珠海学院

**【BJ02】**

**课 程 设 计 报 告**

|  |  |
| --- | --- |
| 院 系： | 吉林大学珠海学院计算机学院 |
| 专 业： | 计算机科学与技术 |
| 题 目： | 基于爬虫、基于内容的推荐系统和SM-2间隔重复算法的单词助记App |
| 科 目： | 大数据技术 |
|  | 04192202郝义鹏 |
| 指导教师： | 林刚老师 |
| 完成时间： | 2022年6月26日 |

**目 录**

[1 引言 1](#_Toc107165599)

[1.1 目的 1](#_Toc107165600)

[1.2 背景 1](#_Toc107165601)

[2 需求分析 2](#_Toc107165602)

[2.1 系统功能 2](#_Toc107165603)

[2.1.1 注册和验证 2](#_Toc107165604)

[2.1.2 文章阅读 2](#_Toc107165605)

[2.1.3 单词查询 2](#_Toc107165606)

[2.2 面向的用户 2](#_Toc107165607)

[3 系统设计 3](#_Toc107165608)

[3.1总体设计 3](#_Toc107165609)

[3.2功能设计 3](#_Toc107165610)

[4 系统开发 8](#_Toc107165611)

[4.1源程序清单 8](#_Toc107165612)

[4.2功能实现 8](#_Toc107165613)

[4.2.1 MongoDB数据库模型模块 8](#_Toc107165614)

[4.2.2 HTTP协议后端接口模块 9](#_Toc107165615)

[4.2.3 登陆与token验证模块 9](#_Toc107165616)

[4.2.4 爬虫模块 10](#_Toc107165617)

[4.2.5 Redis服务 10](#_Toc107165618)

[4.2.6 请求模块 11](#_Toc107165619)

[4.2.7 处理模块 11](#_Toc107165620)

[4.2.8 基于内容的推荐系统模块 13](#_Toc107165621)

[4.2.9 查词功能和单词语音播放模块 14](#_Toc107165622)

[4.2.10 文章操作模块 14](#_Toc107165623)

[4.2.11 用户信息操作模块 14](#_Toc107165624)

[4.2.12 用户单词SM2算法模块 15](#_Toc107165625)

[5 系统测试 17](#_Toc107165626)

[5.1测试方法 17](#_Toc107165627)

[5.1.1 前端测试 17](#_Toc107165628)

[5.1.2 后端测试 17](#_Toc107165629)

[5.2测试实现 17](#_Toc107165630)

[5.2.1 前端测试 17](#_Toc107165631)

[5.2.2 后端测试 17](#_Toc107165632)

[5.3测试结果 20](#_Toc107165633)

[5.3.1 前端测试 20](#_Toc107165634)

[5.3.2 后端测试 27](#_Toc107165635)

[6 结论与心得 32](#_Toc107165636)

[参考文献 34](#_Toc107165637)

# 引言

## 1.1 目的

本项目的初衷是能让非英语母语环境的学习者能够沉浸在纯英语的使用环境中，最重要的是要接触地道的英语母语者写出的文章，并结合当前文章句子的语境学习其中的单词，如果说其他背单词APP只是一个选择题软件，那么我们设计的项目本质上是一个英文文章阅读器，使用者可以在文章中选择自己要背的单词并添加进单词表中，系统会根据智能记忆算法在不同的时间监督用户巩固并复习生词，并在后台评估和量化用户对每个单词的掌握程度。

系统还会根据智能推荐算法对每个用户感兴趣的文章进行个性化推荐新的文章，也会对不同类型的文章进行分类，让用户阅读自己喜欢的文章，保持对英语学习的兴趣。

## 1.2 背景

现有的背单词APP仅仅是简单罗列出应试教育的单词表在屏幕上依次显示出来，这些单词并没有在其对应的英文语境下出现，如某单词软件使用“助记图片”即让使用者的大脑把单词和一个精心绘制的一个图片联系起来，最后考察单词的掌握情况时也仅仅是在四个词义中选择一个。这种背单词的方法不仅依赖于其单词和图片的连接，而且对于大多数单词都有多个释义的情况下使用图片来记忆单个意思实际上并未完全掌握这个单词，并且抛开实际英文语境下的图片联想记忆也仅仅是自我欺骗。

很多人从小到大使用中英单词对照表像背词典一样的记忆方法在短时间内好像是掌握英语的捷径，但仅仅是应付考试而已，小学初中高中大学，对于英语的记忆一次次的重蹈覆辙，考完六级英语甚至考研英语但是无法流畅使用英语自己进行表达的大有人在，归根结底是拿着对照表背单词的方法看似是捷径，但其实对于应当日积月累的英语水平来说这种做法是切实的欲速则不达的行为。

# 2 需求分析

## 2.1 系统功能

用户注册并首次登录后首先可以选择自己感兴趣的文章类别，点进文章后系统的智能推荐算法开始根据用户感兴趣的相似内容进行推送，遇到想要查询的单词可以直接点击查询释义，并且可以播放其发音。

