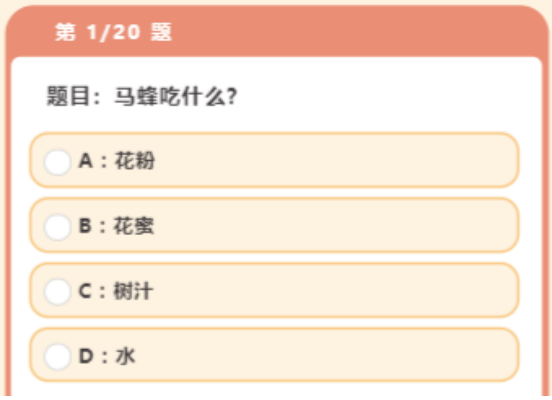

图 3-29 美化后的选项展示

2、textarea 的使用

Textarea 在小程序中翻译、AI 问答、意见反馈等很多需要用户输入的地方得到了广泛的使用，在这里我们以用户反馈为例，代码如图 3-30 所示。

```
<textarea maxlength='500' placeholder-style="color:#5F5F5F;" bindinput='limitWord' value="{{content}}"
placeholder='请输入您的意见(最多可以输入500字)'  class="input_area"  style="width: 610rpx; height: 47vh;
display: block; box-sizing: border-box; left: 0rpx; top: 0rpx">
</textarea>
```

图 3-30 textarea 代码展示

当用户尚未输入内容时，我们可以通过 placeholder 来设置提示，在这我们设置的就是内容提示和字数上限提示。效果如图 3-31 所示。Textarea 的优势是，后台可以动态地获取用户输入的内容，便于后续的处理操作，如在翻译中，用户输入的内容就会被送至服务器进行翻译。

图 3-31 textarea 效果展示

3、picker 的使用

Picker 的作用和他的名字意思一样，本质上就是一个选择器，在“我的”页

面中的年级选择运用到了这个元素，代码展示如图 3-32 所示。

```
<picker bindchange="PickerChange" value="{{index}}" range="{{picker}}" class="mistakes2">年级
    <text wx:if="{{!grade}}" class="grade">请选择你的年级</text>
    <text wx:else class="grade">{{grade}}</text>
</picker>
```

图 3-32 picker 代码展示

在这里，为了方便用户知道这个选择器的作用，我们在用户未选择的情况下设置了提示“请选择你的年级”。在用户点击选择器是，就会跳出我们在 js 文件里已经预设好的选项，用户可以根据实际情况进行选择，点击确定后，用户的选择结果也会立刻同步到后台，我们也因此可以同步将用户的年级选择渲染到前端界面上，图 3-33、图 3-34 就是选择前后的效果展示。

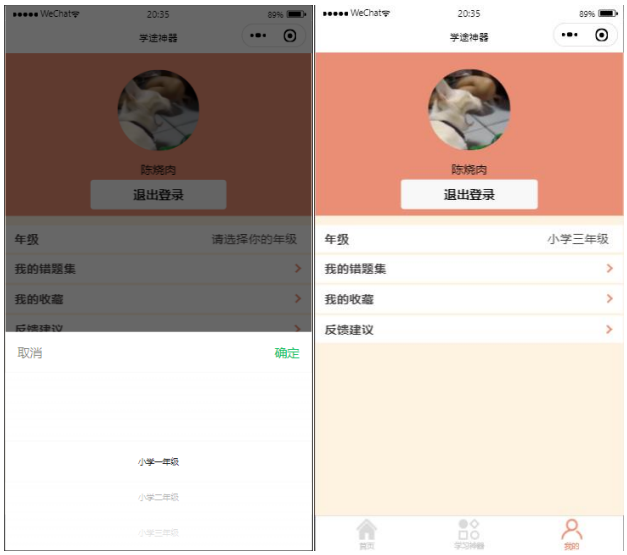

图 3-33、图 3-34 picker 效果展示

3.6.4 健壮性

针对实际情况中非常多的突发情况，我们都增加了相应的提示，这些提示的本质上都是 js 文件中的判断逻辑，我们在这里就不详细叙述判断的方法，只展示判断的效果。图 3-35 展示的是用户第一次登录尚未选择题库就点“继续刷题”的提示；图 3-36 展示的是用户提交反馈意见为空的提示；图 3-37 展示的是还未登录就点击我的错题集/收藏的提示。

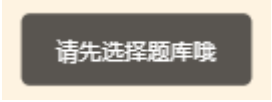

图 3-35 未选择题库的提示

图 3-36 输入为空的提示

图 3-37 未登录的提示

3.6.5 协同性

虽然我们在分工时会分前端与后端工作，但实际上很多功能我们是需要实现前后端的协同的，小程序中的很多地方都体现了这一点，比如在 3.6.3 中我们展示的 picker 就是前后端协同的一个例子。在协同性这一部分，我们重点介绍输入框中已输入字符数/输入上限的动态显示逻辑，因为这一部分并没有借助非常复杂的后端功能，只是在 js 中增加一个函数而已。

