

HNU个人项目互评

一、项目要求

用户:

- 小学、初中和高中数学老师。

功能:

- 命令行输入用户名和密码，两者之间用空格隔开（程序预设小学、初中和高中各三个账号，具体见附表），如果用户名和密码都正确，将根据账户类型显示“当前选择为XX出题”，XX为小学、初中和高中三个选项中的一个。否则提示“请输入正确的用户名、密码”，重新输入用户名、密码；
- 登录后，系统提示“准备生成XX数学题目，请输入生成题目数量（输入-1将退出当前用户，重新登录）：”，XX为小学、初中和高中三个选项中的一个，用户输入所需出的卷子的题目数量，系统默认将根据账号类型进行出题。每道题目的操作数在1-5个之间，操作数取值范围为1-100；
- 题目数量的有效输入范围是“10-30”（含10,30，或-1退出登录），程序根据输入的题目数量生成符合小学、初中和高中难度的题目的卷子（具体要求见附表）。同一个老师的卷子中的题目不能与以前的已生成的卷子中的题目重复（以指定文件夹下存在的文件为准，见5）；
- 在登录状态下，如果用户需要切换类型选项，命令行输入“切换为XX”，XX为小学、初中和高中三个选项中的一个，输入项不符合要求时，程序控制台提示“请输入小学、初中和高中三个选项中的一个”；输入正确后，显示“系统提示“准备生成XX数学题目，请输入生成题目数量”，用户输入所需出的卷子的题目数量，系统新设置的类型进行出题；
- 生成的题目将以“年-月-日-时-分-秒.txt”的形式保存，每个账号一个文件夹。每道题目有题号，每题之间空一行；

附表-1：账户、密码

账户类型	账户	密码	备注
小学	张三 1	123	
	张三 2	123	
	张三 3	123	
初中	李四 1	123	
	李四 2	123	
	李四 3	123	
高中	王五 1	123	
	王五 2	123	
	王五 3	123	

附表-2：小学、初中、高中题目难度要求

	小学	初中	高中	
难度要求	+, -, *, /	平方，开根号	sin, cos, tan	
备注	只能有 +, -, *, / 和 ()	题目中至少有一个平方或开根号的运算符	题目中至少有一个 sin, cos 或 tan 的运算符	

二、黑盒测试

黑盒测试部分 **不关心代码的组织框架**，只考虑**功能是否正常**等方面，并且将尽可能地全面测试程序。

0. 启动程序

这个项目前后端分离。前端使用命令行显示界面，requests 库发送 http 请求；后端使用 python 的后端框架 django 作为服务器接受 http 请求对数据库数据进行处理，数据库使用 sqlite。

开启 django 的 server

```
python manage.py runserver 0.0.0.0:80
```

开启 CLI 的前端界面

```
python Application.py
```

1. 登录部分

- (1) 运行程序后会显示“请输入您的账号”、“请输入您的密码”字样，符合项目要求，使用者也能清楚知道要输入的是什么。

```
*****
* 欢迎使用中小学数学卷子自动生成程序 *
*****

请输入您的账号: 
```

```
*****
* 欢迎使用中小学数学卷子自动生成程序 *
*****

请输入您的账号: 张三1
请输入您的密码: 123
```

(2) 登录成功后会提示"登录成功", 并输出用户的名称。

```
登录成功! 欢迎, 张三1
-----

登录成功! 当前选择为小学出题。

主菜单:
1. 生成题目
2. 切换用户类型
3. 退出
请选择操作 (1/2/3): 
```

(3) 登录失败后, 同样会提示"登录失败"

```
-----

*****
* 欢迎使用中小学数学卷子自动生成程序 *
*****

登录失败, 请输入正确的用户名和密码
请输入您的账号: 
```

2. 主页面控制

(1) 界面显示当前账户的"用户名"和"账号类型", 当前根据附表, 张三 1 的账户类型为小学, 显示正确。

```
登录成功! 欢迎, 张三1
-----

登录成功! 当前选择为小学出题。

主菜单:
1. 生成题目
2. 切换用户类型
3. 退出
请选择操作 (1/2/3): 
```