每过一段时间，系统根据智能记忆算法计算出用户何时该复习之前选中的单词，用户只需打开软件的复习巩固功能即可自我考察单词掌握情况。

### 2.1.1 注册和验证

用户可通过用户名，密码和邮箱创建一个用户。注册成功后本地会记录登录信息，在一段时间内登录有效。连续登陆可以更新登陆状态。

### 2.1.2 文章阅读

用户可根据不同主题的标签查看不同类别的文章，在点开文章之后系统会根据之前感兴趣的文章推送下一篇文章。

### 2.1.3 单词查询

点击文章内的单词，可以查看单词的释义并且播放单词发音。可以把选择的单词加入要背的列表，以便日后巩固复习。

## 2.2 面向的用户

本系统可面向的用户包括但不限于各个年龄段学习英语的学生，准备各类英语考试的学生，推送的文章尤其对六级考试和考研英语考试者友好。

# 3 系统设计

## 3.1总体设计

使用者可以在文章中选择自己要背的单词并添加进单词表中，系统会根据智能记忆算法在不同的时间监督用户巩固并复习生词，并在后台评估和量化用户对每个单词的掌握程度。系统还会根据智能推荐算法对每个用户感兴趣的文章进行个性化推荐新的文章，也会对不同类型的文章进行分类，让用户阅读自己喜欢的文章。

## 3.2功能设计

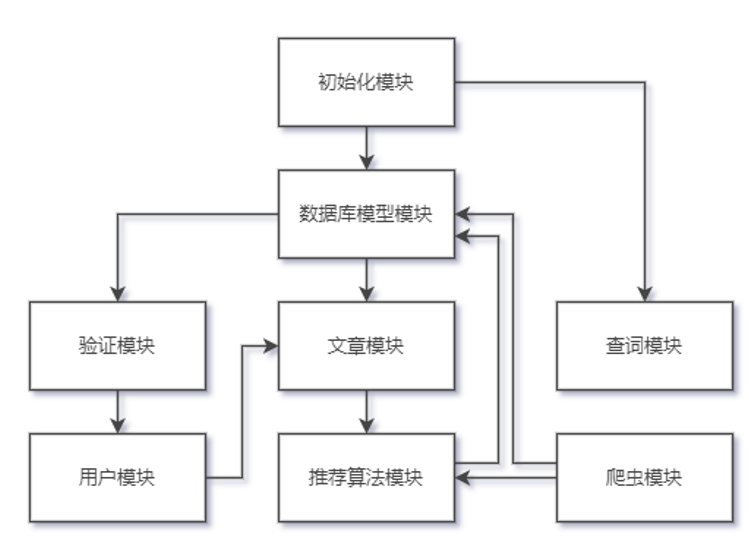


图1 模块图

表1 后端接口说明表

|  |  |  |
| --- | --- | --- |
| **接口** | **名称** | **作用** |
| /auth/token | Login For Access Token | 验证token |
| /auth/refresh | Refresh Token | 更新token |
| /register/register | Register New User | 注册新用户 |
| /register/updateProfile | Update Profile | 更新用户信息 |
| /article/get\_article\_info | Get Article Info | 根据文章标题获取文章详细信息：内容、分类、推荐文章标题列表、 |
| /article/get\_random\_articles | Get Random Articles | 未登录状态下随机获取文章,返回amount数量的文章list及其预览图list（base64编码） |
| /article/get\_recommend\_articles | Get Recommend Articles | 已登录状态下获取文章,返回amount数量的文章list及其预览图list（base64编码） |
| /user/add\_word | Add Word | 添加用户选择的单词（参数为单词字符串） |
| /user/get\_words | Get Words | 获取用户选择的单词（返回单词对象列表，包含sm2算法信息等） |
| /user/delete\_word | Delete Word | 删除用户选择的单词（参数为单词字符串） |
| /user/add\_article | Add Article | 添加用户选择的文章（参数为文章标题字符串） |
| /user/get\_articles | Get Articles | 获取用户选择过的文章（返回文章标题字符串列表） 得到文章标题后使用/article/get\_article\_info接口获取文章详细信息 |
| /user/delete\_article | Delete Article | 删除用户选择的文章（参数为文章标题字符串） |
| /sm2/<word>/updateEFactor/<quality> | Update Word E Factor | 更新单个单词的背诵情况 |
| /word/<word>/audio | Audio | 生成指定单词的音频文件 |
| /word/<word>/lookup | Lookup | 从redis中根据key查询 |