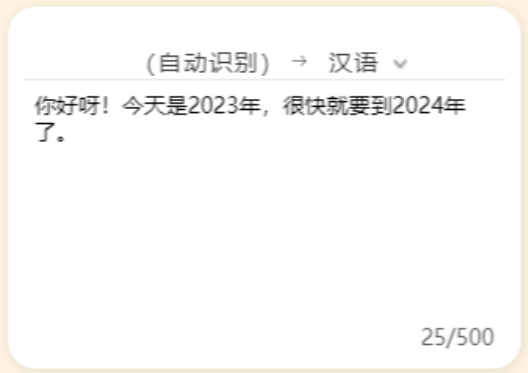

图 3-38 动态统计字数效果展示

我们实现的效果如图 3-38 所示。为了动态显示字数的变化，我们在 wxml 文件中就用了{{变量名}}的方式来放置变量。在微信开发者工具中，所有随着后端数据变化而动态渲染至前端的数据都要用双“{”框起来。因此在 wxml 文件中，

我们采用了如图 3-39 所示的代码来容纳当前字符数与上限数。

```
<textarea maxlength='500' placeholder-style="color:#5F5F5F;" bindinput='limitWord' value="{{content}}"
placeholder='请输入需要翻译的文本(最多可以输入500字)'  class="input_area"  style="width: 610rpx; height:
25vh; display: block; box-sizing: border-box; left: 0rpx; top: 0rpx">
</textarea>
<view style="width: 610rpx; height: 3vh; display: block; box-sizing: border-box">
  <text style="float: right; color:#5f5f5f;">{{currentWord}}/{{maxWord}}</text>
</view>
```

图 3-39 实现动态统计字数的 wxml 代码

我们在输入框架 textarea 中，绑定一个随用户输入内容而反复调用的函数 limitWord，每当用户输入的内容有所改变，就调用函数计算出当前字符串的长度，随后利用 setData 将改变后的变量渲染至前端页面。limitWord 函数的内容如图 3-40 所示。

```
limitWord:function(e){
  var that = this;
  var value = e.detail.value;
  //解析字符串长度转换成整数。
  var wordLength = parseInt(value.length);
  if (that.data.maxWord < wordLength) {
    return ;
  }
  that.setData({
    currentWord: wordLength,
    input: value
  });
},
```

图 3-40 统计字数 limitWord 的 js 代码

诸如此类可以体现前后端协同性的内容还有很多，比如同样在翻译页面的选择翻译目标语言、选择识图的传图方式等等。前端与后端的协同让我们的小程序功能并不割裂，既能带给用户很好的体验，也能减轻后端的工作。

4 实现与测试

4.1 实现环境与代码管理

4.1.1 实验环境

在 windows10/11 系统下，使用微信小程序开发工具进行开发，云端数据库采用微信开发者提供的云开发工具，后端选取腾讯云的学生轻量化服务器，型号选取 WordPress，架构选择轻量化 flask 框架，并把服务器的一个终端作为后端进行架设。

4.1.2 代码管理

1、微信开发者协同代码工作平台

第一个是采用微信开发者提供的协管代码工作平台进行管理。我们利用微信小程序开发工具中的协同代码开发功能。该工具允许我们在版本管理中推送和拉取代码，实现团队成员之间的协同开发。作为前端负责人之一的两位同学可以将各自完成的界面推送到微信提供的代码管理平台，并进行合并操作。另一位同学可以轻松地拉取并将代码合并到自己的项目中，实现了两人工作的高效合并。代码管理界面和成员管理界面提供了清晰直观的管理和协作操作，为团队合作提供了便利。管理成员界面如图 4-1 所示。

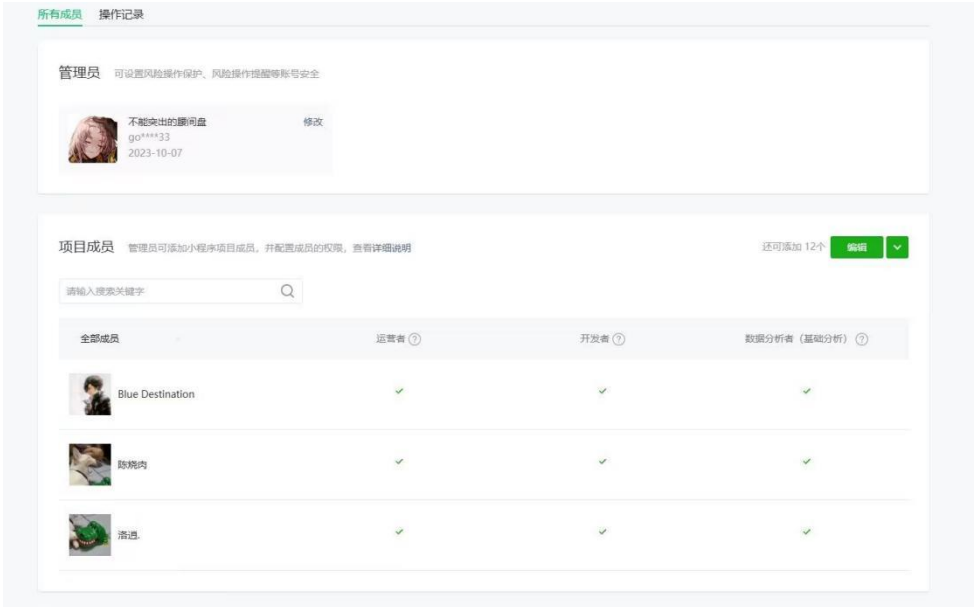

图 4-1 管理成员列表

2、gitee协同开发管理

另一个是采用 gitee 协同开发管理。可以用于代码托管和团队协作。在本次协同开发中，我们主要以微信开发者提供的代码托管平台为主，而将 gitee 作为辅助。通过 gitee，我们保存了代码并能够向外展示，增强了代码的可视化和分享性。在 git 代码托管平台中，文件列表清晰列示了项目的结构和内容，为团队成员提供了便捷的查看和管理途径。gitee 平台相关情况如图 4-2、图 4-3 所示。