(2) 输入"1", 会提示要输入的题目数量

当输入不是 10 ~ 30 之间的整数时候, 会提示。

```
主菜单:
1. 生成题目
2. 切换用户类型
3. 退出
请选择操作 (1/2/3): 1
-----
请输入题目数量 (输入 '-1' 返回): 
```

```
-----
请输入题目数量 (输入 '-1' 返回): 西安市
题目数量必须在10 ~ 30之间的整数, 请重新输入。
-----
```

```
-----
请输入题目数里 (输入 '-1' 返回): 2
题目数里必须在10 ~ 30之间的整数, 请重新输入。
-----

请输入题目数里 (输入 '-1' 返回): 35
题目数里必须在10 ~ 30之间的整数, 请重新输入。
-----
```

输入正确时，输出题目，并保存到对应文件中

```
-----
请输入题目数里 (输入 '-1' 返回): 10
题目生成成功:
1. 6 * 71 + ( 8 + 66 )

2. 20 / 82 * ( 83 + 83 ) * 47

3. 74 * 57 * 49 * 3

4. 88 / 52 * 24 / ( 32 - 31 )

5. 11 * 60 * ( 64 + 1 )

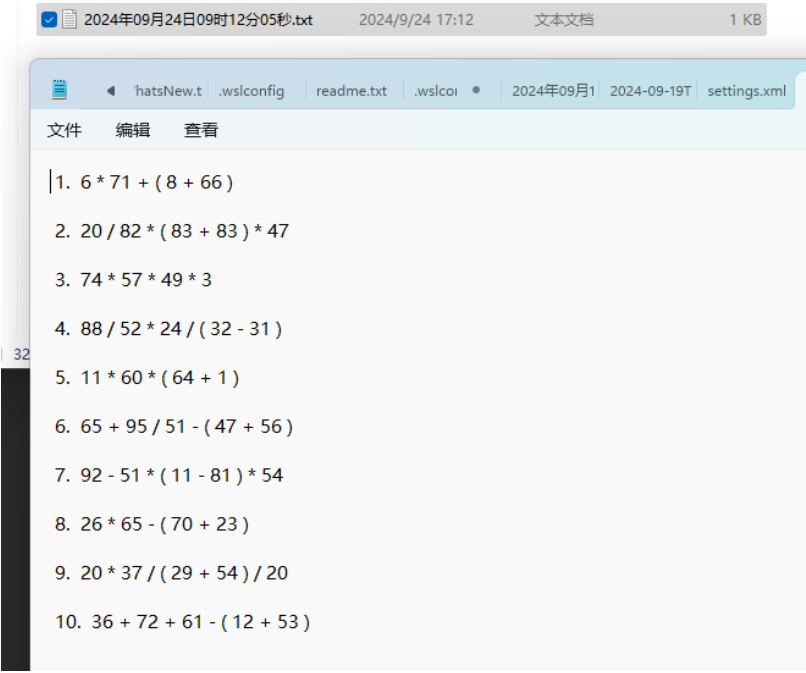
6. 65 + 95 / 51 - ( 47 + 56 )

7. 92 - 51 * ( 11 - 81 ) * 54

8. 26 * 65 - ( 70 + 23 )

9. 20 * 37 / ( 29 + 54 ) / 20

10. 36 + 72 + 61 - ( 12 + 53 )
```



输入"-1"时不能正常返回，原因是 `input()` 默认接受字符串类型，需要 `int(input())` 转换成 `int` 类型

```
-----
请输入题目数里 (输入 '-1' 返回): -1
题目数里必须在10 ~ 30之间的整数, 请重新输入。
-----

请输入题目数里 (输入 '-1' 返回): 1
题目数里必须在10 ~ 30之间的整数, 请重新输入。
-----
```

```
def generate_questions(self):
    """生成题目"""
    while True:
        clear_screen()
        count = input("请输入题目数量 (输入'-1'返回): ")

        if count == -1:
            # if is_integer(count) and int(count) == -1:
            return
        elif is_integer(count) and 10 <= int(count) <= 30:
            break
        else:
            print("题目数量必须在10 ~ 30之间的整数，请重新输入。")

    self.auth.apply_generate_questions(self.auth.user_info['user_id'], count, self.auth.us
```

更改后：

```
def generate_questions(self):
    """生成题目"""
    while True:
        clear_screen()
        count = input("请输入题目数量 (输入'-1'返回): ")

        if is_integer(count) and int(count) == -1:
            return
        elif is_integer(count) and 10 <= int(count) <= 30:
            break
        else:
            print("题目数量必须在10 ~ 30之间的整数，请重新输入。")

    self.auth.apply_generate_questions(self.auth.user_info['user_id'], count, self.auth
```