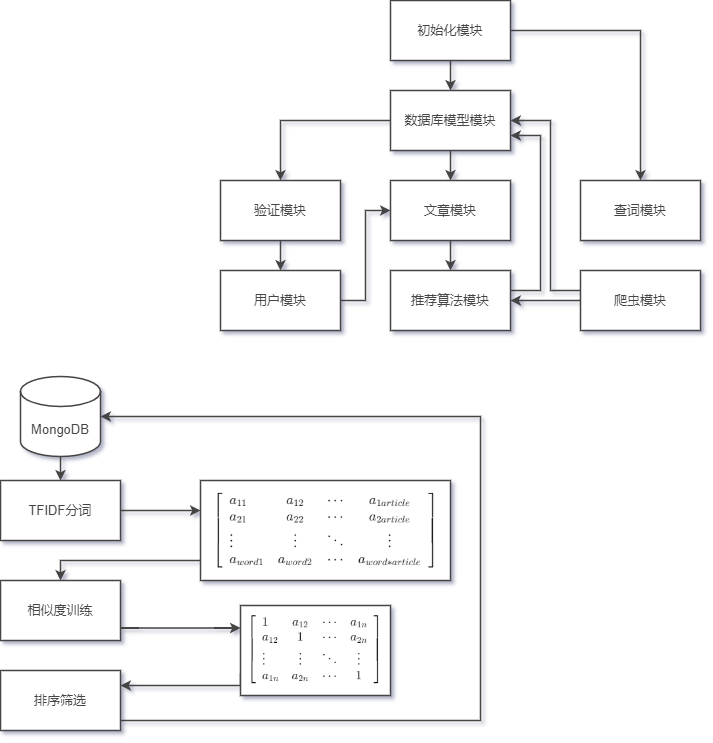


图2 推荐系统

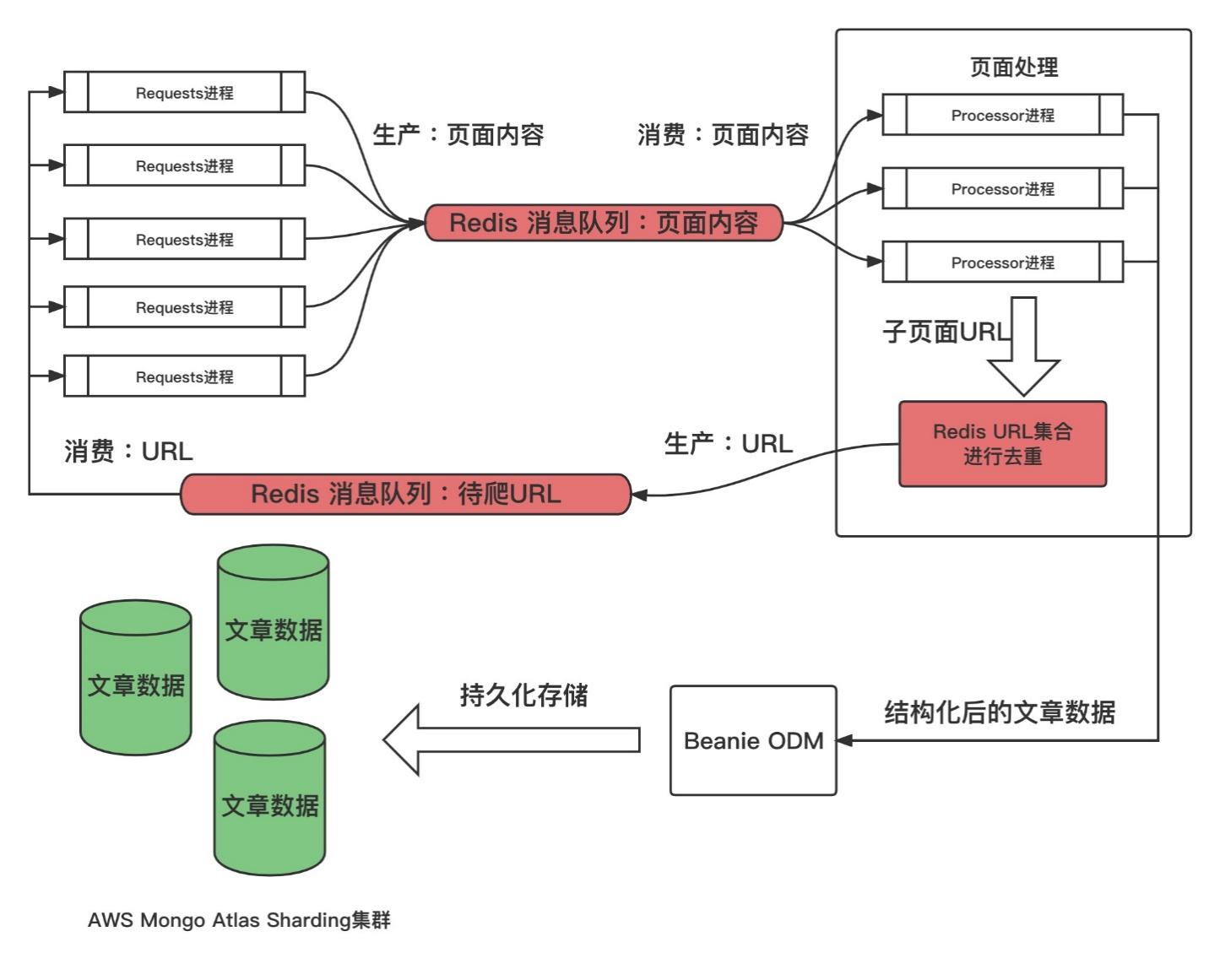


图3 爬虫系统

# 系统开发

## 4.1源程序清单

前端代码仓库：

Public/：静态资源文件

Src/：源代码目录

Assets/：资源文件目录

Component/：组件目录

Router/：路由目录

Store/：存储类目录

Views/：页面目录

后端代码仓库：

Docs/：文档目录

Env/：服务器环境目录

.gitignore：git上传忽略列表

Article.py：文章操作模块

Auth.py：验证模块

Ecdict.csv：持久化字典文件

KmeansTest.py：Kmeans训练算法

Main.py：服务器主逻辑

Model.py：数据库模型模块

Mongodb\_url：MongoDB数据库地址

Proxies：代理池地址

README.md：服务器手册

Recommend.py：基于内容的推荐系统模块

Register.py：注册模块

SM2：记忆算法模块

Spider.py：爬虫模块

User.py：用户操作模块

Utility.py：工具类模块

WordLookup.py：查词模块

## 4.2功能实现

## 4.2.1 MongoDB数据库模型模块

本项目使用MongoDB数据库存储用户注册信息（包含账户名、邮箱、密码哈希值等）、文章信息（文章标题、内容、插图等）、每个用户要背的每个单词的掌握程度信息（单词初次添加时间、复习时间等）。为了能够让各个模块与数据库灵活交互、互相解耦，使用beanie数据库操作框架定义设计了一系列模型与MongoDB交互，每个数据库中的表都被抽象成一个类，每个数据库对象则关联数据库中的行。

为了保护用户的个人隐私，数据库内不允许明文存储账号密码，而是使用哈希值，提高不法分子的犯罪成本。

数据库操作模型设计符合最小权限原则，即每个模块应该具有完成其任务所必需的最小权限集合，例如在本项目中用户模型类被分别设计成两个类：”后端用户类“和”数据库用户类“，其中的区别则是后端用户类比数据库中的用户类缺少一个密码哈希值字段，这样做的目的是除了登陆与token验证模块可以得到用户的密码哈希值进行必要的验证步骤外，当数据库用户类对象随着系统流程进入其他模块操作时则会变为后端用户类对象，不仅对后面的流程优化掉了无用的字段，让模块之间的流程更加清晰，也屏蔽了敏感数据减少了系统出现漏洞的可能性，提高了系统的安全性。