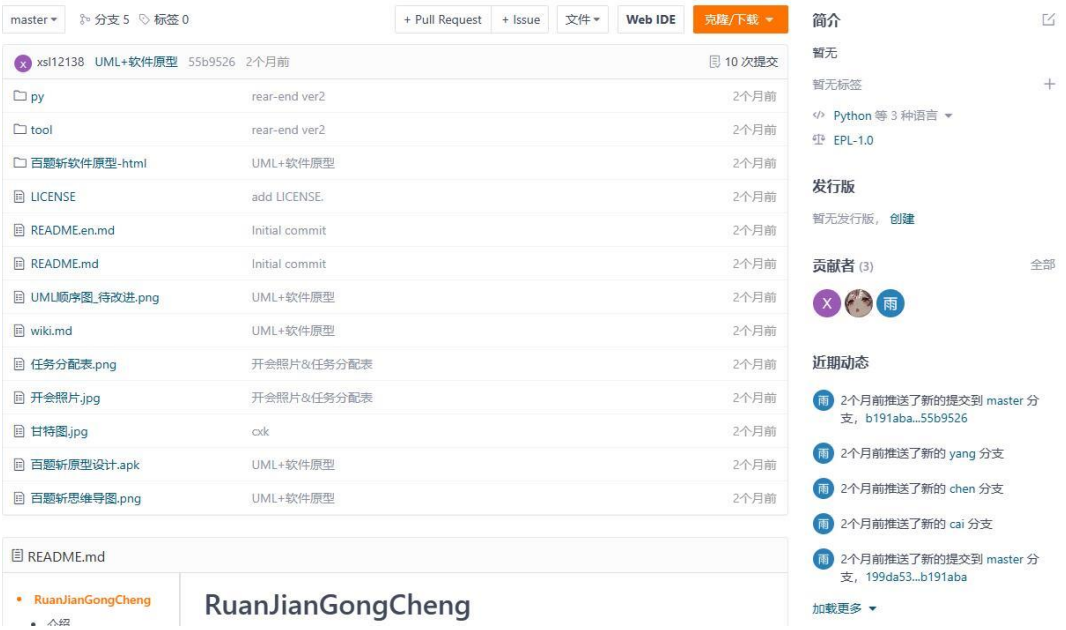

图 4-2 gitee 平台文件列表

图 4-3 gitee 平台管理图

4.1.3 小程序发布管理

通过微信开发者工具上传代码后，可以在微信小程序开发官网进行版本管理、提交审核等功能，并且后续可以不断修改 bug 并重新发布。经多轮测试，我们共提交了若干版本如图 4-4 所示。其中用户测试版本为 1.0.3。我们原本打算将小程序正式发布上线，但是最后由于小程序使用了 AI 相关技术的原因未能如愿。

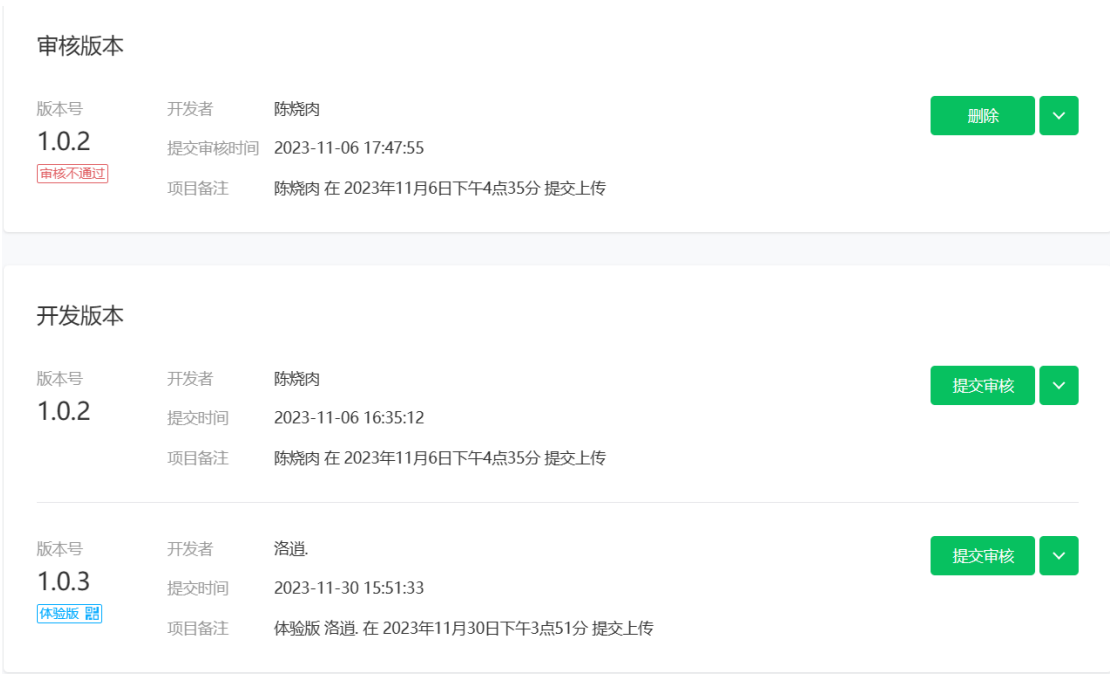

图 4-4 版本管理

4.2 关键函数说明

4.2.1 页面逻辑处理函数

由于页面众多，且每一页面所调用的函数多而不同，故我们只选取部分具有代表性的函数进行说明。

做题页面

1、radiochange()