(3) 输入"2"，会提示要切换的用户类型，输入新类型后回到控制主页面，可以看见当前用户类型已经是初中

```
-----
登录成功！当前选择为小学出题。

主菜单：
1. 生成题目
2. 切换用户类型
3. 退出
请选择操作 (1/2/3): 2
-----
请输入新的用户类型 【小学/初中/高中】 (输入'-1'返回): 初中
用户类型已切换
-----
登录成功！当前选择为初中出题。

主菜单：
1. 生成题目
2. 切换用户类型
3. 退出
请选择操作 (1/2/3):
```

当输入不符合规格时，会提示"请输入"小学", "初中", "高中"中的一个"

```
-----
请输入新的用户类型 【小学/初中/高中】 (输入'-1'返回): 显示
请输入"小学", "初中", "高中"中的一个
-----
请输入新的用户类型 【小学/初中/高中】 (输入'-1'返回):
```

输入"-1"成功退出

```
-----
请输入新的用户类型 【小学/初中/高中】 (输入'-1'返回): -1
-----
登录成功！当前选择为初中出题。

主菜单：
1. 生成题目
2. 切换用户类型
3. 退出
请选择操作 (1/2/3):
```

(4) 账号类别不同，试卷难度不同

```
*****
* 欢迎使用中小学数学卷子自动生成程序 *
*****

请输入您的账号：张三1
请输入您的密码：123
登录成功！欢迎，张三1
-----

登录成功！当前选择为初中出题。

主菜单：
1. 生成题目
2. 切换用户类型
3. 退出
请选择操作 (1/2/3): 1
-----

请输入题目数 (输入 '-1' 返回): 10
题目生成成功：
1.  $4^2 - 98 - (6^2 - 22) * 4^2$ 

2.  $4^2 / 7^2 - (6^2 - \sqrt{16}) * 83$ 

3.  $6^2 / \sqrt{4} / 39 * 53 / \sqrt{9}$ 

4.  $5^2 * (\sqrt{81} - \sqrt{16}) * 70$ 

5.  $(7^2 - 5) / 78 * \sqrt{36}$ 

6.  $7^2 / (56 + 49) / \sqrt{49}$ 

7.  $6^2 * 29 - (97 - 2^2) * 6^2$ 

8.  $7^2 / 2^2 + (8^2 - 6^2) * 5^2$ 

9.  $(5^2 - 3^2) / \sqrt{64} / \sqrt{49} / \sqrt{100}$ 

10.  $6^2 + 93 / 24 - (8^2 - 11)$ 
```

```
-----

登录成功！当前选择为高中出题。

主菜单：
1. 生成题目
2. 切换用户类型
3. 退出
请选择操作 (1/2/3): 1
-----

请输入题目数 (输入 '-1' 返回): 10
题目生成成功：
1.  $\sin(60^\circ) / \sin(45^\circ) / (\tan(60^\circ) - 0)$ 

2.  $\sin(60^\circ) + (\sqrt{36} - 92) / 38$ 

3.  $\sin(60^\circ) / 15 / (\sin(45^\circ) + 28) / \cos(60^\circ)$ 

4.  $\sin(45^\circ) - (33 + \sqrt{100}) * 10^2 / \sin(45^\circ)$ 

5.  $\sin(30^\circ) / \tan(30^\circ) * \sin(60^\circ) * 61$ 

6.  $\sin(60^\circ) * 4^2 - 3 / (65 - \sin(60^\circ))$ 

7.  $\sin(45^\circ) + 4^2 * 9^2 / (\cos(45^\circ) - \tan(60^\circ))$ 

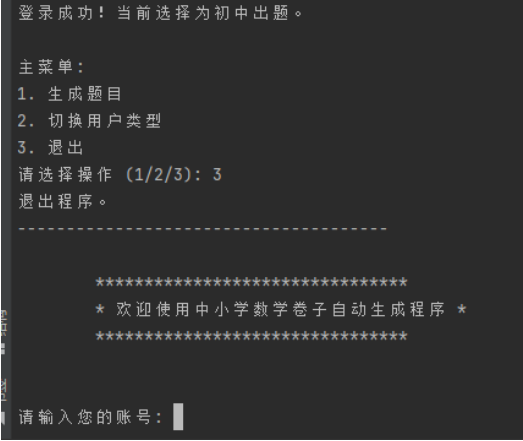
8.  $\sin(30^\circ) + \sqrt{64} - 5 - (\sqrt{16} - 54)$ 

9.  $\sin(30^\circ) + 37 * (13 - 24)$ 

10.  $\sin(60^\circ) * (\cos(30^\circ) - 70) / 6^2 / \tan(30^\circ)$ 
-----










登录成功！当前选择为高中出题。
```