## 4.2.2 HTTP协议后端接口模块

本项目后端使用FastAPI库对HTTP协议接口进行定义，每个与前端交互的模块暴露出HTTP接口。规定需要用户登录的接口以及返回即时性数据的接口使用“POST”方法定义，例如用户的文章喜好推荐、正在背的单词等；与用户登录无关且长时间不会修改可以利用客户端缓存提高用户体验的非即时性数据使用“GET”方法定义，例如文章内容、插图等；

后端接口划分出七个路由，接口功能设计简约紧凑，使用接口文档工具实时生成文档，与前端完全解耦，模块分工明确，开发时约定好接口后则开始前端与后端同时开发。

## 4.2.3 登陆与token验证模块

用户的注册与登陆使用的是基于JWT(Json Web Token, RFC 7519)的OAuth2认证标准，在后端提供两个接口用于签发Token和更新Token。

传统基于session的验证模式在需要对session的信息在服务端进行存储，处理请求时需要在后端对其进行查询和校验。而JWT无需查询，JWT通过以在使用服务端指定的密钥进行Token的签发，签发后，客户端发送请求时附带此Token，服务端使用签发时的token进行同样的加密，与客户端回传的签名进行比对。

所以，基于此特性，跨域认证问题也得到了解决，在不同的服务器上只需要共用同一个secret，就可以解决跨域认证问题。

一个使用JWT进行验证的请求由三部分组成：header、payload、signature，它们由符号"."进行链接。一个典型的JWT结构如下：

``header.payload.signature``

Header中有两个参数：alg(orithm),typ(e)，存储这用于所使用的加密算法及验证类型：

``{ "alg": "HS256", "typ": "JWT" }``

payload中，同样是以 JSON 的方式存储 JWT 验证系统的一些元数据，如过期时间、签发时间、 Token 签发者，同时还保存了其他由开发者定义的想要传递的数据，如用户权限、用户信息等，以减少信息交换次数，减轻服务器负担：

``

{

"username": "Huang Jing Jun",

"uid": "04191021",

"hobby": "programming",

"aud": "abc",

"iss": "def",

"exp": 1622360841,

"iat": 1622360841

}

``

signature：对 header 和 payload 进行 Base64URL 编码并拼接后，使用 header 中指定的加密方式，通过服务端的密钥进行签名，最后与 header 和 payload 进行拼接：