用户点击响应函数，一旦用户选择选项，isSelect 置 1，将用于之后逻辑处理；用户选择的答案保存在 userSelect，用于用户界面做对应的显示，以及进行是否做对题目的判断。

2、submit()

用户点击提交后的反馈函数，首先若用户此时未选择选项（通过 isSelect 来判断），函数会作出“你还没选择选项”的提示，并直接 return；若用户已经选择了选项，则判断与正确答案是否一致，若不一致，根据此道题目的信息（即保存在数据库中的这道题目的所属科目）存入对应的错题集，此外，存入错题集之前还会判断一遍是否已经加入错题集，避免加入重复题目。

3、judge_shoucangstatus()

在做题界面，每道题目右上角会有一个收藏的图标，而已收藏和未收藏的题目对于该图标的显示是不同的。此函数的功能便是判断当前题目是否已经加入了收藏夹，使用户界面的收藏图标呈现出不同的样式。

4、shoucang()

收藏功能对应的函数，当一用户点击了收藏图标后，先判断用户是否登录，未登录会跳出“请先登录”的提示，否则，若为未收藏状态（在进入此函数之前已经用 judge_shoucangstatus 函数更新了相应的判断变量），则访问数据库，将题目 id 至收藏夹；若为已收藏状态，则从收藏夹中把此题目 id 去除。

5、next()

切换到下一题的函数，每次切换需要判断是否已经为最后一题，如果是，那么切换后会跳转到结算页面，并更新用户的做题记录。

6、delete()

此函数仅在从错题集进入的做题页面的 js 文件有写（为了实现不同功能，我们的小程序共有三个做题页面，分别是从题库/收藏夹/错题集进入）。用户点击右上角的垃圾桶图标后，小程序先跳出提示“确定将本题移出错题本吗？”若用户点击了确定，访问云端数据库，删除错题集中的这道题目，随后更新页面显示错题集中的下一道题，同时页面左上角总题数显示也进行对应的改变。这里还要对删除的是否是错题集中的最后一道题进行特判，如果是，则删除后直接跳到结算页面。

“我的”页面

1、login()

登录用函数，首先调用云端的 getopenid 函数获取用户的 id（具有唯一性），之后用 wx.getUserProfile 函数获取用户名、用户头像等信息，随后在云端数据库创建与用户有关的各项记录，包括个人信息、收藏夹、错题集、AI 对话记录（注意，如果用户不是首次登录，将不会再次创建），之后再次获取用户的收藏夹，以便进入收藏夹刷题时使用。

2、logout()

退出登录函数，重置所有变量

3、PickerChange()

选择年级函数，因为我们的选择年级功能是使用 picker 组件实现的，因此起了这个名字。函数的功能是根据用户的选择将用户标记为对应的年级，即在云端数据库的 user_information 集合中将该用户记录中 grade 项设置为所选年级。

4、collections()/errorlist()/feedback()

跳转函数，跳转收藏夹/错题集/反馈页面

4.2.2 前后端接口及后端函数

1、wx.vrequest()

带普通文本 http 的请求，实现对 AI 对话以及自然语言文本翻译的请求。

2、wx.urequest()

处理用户输入的图片，用 base64 的库转成 base64 编码，然后将此种形态的图片实现 http 请求。

3、chat_with_AI()

后端实现对路由报文的解析，获取对话内容 word 和用户 uid，通过 uid 实现对唯一 Session 的创建、获取、维护、更新，然后调用 chat_with_gpt()来实现和远程 GPT 模型的交互。

4、translate_Youdao()