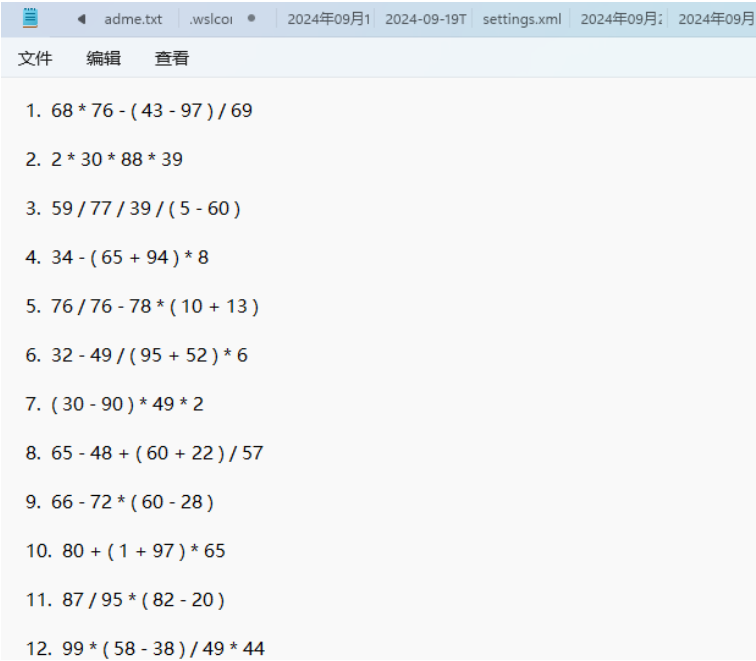
(5) 输入"3"，成功退出账号



3. 试卷部分

试卷存储符合规定

 1	2024/9/24 17:12	文件夹	
 2	2024/9/18 3:41	文件夹	
 3	2024/9/18 3:38	文件夹	
 5	2024/9/18 3:28	文件夹	
 7	2024/9/18 0:11	文件夹	
 2024年09月17日15时54分33秒.txt	2024/9/24 16:31	文本文档	1 KB
 2024年09月17日18时34分18秒.txt	2024/9/24 16:31	文本文档	1 KB
 2024年09月24日09时11分51秒.txt	2024/9/24 17:11	文本文档	1 KB
 2024年09月24日09时12分05秒.txt	2024/9/24 17:12	文本文档	1 KB



4. 测试结果：

程序基本上完成了项目要求，功能正常。但是由于项目作者提交之前可能没有全面测试程序，导致有个别比较容易解决的bug没有处理。但是优点明显：对于各种异常输入都有处理方法，界面美观清晰，提示信息完善，三角函数和平方根号很合理。

三、工程分析

0. 代码分析

Application.py

功能

这个文件是整个应用程序的入口，主要负责命令行界面的显示和与后端服务器的交互。它包含三个主要的类：

- Authenticator: 负责与后端通信, 功能包括验证登录、生成题目和切换用户类型。
- CLIApp: 负责命令行界面的显示, 调用Authenticator类与服务器交互。
- Application: 主程序类, 协调各个组件并运行主程序。

优点

- 结构清晰: 代码结构清晰, 职责分明。Authenticator类负责与后端通信, CLIApp类负责命令行界面, Application类负责协调各个组件。
- 注释详细: 每个方法都有详细的注释, 便于理解和维护。
- 异常处理: 在与后端通信时, 有异常处理机制, 能够捕获请求异常并打印错误信息。

缺点

1. 模块化不足: 所有类都在一个文件中, 导致文件较长, 不利于维护。可以将不同的类拆分到不同的文件中。
2. 硬编码: 一些URL和用户信息是硬编码的, 不够灵活。可以考虑将这些信息配置化。
3. 清屏功能: 清屏功能有bug, 不能使用os.system('cls')清除屏幕。可以考虑使用其他方法实现清屏。

accounts/tests.py

功能

这个文件包含对accounts模块中方法的测试代码, 主要测试登录和切换用户类型的功能。

优点

1. 测试覆盖全面: 测试了登录和切换用户类型的各种情况, 包括成功、失败和无效输入等。
2. 使用Django测试框架: 使用了Django自带的测试框架, 便于集成和运行测试。
3. 数据准备充分: 在每个测试方法运行之前, 使用setUp方法准备测试数据, 确保测试的独立性。