``

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6Ikh1YW5nIEppbmcgSnVuIiwidWlkIjoiMDQxOTEwMjEiLCJob2JieSI6InByb2dyYW1taW5nIiwiYXVkIjoiYWJjIiwiaXNzIjoiYWRtaW4iLCJleHAiOjE2MjIzNjA4NDEsImlhdCI6MTYyMjM2MDg0MX0.Sb8VYQt3QZ6QrJpNr6U\_Zx1xaSeUPRwx244-qbQMQrM

``

把以上 Token 返回给客户端，即完成 Token 签发，客户端仅需在请求时，附带此 Token，即可配合服务端进行校验。

同时，带来的安全问题也不容忽视，由于服务器不保存 session 状态，因此无法在使用过程中废止某个 token，或者更改 token 的权限。也就是说，一旦 JWT 签发了，在到期之前就会始终有效，除非服务器部署额外的逻辑。

一般的解决方案是使用 Redis 创建一个 Token 黑名单集合，若用户手动登出，则将此 Token 加入 Redis 集合中，并指定其过期时间，过期后释放，减少内存负担。但本项目出于开发周期考虑没有实现登出功能，登出由客户端抛弃 Token 实现，若 Token 被盗用且未过期，则存在一定安全风险。

## 4.2.4 爬虫模块

爬虫模块主要分为四个模块：请求模块、处理模块、Redis服务、持久化模块，每一个模块均可独立部署在任意节点上，模块间通过共享的远程Redis服务器进行通信，亦可在本地同时部署。

## 4.2.5 Redis服务

Redis服务是爬虫模块的核心，其负责着系统的控制与信息交换。该模块通过Redis List、Redis Set、Redis Hash，来实现了系统的控制与通信。

URL Redis List：负责把处理模块生产的URL传递给请求模块

页面内容Redis List：负责把请求模块生产的页面内容HTML文本传递给处理模块

URL Redis Set：记录从页面中分离出来的URL，用于URL去重，避免循环重复爬取。

其他：一部分大Key用来控制系统起停。

缺陷：使用Redis来控制系统的启停存在单点失效问题。

## 4.2.6 请求模块

请求模块由Python Requests库实现，通过从Redis URL消息队列中作为消费者，请求待爬取URL，并对其发送请求，返回原始的HTML。同时可以指定User Agent信息、请求体、代理、构造登陆信息等操作。

缺陷：对于使用Ajax等技术进行异步加载的CSR（Client-side Rendering，客户端渲染）页面，需要执行其Javascript进行渲染才能加载出页面的核心内容，否则无法读取页面内容，因此，对于异步加载的页面，需要使用浏览器引擎进行页面渲染后，才能获取到包含内容的HTML，常见的解决方案是基于浏览器引擎进行渲染（比如Headless Chrome），亦或者使用支持CSR的请求模块。

在本项目场景中，爬取的以新闻信息为主，网站为了提升性能和减少费用，常通过CDN分发内容，一般都为纯静态页面或SSR（Server-side Rendering，服务端渲染）且无用户状态的预渲染页面，故一般无需考虑异步加载问题，本项目中更值得关注的，是由CDN服务商（如Cloudflare）等CDN服务提供商所设置的防火墙，因为高频的请求可能会出发CDN服务的防火墙策略，导致请求被拒绝。 对于此类情况，常通过购买代理池来解决。

## 4.2.7 处理模块

内容模块扮演着从原始的HTML标签语言里提取文本内容的角色。

在本项目中，处理模块负责从页面原始的HTML里提取文章和子页面的URL，并对其进行清洗和结构化。因为网页属于非结构化的数据，不同于使用API接口的爬取。处理模块需要针对爬取的目标网站进行单独开发，这是无法避免的，但思路和方法都是一样的，对于本项目，分析与实现思路如下：

**标签分析:**

对于文章标题的提取，通过提取标签的内容即可完成。



对于图片也可以这么处理。

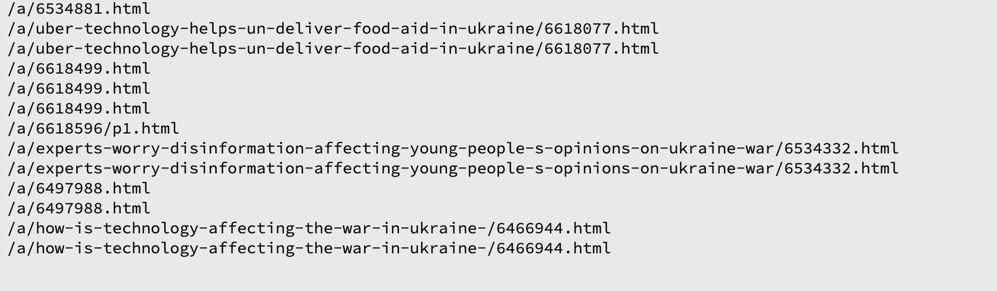
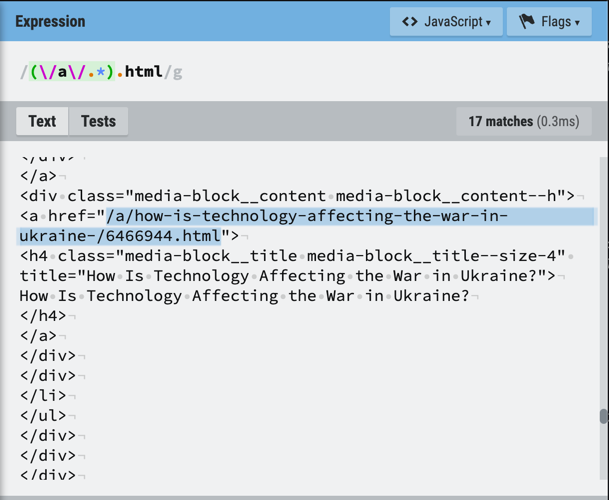
**正则匹配:**