后端实现对路由报文的解析，获取翻译内容 word 和用户 uid，然后通过对 TR 翻译器实体的类函数调用来实现翻译。

5、upload_trans（）

后端实现对路由报文的解析，获取 base64 编码格式的 image_data，然后通过对 TPR 翻译器实体的类函数调用来实现翻译。

还有部分函数在前文已经提到过，这里不再赘述。

4.3 测试计划和测试用例

1、本地测试

我们通过本地的函数的输出来检查数据是否成功的存进数据库或者成功的把数据从数据库中读出。

（1）登录功能

检查是否成功获得用户的基本信息如用户 id、昵称等，如图 4-5 所示。

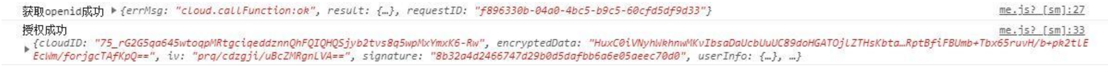

图 4-5 检查是否成功获得用户的基本信息

检查是否成功创建用户的收藏夹、错题集等信息，如图 4-6 所示。

图 4-6 检查是否成功创建用户的收藏夹、错题集等信息

（2）做题界面

检查用户选项，选择选项提交答案后，能否判断用户答题正误，如果用户做错题是否写进错题集，如图 4-7 所示。

图 4-7 检查用户选项

（3）我的错题集界面

检查是否成功删除错题集数据库中的对应题目，如图 4-8 所示。

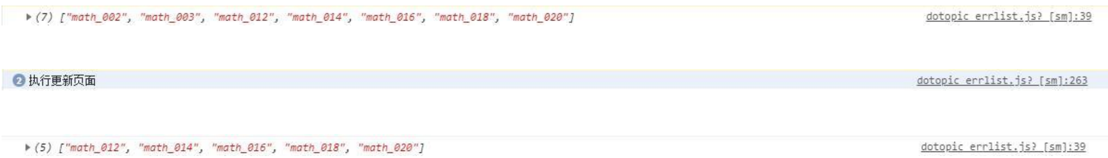

图 4-8 检查是否成功删除错题集题目

（4）我的收藏界面

检查收藏数据库中是否成功收藏对应题目，如图 4-9 所示。

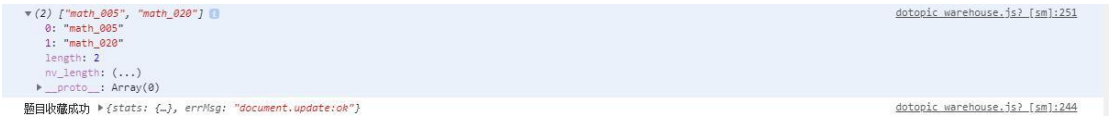

图 4-9 检查是否成功收藏题目

（5）打卡功能

检查是否成功打卡，观察打卡后数据库变化，如图 4-10 所示。

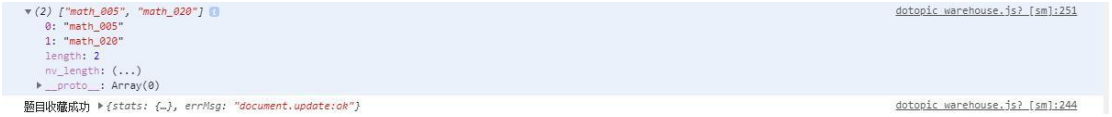

图 4-10 检查是否成功打卡

（6）学习神器界面

检查是否成功实现文字识别功能，如图 4-11 所示。

图 4-11 检查是否成功实现文字识别功能

检查是否实现智能翻译功能，是否能实现不同语言的翻译，如图 4-12 所示。

图 4-12 检查是否成功实现智能翻译功能

检查是否成功实现 AI 问答功能，如图 4-13 所示。

图 4-13 检查是否成功实现 AI 问答功能

2、整体的调试

在整个微信小程序的开发过程中，我们通过微信小程序的仿真器进行了整体的调试。仿真器为我们提供了实时的效果展示，使得在写前端代码时能够直观地观察界面的变化，从而更加高效地进行调整，确保最终获得一个美观且用户友好的前端界面，初始的做题界面和修改后的做题界面如图 4-14、图 4-15 所示。

图 4-14、图 4-15 初始做题界面(左)和修改后做题界面(右)