缺点

- 重复代码: 一些测试方法中有重复的代码, 可以考虑提取公共部分, 减少重复。
- 缺少更多边界测试: 虽然测试了基本的功能, 但缺少一些边界情况的测试, 比如极端输入等。

common/models.py

功能

这个文件定义了数据库模型, 包括用户表、文件表和题目表。

优点

1. 模型设计合理: 模型设计合理, 符合数据库设计规范。每个模型都有明确的字段和关系。
2. 使用Django ORM: 使用了Django的ORM, 便于数据库操作和查询。
 - 字段注释清晰: 每个字段都有详细的注释, 便于理解和维护。

缺点

1. 缺少更多字段验证: 虽然定义了字段类型, 但缺少更多的字段验证, 比如长度限制、格式验证等。
2. 缺少索引: 在一些常用查询字段上缺少索引, 可能会影响查询性能。

accounts/views.py

功能

这个文件定义了accounts模块中的视图函数, 主要处理登录和切换用户类型的请求。

优点

1. 功能实现简单: 每个视图函数都实现了特定的功能, 逻辑简单明了。
2. 使用JsonResponse: 使用了Django的JsonResponse返回JSON数据, 便于前端处理。
3. 异常处理: 在查询用户时, 有异常处理机制, 能够捕获用户不存在的异常并返回相应的错误信息。

缺点

1. 缺少序列化器: 没有使用Django的序列化器, 直接操作模型对象, 可能会导致代码冗长和不易维护。
2. 缺少更多验证: 在处理请求数据时, 缺少更多的验证, 比如字段类型、长度等。

questions/views.py

功能

这个文件定义了questions模块中的视图函数, 主要处理生成题目的请求。

优点

1. 功能实现完整: 实现了生成题目的完整流程, 包括生成题目、查重、写入文件和保存数据库。
2. 使用事务: 在写入文件和保存数据库时, 使用了事务, 确保数据的一致性和完整性。
3. 代码结构清晰: 代码结构清晰, 每个函数都有明确的职责和注释。

缺点

- 缺少序列化器：没有使用Django的序列化器，直接操作模型对象，可能会导致代码冗长和不易维护。
- 硬编码：一些文件路径是硬编码的，不够灵活。可以考虑将这些信息配置化。
- 异常处理不足：在生成题目和写入文件时，异常处理不足，可能会导致程序崩溃。

test_paper_generation/urls.py

功能

这个文件定义了Django项目的URL配置，将URL路由到相应的视图函数。

优点

1. 配置简单：URL配置简单明了，每个URL都路由到相应的视图函数。
2. 使用Django URL配置：使用了Django的URL配置，便于扩展和维护。

缺点

1. 缺少命名空间：没有使用命名空间，可能会导致URL冲突。可以考虑使用命名空间来区分不同模块的URL。
2. 缺少更多注释：缺少更多的注释，便于理解和维护。

utils/ProblemGenerator.py

功能

这个文件定义了生成数学题目的工具类，包括小学、初中和高中的题目生成器。

优点

1. 使用抽象类：使用了抽象类和继承，便于扩展和维护。
2. 功能实现完整：实现了生成不同难度题目的功能，逻辑清晰。
3. 代码结构清晰：代码结构清晰，每个类和方法都有明确的职责和注释。