通过对本项目选用的VOA Learning English为例，对目标页面的URL进行分析：



可以发现该网站含文章的URL路径均为协议://example.com/a/\*.html的形式，针对这一特征，可以通过Python的re进行匹配，提高文章URL的命中率。

通过正则表达式(\\/a\\/.\*).html进行匹配的结果：



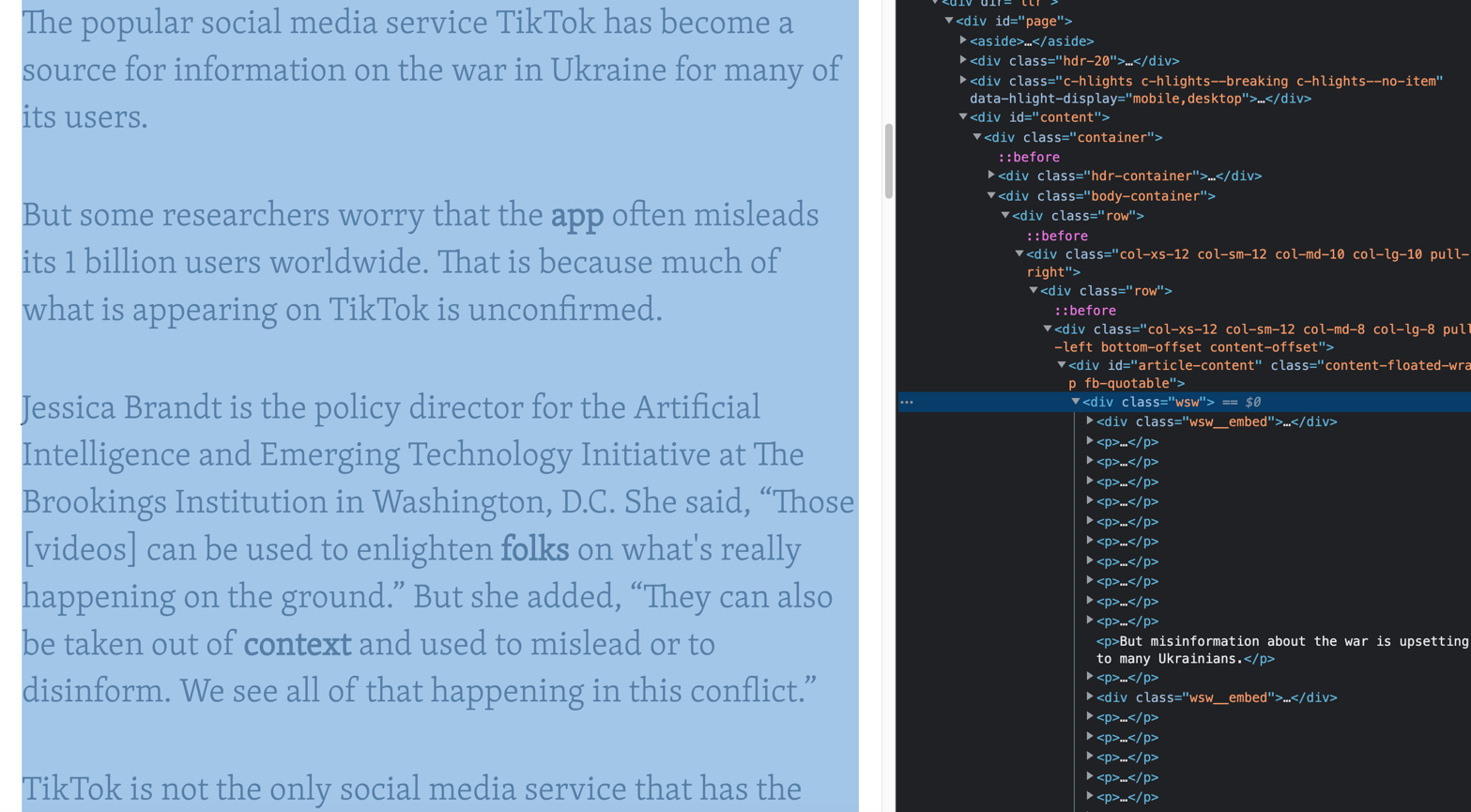
**CSS Selector分析:**

网站上不同类别的内容常常使用不同的CSS样式表，通过这一特点，可以分析网站内欲爬取数据的CSS样式表共同点。

对该网站的样式表进行分析，发现文章内容的 class 均为 wsw, 通过BeautifulSoup进行搜索，抽取出包含文章正文的标签：

**过滤:**

使用以上方法对于一些无关内容、多媒体内容、失效URL设定规则进行过滤：



**不足：**

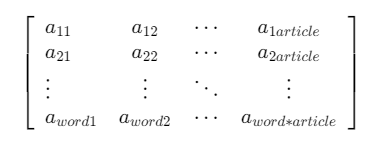
在一个爬取的里面，包含许多非文章页面URL，降低爬取效率，故抛弃掉所有不被正则匹配的URL。 但导致了，对于深度较深的我文章无法爬取（如：需要多次跳转才到能打的深度页面）。因为爬虫模块的URL消息队列采用普通的队列的实现，无法对URL进行分级，若改用优先队列，给URL指定优先级进行爬取，可以解决这个问题。



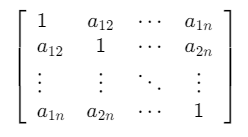
## 4.2.8 基于内容的推荐系统模块

本项目使用的推荐系统是基于内容的推荐系统，即对每个文章的内容都会进行训练和处理，此推荐系统独立于后端服务器运行。以下为推荐系统的工作流程：

1. 爬虫收集到的所有文章在数据库，将文章内容全部下载到机器里