我们还实现了复杂的扩展功能，都包含在学习神器界面，以下是文字识别，智能翻译和 AI 问答的界面，如图 4-16 至图 4-19 所示。

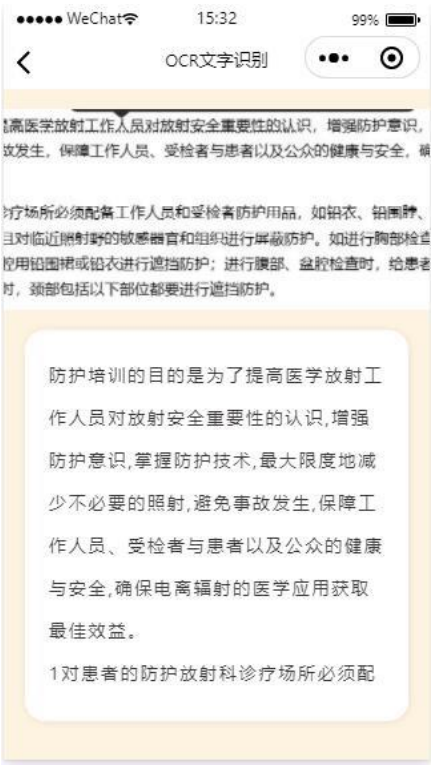

图 4-16 文字识别界面

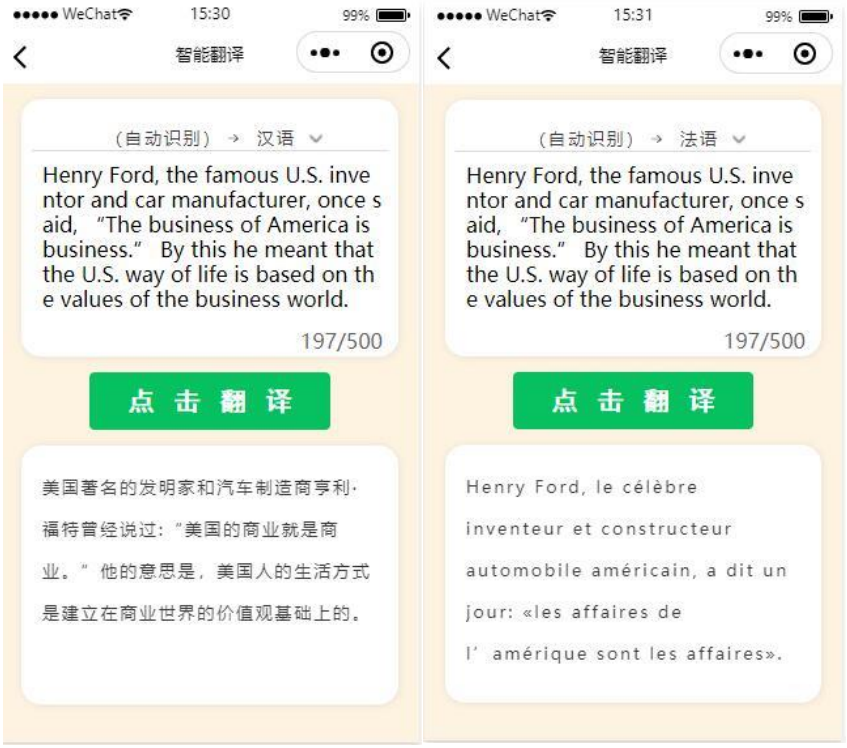

图 4-17、图 4-18 智能翻译界面

图 4-19 AI 问答界面

3、管理员真机测试

在微信开发者工具中，开发者（管理员）可以进行真机调试，一开始我们发现在不同机型下，前端界面会发生变化，比如在较大的机型下，前端界面会出现右边有白边的问题，在较小的机型下，前端界面会出现显示不全的问题，后俩发现是在编写 wxss 文档时距离单位错误所致，导致界面不能随着机型大小的变化而变化。之后我们进行了修改，现在版本的小程序已经可以根据界面内容显示的量与机型而进行变化，具有良好的适应性。

4.4 结果分析

用户登录与退出：能正常实现，用户登录后可以获取该用户的 id、微信昵称、头像等信息并能上传至数据库，退出后信息保留。

打卡：能正常实现，点击“打卡”按钮后完成打卡，用户界面有相应变化。

题集的选择与刷题界面的显示：进入“海量题库”界面可以正常选择年级和科目，刷题界面能够正常显示题目和选项，点击选项并点击提交答案按钮答案可以正常保存，显示做题正误情况，此时点击查看题解可以正确显示对应的题目解

答，点击返回可以返回题目正误情况界面，点击下一题可以继续答题，做完一套题后点击接着刷题可以跳转到选择题目类型的界面，点击返回首页可以跳转到首页。

错题记录与删除：初始状态下或未登录状态下正确显示“没有错题”。刷题过程中能够正确记录用户的错题和错误选项到数据库，并在错题集界面的不同科目分类下正确显示。点击删除后错题可以正常从数据库中删除。

收藏与删除：初始状态下或未登录状态下正确显示“没有收藏的题目”。刷题过程中点击收藏可以正常保存到数据库。并在收藏界面中正确显示。再次点击收藏按钮可以正常从数据库删除，取消收藏。

文字识别：学习神器页面，点击“文字识别”按钮可以正确弹出选择按钮，选择通过“相机”或“图库”上传图片到后端服务器，如果是在电脑上仿真测试则默认从图库选择图片。上传后可以正常实现文字识别功能，将识别结果显示在界面上。

智能翻译：学习神器页面，点击“智能翻译”按钮可以正确跳转到翻译页面，能够正确自动识别输入文字的语言，并可自由选择 12 种目标翻译语言，点击“点击翻译”按钮可以正确进行翻译，翻译结果显示在界面上。

AI 问答：学习神器页面，点击“AI 问答”按钮可以正确跳转到问答页面，可在输入框自由输入问题，点击“发送”按钮将问题传入后端服务器，最终回答结果可以正确显示在问答界面。

5 总结

5.1 用户反馈

我们使用微信开发者工具上传了 1.0.3 版本的小程序后，将其设置为体验版，供用户体验（因为 AI 相关的技术原因，小程序无法正式发布上线，因此只能采取此方式进行用户反馈的收集），相关反馈如图 5-1 至图 5-4 所示。