缺点

1. 缺少更多测试：缺少更多的单元测试，确保生成题目的正确性和稳定性。
2. 硬编码：一些生成规则是硬编码的，不够灵活。可以考虑将这些规则配置化。

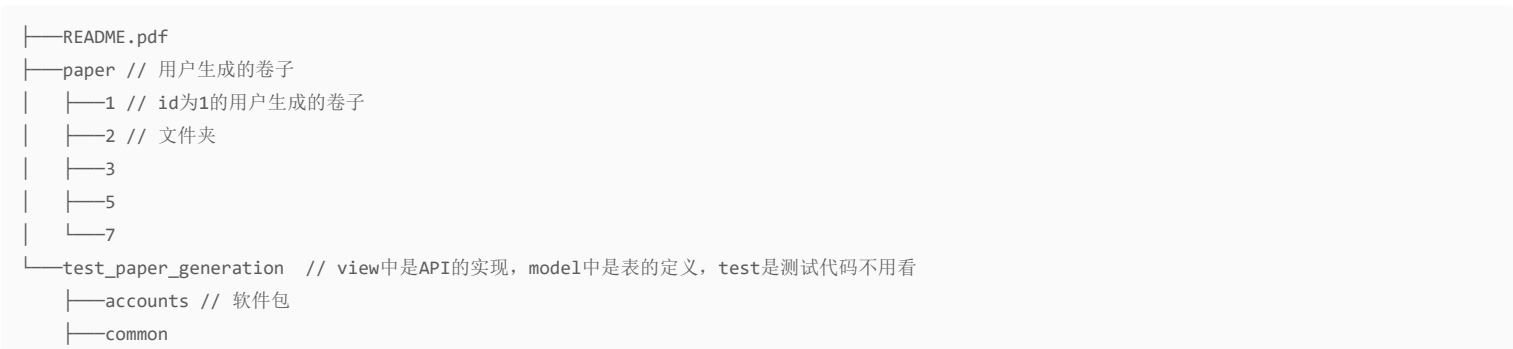
1. 查重功能的实现

该功能已经实现。这里题目是一个一个生成，每次生成一个题目就会把题目信息存入 Question 表中，下一道题就会查询表中是否有完全一样的题目，没有就丢弃。

```
# **查重** , 由于已经把之前生成的所有题目记录下来了，所以直接在Question表中查就可以
questions = []
cnt = int(count)
while cnt:
    e = generator.generate_expression()
    try: # 如果不存在才会存进列表
        Question.objects.get(content=e)
    except Question.DoesNotExist:
        questions.append(e)
        cnt = cnt - 1
# 写进paper/{user_id}/...
file_id, now_time = write2txt(user_id, questions)
# 保存进数据库
save_questions_2db(user_id, questions, file_id, user_type, now_time)

return JsonResponse({
    "status": 0,
    "message": "题目生成成功",
    "questions": questions
})
```

2. 组织架构




```
|—questions
|—test_paper_generation
|—utils
|—Application.py // 前端启动文件
|—manage.py      // 后端启动文件
|—db.sqlite3     // 数据库文件
```

- `\test_paper_generation\test_paper_generation` 是框架的配置软件包，重点在 `urls.py`，其中包含了路由和 API 接口的配置
- `\test_paper_generation\...` 是各个功能模块
 - `common`：在 `model` 在定义了表
 - `accounts`：登录和切换账号功能
 - `questions`：生成题目
 - `utils`：三种题目生成器，被 `questions` 调用
- `\test_paper_generation\manage.py` 是后端启动文件
- `\test_paper_generation\Application.py` 是前端启动文件

3. API 结合

因为此工程是前后端分离，API 很关键，在此列出：

登录认证

- **URL:** `/api/login/`
- **Method:** POST
- **Description:** 用户通过此接口提供用户名和密码进行登录。
- **Request Body:**

```
{
  "username": "张三1",
  "password": "123"
}
```

- **Response:**
 - 成功:

```
{
  "user_id": 1,
  "status": 0,
  "message": "登录成功",
  "user_type": "小学" // 或 "初中", "高中"
}
```

- 失败:

```
{
  "status": 1,
  "message": "用户名或密码错误"
}
```

生成题目

- **URL:** `/api/questions/`
- **Method:** POST
- **Description:** 用户通过此接口指定题目数量并生成题目。
- **Request Headers:** 包含认证信息（例如：Authorization: Token xxxxx）
- **Request Body:**

```
{
  "user_id": 1,
  "count": 20,
  "user_type": "小学" // 或 "初中", "高中"
}
```

- **Response:**

- 成功:

```
{
  "status": 0,
  "message": "题目生成成功",
  "questions": [
    "1 + 2 = ?",
    "3 * 4 = ?",
    ...
  ]
}
```

- 失败:

```
{
  "status": 1,
  "message": "题目数量无效"
}
```

切换用户类型

- URL: /api/switch/
- Method: POST
- Description: 用户通过此接口切换用户类型。
- Request Headers: 包含认证信息
- Request Body:

```
{
  "user_id": 1,
  "new_type": "初中"  // 或 "小学", "高中"
}
```