2. 将文章内容使用TFIDF算法进行分词处理
3. 将分词处理后的数据使用格拉姆矩阵计算余弦相似度



1. 对计算后的相似度矩阵进行排序
2. 对排序后的矩阵只保留相似度前几名



1. 对相似度训练结果录入数据库

训练结束后每个文章就会都关联一些与其内容相似的文章，用于给其他模块做推送。

## 4.2.9 查词功能和单词语音播放模块

本项目单词词库高达七十八万条，每个单词包含音标、发音、各种知名词典中的释义等十多条信息，这些数据在系统启动时会被导入到高性能内存数据库Redis中，以单词字符串为键索引，其他信息为其值。当用户需要查询一个单词时则会立刻返回单词信息，还可以请求通过TTS生成的，经过base64编码的单词的发音音频（可选）。

## 4.2.10 文章操作模块

文章操作模块包含了对文章操作的接口，即增、删、改、查、关联推荐、随机推荐等。

## 4.2.11 用户信息操作模块

每个用户在数据库中除了账户密码等基础数据外都会为其生成一个用户选择的背单词的列表和一个记录感兴趣的文章的列表，这个模块的作用则是对数据库中的用户数据进行操作，比如获取用户某个单词的掌握程度，根据用户的文章喜好返回新的关联推荐文章等。

## 4.2.12 用户单词SM2算法模块

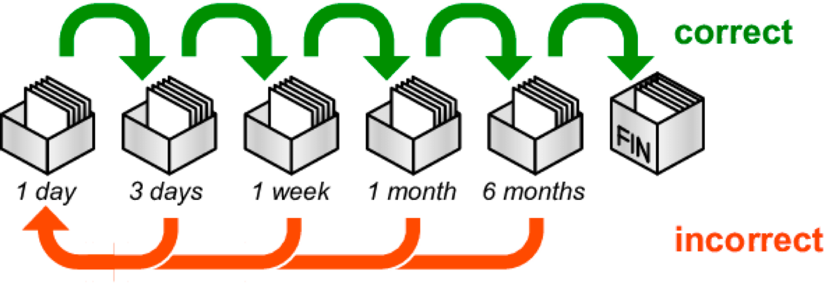
**Flashcard和间隔记忆算法**

Wikipedia上的定义：

抽认卡或称闪卡（英文：Flash Card）、字卡，是一小块纸片，在学校里用来作辅助教学。抽认卡可以用来记录词汇、历史事件时间、公式等等。使用抽认卡的目的主要是帮助记忆，提高学生的学习动机、以及在课堂上为学生提供多种感官的刺激。你在每个卡片上写下一个问题（同时在背面记下答案），用它们来测试自己，并根据你的测试、学习结果把它们进行排序、分组。