图 5-1 至图 5-4 用户反馈（部分）

综合用户反馈，我们了解到在用户视角小程序有以下优点：

1、AI 功能非常有趣，并且功能比较强大，对于各种问题都能有所回复；

2、整体使用非常流畅，界面优美，小程序与不同机型的适应性也很好；

3、翻译功能和文字识别功能十分贴合用户需求；

4、收藏、错题、快捷的刷题入口等功能，都十分贴合小程序受众的需求。

同时，我们也注意到了以下缺点：

1、答题界面，选项按键只能点击特定区域进行选择，如果将整个选项都变成有效点击区域会更好；

2、打卡目前为手动打卡不太方便，后续可以考虑换成做过题目后自动打卡；

3、主界面“海量题库”可能造成歧义导致用户在初次使用时找不到答题入口，因此可以改成“开始刷题”等描述；

4、文字识别功能，对于没有文字的图片会不作任何提示，虽然不会造成程序崩溃，但是会让用户一头雾水，后续可以加入相应提示。

后续我们也计划针对用户提出的建议，进一步修改与完善小程序。

5.2 全文总结

历时近两个月，小组成员基于微信开发者工具和腾讯云轻量化服务器完成了学途神器小程序的设计开发，主要工作如下：

（1）基于 NABCD 模型分析需求，确定了以中小学生为手中主体的刷题软件的项目目标。

（2）基于墨刀原型设计软件完成了以简洁美观实用为主要目标的 UI 设计。

（3）基于微信开发者工具和腾讯云轻量化服务器完成了完整的可交互程序，其中包含：

1、使用微信开发者工具自带的云数据库完成了数据模型创建，使用 json 文件格式的题集进行一键导入；

2、实现了学习神器的三样功能：调用有道云 OCR 接口的文本识别功能，支持 5mb 以内图片的印刷体文字识别；调用有道云文本翻译接口的智能翻译功能，支持包含英语、日语、法语、意大利语等近 20 种语言的互译；支持 GBT3.5Turbo-0613-16k 模型的 AI，实现用户与 AI 的在线交互。

3、实现了海量题库、继续刷题、收藏夹、错题集等一系列较为完整的功能，优化整体逻辑使交互更加流畅。

4、在墨刀原型与小程序初样的基础上进行美化，形成了简洁美观的主题风格，也使小程序健壮性进一步提高，面对不同机型与文字显示情况均能自适应调整界面，不会出现奇怪的显示情况。

（4）基于测验与反馈不断修改，申请了小程序发布上线，遗憾的是，由于AI 相关的原因最终未能成功发布。

6 体会

1、CS2107 cxk

分工：负责全过程文档整理、wiki 文档编写、分支项目集合、总体进度把控。负责软件功能设计、数据结构设计、云端数据库管理、云端数据库调用与可视化、应用整体逻辑优化等。

体会：本次软件工程的课设充满着趣味与挑战，工程量、持续时间比起其他任何一门课程的大作业都遥遥领先。我们的工程一开始开展地并不顺利，即使因为怕完不成任务而选了个（我们认为）相对简单的儿童学习 app 的选题，但我们小组没有任何一个人有关团队或个人开发应用的经验，导致我们一开始的目标不明确，分工也没法分。好在我们及时调整状态，在经过了一系列调研过后，我们完成了初步的构想（即主流的学习类 app 都有的刷题、收藏、错题等功能，以及我们的特色学习神器）。在实现途径方面，得益于微信小程序有较为完整的开发工具、开发手册等，我们便放弃了 app 的想法，开始专心于对小程序的研究。

本次团队任务中，我负责最多的是云端数据库部分，包括数据结构的设计、数据库调用与可视化等。想使用云端数据库并不难，导入数据、调用数据这些都是家常便饭，正因为这一部分较为简单，我放松了警惕，殊不知接下来仍有一场恶战要打。微信小程序的 js 文件几乎全都是异步调用，云端和云端是异步，云端和本地是异步，甚至有的时候，本地和本地都是异步，这对于写惯了 C++的我简直是一个噩耗。解决数据库调用的异步问题便成了第一道坎，我在翻阅了众多资料后又试了无数次后才终于找到了可行的办法（后来发现这个办法有的时候还是无能为力）。由于在调用数据库后，必须在 js 和 wxml 文件中都做相应的修改才能在用户界面看到变化，于是我自然而然地也学习了接口的工作。在看到界面上成功显示了数据库的内容时，心中不免有一些小小的成就感。

随后便是整个应用调用数据库部分的逻辑构建，让我印象最为深刻的是取消收藏和删除错题这两个功能，取消收藏的功能方面，我最后选择在做题界面使用一个局部变量暂时存储要取消收藏的题目 id，等做完题集或返回时统一删除，保证了页面的逻辑不至于出现混乱。而删除错题的功能方面，需要删除后立马显示下面的题目，并且还要对是否删除了最后一道题目进行特判……嗯，又是一场关

于调用数据库逻辑的恶战。

总之，这次的任务让我感到痛苦的同时，也为我带来了巨大的回报，小程序完成的那一刻，成就感是之前的课设中从未体会到的，比成就感更宝贵的便是团队开发的知识，相信下一次我能够做的更好！

2、CS2107 yy

分工：负责服务器搭建与维护，参与全过程核心技术开发，包括服务器函数部署与测试、后端整体需求设计、后端函数测试更新、接口设计、后端数据库维护等，负责学习神器三样功能的实现。

体会：本次软件工程的课设是一次非常难得的经历，它有非常大的空间来给我们发挥我们的独特创意。我们组的选题虽然是老师给定的几个初始选题之一，但是在我们的讨论和分析以及对市场的调研之后我们加入了非常多的额外内容，使得原先枯燥单一的初始选题内容焕然一新。我在本次小组课设的分工中主要负责后端服务器的搭建以及前后端交互的设计。平心而论，我时间花的最多是环境的配置，因为微信小程序的服务后端有非常严格的安全性要求，所以不能支持一些没有安全性声明的域名/IP 地址的请求和响应。我主要是花时间在腾讯云购买配置服务器，配置公网 IP，购买、配置与解析域名，在 wordpress 系统上面配置 Ngnix 转发服务，配置域名的 SSL 证书，在湖北省工信部的网上服务平台申请 ICP 备案。这一套流程的不出差错才能实现微信小程序的安全性要求，而我在并行开发的过程中还特地设计了一套云函数代理转发来跳过它的安全性检查。