- Response:
 - 成功:

```
{
  "status": 0,
  "message": "用户类型已切换"
}
```

- 失败:

```
{
  "status": 1,
  "message": "用户类型无效"
}
```

4. 表的构建

用户表 (users): 用于存储用户信息，包括用户名、密码、账户类型等。

字段名	类型	描述
id	INT AUTO_INCREMENT	主键，自动增长
username	VARCHAR (255)	用户名
password	VARCHAR (255)	密码
account_type	VARCHAR (255)	账户类型

	id	username	password	account_type
1	1	张三1	123	高中
2	2	张三2	123	初中
3	3	张三3	123	高中
4	4	李四1	123	小学
5	5	李四2	123	初中
6	6	李四3	123	高中
7	7	王五1	123	小学
8	8	王五2	123	初中
9	9	王五3	123	高中

文件表 (files): 用于存储生成题目的文件信息。

字段名	类型	描述
id	INT AUTO_INCREMENT	主键, 自动增长
user_id	INT	用户 ID, 外键关联 users 表
filename	VARCHAR (255)	文件名
created_at	DATETIME	创建时间

	id	filename	created_at	user_id
1	1	2024年09月14日19时42	2024-09-14 19:42:29.265460	1
2	2	2024年09月14日19时46	2024-09-14 19:46:49.125240	1
3	3	2024年09月14日19时51	2024-09-14 19:51:17.336711	1
4	4	2024年09月14日20时08	2024-09-14 20:08:59.386772	1
5	5	2024年09月14日20时11	2024-09-14 20:11:17.762305	1
6	6	2024年09月14日20时11	2024-09-14 20:11:44.777070	1
7	7	2024年09月14日20时32	2024-09-14 20:32:08.753104	1
8	8	2024年09月14日20时32	2024-09-14 20:32:33.461643	1
9	9	2024年09月14日20时41	2024-09-14 20:41:12.869278	1
10	10	2024年09月17日15时54	2024-09-17 15:54:33.663272	1
11	11	2024年09月17日16时11	2024-09-17 16:11:08.683458	7
12	12	2024年09月17日16时11	2024-09-17 16:11:48.499128	7
13	13	2024年09月17日18时34	2024-09-17 18:34:18.043949	1
14	14	2024年09月17日18时36	2024-09-17 18:36:46.479998	2
15	15	2024年09月17日18时37	2024-09-17 18:37:48.201298	2
16	16	2024年09月17日19时28	2024-09-17 19:28:05.951420	5
17	17	2024年09月17日19时37	2024-09-17 19:37:07.336643	3
18	18	2024年09月17日19时38	2024-09-17 19:38:32.433826	3
19	19	2024年09月17日19时41	2024-09-17 19:41:40.425413	2
20	20	2024年09月24日09时11	2024-09-24 09:11:51.471784	1
21	21	2024年09月24日09时12	2024-09-24 09:12:05.079103	1
22	22	2024年09月24日09时23	2024-09-24 09:23:43.636954	1
23	23	2024年09月24日09时24	2024-09-24 09:24:47.239952	1

题目表 (questions): 用于存储生成的题目信息，并增加一个外键来关联文件表。

字段名	类型	描述
id	INT AUTO_INCREMENT	主键, 自动增长
user_id	INT	用户 ID, 外键关联 users 表
file_id	INT	文件 ID, 外键关联 files 表
content	TEXT	题目文本
type	VARCHAR (255)	难度等级
created_at	DATETIME	创建时间

	id	content	type	created_at	file_id	user_id
1	1	92 + 22 + (42 - 8) / 33	小学	2024-09-14 19:42:29.265460	1	1
2	2	2 89 / 8 / 41 * 81	小学	2024-09-14 19:42:29.265460	1	1
3	3	40 - 17 * (35 + 91)	小学	2024-09-14 19:42:29.265460	1	1
4	4	73 * 49 - (61 + 52) / 70	小学	2024-09-14 19:42:29.265460	1	1
5	5	32 / 83 / 86 / 36 / 41	小学	2024-09-14 19:42:29.265460	1	1
6	6	13 + 18 / (76 + 75) / 71	小学	2024-09-14 19:42:29.265460	1	1
7	7	93 / 85 * 79 / 1	小学	2024-09-14 19:42:29.265460	1	1
8	8	18 / (46 - 27) * 64	小学	2024-09-14 19:42:29.265460	1	1
9	9	96 / 45 * (80 + 13)	小学	2024-09-14 19:42:29.265460	1	1
10	10	51 / (18 - 76) * 9 / 95	小学	2024-09-14 19:42:29.265460	1	1
11	11	77 / (66 + 38) / 78	小学	2024-09-14 19:42:29.265460	1	1
12	12	79 - 46 * 67 * (8 + 12)	小学	2024-09-14 19:42:29.265460	1	1
13	13	65 + (74 + 89) * 80	小学	2024-09-14 19:42:29.265460	1	1
14	14	1 - 91 * (38 - 100) / 47	小学	2024-09-14 19:42:29.265460	1	1
15	15	37 * (75 + 82) * 24	小学	2024-09-14 19:42:29.265460	1	1
16	16	98 * 37 + (5 + 30)	小学	2024-09-14 19:46:49.125240	2	1
17	17	14 - (34 + 14) * 85	小学	2024-09-14 19:46:49.125240	2	1
18	18	(34 - 75) * 6 / 27 * 36	小学	2024-09-14 19:46:49.125240	2	1
19	19	85 - 26 + 86 / (71 - 55)	小学	2024-09-14 19:46:49.125240	2	1
20	20	75 - 16 * 51 * (94 + 77)	小学	2024-09-14 19:46:49.125240	2	1
21	21	12 + 87 + 7 + (94 + 69)	小学	2024-09-14 19:46:49.125240	2	1
22	22	14 * 11 + (96 + 9)	小学	2024-09-14 19:46:49.125240	2	1
23	23	79 + 3 / 60 * (80 - 59)	小学	2024-09-14 19:46:49.125240	2	1
24	24	82 * 27 * 87 / 23	小学	2024-09-14 19:46:49.125240	2	1
25	25	92 - 7 + (27 + 7)	小学	2024-09-14 19:46:49.125240	2	1
26	26	17 * 5 * 39 / (81 - 71)	小学	2024-09-14 19:46:49.125240	2	1
27	27	35 - 28 / (65 - 16)	小学	2024-09-14 19:46:49.125240	2	1
28	28	80 - 48 + (68 + 1) / 57	小学	2024-09-14 19:46:49.125240	2	1

关系定义

- users 表中的 id 字段是 files 表中的 user_id 字段的外键。
- files 表中的 id 字段是 questions 表中的 file_id 字段的外键。
- users 表中的 id 字段也是 questions 表中的 user_id 字段的外键。

这里 Question 表不满足第三范式，多出来的冗余是为了更好地查询

四、总结与感悟

看完这些代码，我有以下几点思考和感悟：

1. 项目结构和模块化

项目的整体结构比较清晰，每个模块都有明确的职责和功能。然而，部分代码文件过于庞大，尤其是Application.py，包含了多个类和方法，导致文件长度较长，不利于维护和扩展。可以将不同的类拆分到不同的文件中，按照功能模块化。例如，将Authenticator类和CLIApp类分别放到单独的文件中，这样可以提高代码的可读性和可维护性。

2. 异常处理

在与后端通信和文件操作时，代码中有一定的异常处理机制，但还不够全面。例如，在生成题目和写入文件时，异常处理不足，可能会导致程序崩溃。可以在关键操作中增加更多的异常处理，确保程序的稳定性和健壮性。同时，可以考虑使用日志记录异常信息，便于后续排查和调试。

3. 硬编码问题

代码中存在一些硬编码的部分，例如URL、文件路径和用户信息等。这些硬编码的信息不够灵活，不利于配置和维护。可以将这些硬编码的信息配置化，使用配置文件或环境变量来管理。这样可以提高代码的灵活性和可维护性。

4. 测试覆盖率

测试代码覆盖了登录和切换用户类型的基本功能，但缺少对其他功能的测试，例如题目生成、文件写入和数据库操作等。可以增加更多的单元测试和集成测试，确保各个功能模块的正确性和稳定性。特别是对关键功能和边界情况的测试，能够提高代码的可靠性。

5. 代码注释和文档

代码中有一定的注释，但部分注释不够详细，尤其是在一些复杂的逻辑和算法部分，可能会导致理解困难。可以增加更多的注释和文档，特别是在关键逻辑和算法部分，便于后续维护和扩展。同时，可以考虑编写项目的README文件，详细说明项目的功能、架构和使用方法。

总结

总体来说，这个项目的代码结构清晰，功能实现完整，但在模块化、异常处理、硬编码、测试覆盖率和注释文档等方面还有提升空间。通过优化这些方面，可以提高代码的可读性、可维护性和可靠性。