然后关于后端的具体设计其实我没有花费特别多的时间，因为我平时就是那种非常喜欢钻研一些有趣的 API 服务的人，自己也基于 GoCQHTTP 去搭建过一些机器人和小程序，所以对后端的实现和搭建有一些自己的心得和经验，我在我的历史仓库中找寻了最契合本次主题的后端源码，进行必要的删改之后再推出我们的功能，并再一次启发我们的前端设计的同学，也就是说我们的开发设计是在不断的迭代过程中的。本次的课设给了我一个把爱好转化成实际成果的机会，我十分珍视，是我一段美妙的经历。

3、CS2110 csr

分工：完成原型系统设计，完成大部分的前端页面设计、全部页面的梅花工

作，设计图标等。

体会：本次课程设计是一次很难得的项目体验，不仅需要小组成员紧密协作，还需要很长一段时间的投入。

在最开始时，我们很快地确定了选择老师所给的题目之一“儿童学习”，但是仅仅实现这个功能是很单薄的。因此在小组成员讨论后，我们一致决定加入了错题、收藏、AI 对话、OCR、翻译等等拓展功能。由于我们相关的经历都非常少，所以对于我们需要做什么、如何分工，我们的感觉是迷茫的。在这时我们就选择了先学习，学习简单的小程序是如何完成的。在初步掌握了一个小程序搭建的过程后，我选择了自己比较擅长的“视觉”部分，即根据小组同学的想法将功能设计成原型设计，再根据实际情况将原型转化为小程序页面，最后再一步步美化、健壮。

其实前端的工作难度并不是很高，但我并不认为自己在小组中扮演的是一个或有或无的角色。我们的小程序页面不仅仅实现了功能性，更在此基础上变得美观，也更多地考虑了用户的需求和体验；我们还针对不同的机型调整了小程序的适应性，选择了相对长度单位，通过计算确定页面中每一个组件应当处在的相对位置；我还意识到，前端工作与后端工作并不是分离的，一个优秀的前端设计应当和后端的部分代码很好地结合起来，我觉得在翻译页面的输入字数就是一个很好的体现。

这次项目中，对于我挑战最大的还是 AI 界面，不断更新的交互、特殊情况繁多的内容，经过了许多遍的调试才最终变成了和实际聊天界面非常相仿的结果。也很感谢小组的其他同学，虽然我们分别负责不同的内容，但是仍然会就我当时遇到的困难提出一些解决思路，也正是小组成员不断的沟通、讨论、改进，才能让我们的小程序最终呈现出这样完成度高的作品。我也会以这次软件工程课程为一个起点，不断探索新的领域。

4、CS2107 cxy

分工：前期主要负责设计 UML 图，后期主要实现刷题界面前端代码；“刷题”界面、“收藏”界面、“错题集”界面设计。

体会： 在软件工程课程的学习中，我们团队选择了学习类微信小程序开发

作为选题。整个过程充满了探索、困惑，以及最终的成就感。回顾这段经历，我深刻体会到了实践的力量以及团队协作的重要性。

初期，我们确定了开发主题为刷题小程序，并完成了 UML 图和原型设计。然而，一开始进入实际的开发环节时，我们感到一片陌生。小程序开发、工具使用、开发语言等领域，几乎是一个全新的世界。在这个阶段，我们经历了挣扎，尝试理解并最终接受这些新知识。

对于我个人而言，之前没有小程序开发经验，对开发流程一无所知。面对庞大的任务，我们团队通过讨论和相互鼓励，开始分工学习。我首先尽可能快地完成了 UML 图的设计，之后就立刻通过 B 站等平台，开始学习小程序开发的基本流程和语言规范，我们逐渐找到了方向。即便在实现“学途神器”小程序雏形时遇到了问题，我们也通过查阅文档和不断调试成功解决了困扰我们的难题。

分工明确后，我们开始逐步完善小程序，学习 CSS 和微信云开发函数，实现了界面美化和数据库的功能。我们还加入了一系列的“学习神器”拓展功能，如文字识别，智能翻译，AI 问答等，使小程序更加惊艳。这其中我主要负责了刷题界面前端代码，包括“刷题”界面、“收藏”界面、“错题集”界面设计。

这次项目让我深刻领悟到动手实践的不可替代性。理解教程视频中的实现过程远远不如亲自动手实现来得深刻。在学习 CSS 时，只学习网上的教程很难有所收获，只有亲自动手试一试，才能更好地理解样式的效果，学习过程也变得更加轻松。

另一方面，团队协作在整个软件开发过程中起到了至关重要的作用。在团队中，每个成员都分担了不同的任务和学习内容，通过合作克服了项目初期的迷茫感。个人的努力与团队的协同使得最终的成果更加圆满。这次经历不仅是对软件工程复杂性的深刻认识，也是对团队协作不可或缺性的感悟。

总的来说，这次微信小程序开发的经历不仅丰富了我的实践经验，也加深了对团队协作和动手实践的理解。在迈向未来的软件工程之路上，这些体会将成为我不断进步的动力和指引。