市面上常见的flashcard产品



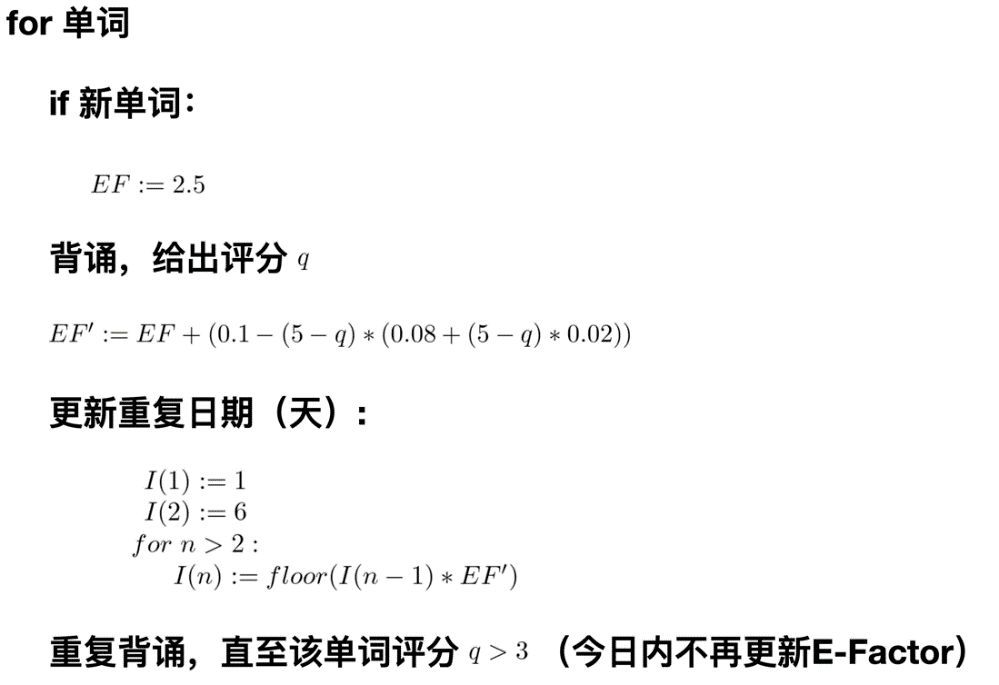
Leitner System

间隔记忆是一种循证教育学习方法，是一种通过不断重复并不断增加时间间隔的记忆方法。配合Flashcard的记忆方法首次出现在Sebastian Leitner 在1973年提出的 Leitner System，为基于Flashcard的讲个记忆系统奠定了雏形。

本项目采用的为SuperMemo-2算法，由波兰学者Piotr Woźniak与1990年在他的学位论文中提出，并开发出各平台版本，是最早期的计算机辅助记忆软件。现在用户最多的记忆软件，如Anki也仍然在使用SM-2记忆算法。

遍历到一张单词Flashcard，对单词卡进行以下操作

该算法可以被描述为（有简化）：



其中E-Factor为该单词记忆指数，E-Factor越高，记忆越牢固

其中q为用户背诵时的评分，该评分基于以下规则给出：

0: "Total blackout", complete failure to recall the information. 彻底遗忘。

1: Incorrect response, but upon seeing the correct answer it felt familiar. 回答错误，但是正确答案很熟悉。

2: Incorrect response, but upon seeing the correct answer it seemed easy to remember.回答错误，但很容易回忆起正确答案。

3: Correct response, but required significant effort to recall. 回答正确，回忆过程有困难

4: Correct response, after some hesitation. 回答正确，但是选择不够果断

5: Correct response with perfect recall. 回答正确，秒选

SM-2根据以上规则对用户的记忆效果进行评估，从而调整E-Factor，根据公式：

img13png

易知E-Factor随着复习质量q的上升而上升，而E-Factor越大，复习的时间间隔就越大。

由首次记忆某一单词日期算起，除第一次复习在次日及第二次复习在六天后以外，第img16png此复习日期img14png与E—Factor的关系可表示为一个等比数列：

img17png

对于非整数结果，应进行向下取整或四舍五入。

通过在后端定义的ODM模型，存储每一个用户的待记忆单词的E-Factor、记忆次数、及其初始记忆日期等属性到MongoDB中。

# 5 系统测试

## 5.1测试方法

## 5.1.1 前端测试

前端使用via进行测试。

## 5.1.2 后端测试

后端使用FastAPI部署的测试页面进行接口测试。

## 5.2测试实现

## 5.2.1 前端测试

**1．登录页面**

**2．浏览文章并在文章内点击单词来查词**

**3．单词复习**

**4．文章列表**

## 5.2.2 后端测试

**1.首先使用用户名、密码、邮箱注册一个账号：**

curl -X 'POST' \

'http://127.0.0.1:8000/register/register' \

-H 'accept: application/json' \

-H 'Content-Type: application/json' \

-d '{

"username": "test",

"email": "test@test.com",

"raw\_password": "testpassword"

}'

Request URL：

<http://127.0.0.1:8000/register/register>

**2.然后就可以使用这个账号登陆：**

curl -X 'POST' \

'http://127.0.0.1:8000/auth/token' \

-H 'accept: application/json' \

-H 'Content-Type: application/x-www-form-urlencoded' \

-d 'grant\_type=&username=test&password=testpassword&scope=&client\_id=&client\_secret='

Request URL：

<http://127.0.0.1:8000/auth/token>

**3．添加单词：**

curl -X 'POST' \

'http://127.0.0.1:8000/user/add\_word?word=nickel' \

-H 'accept: application/json' \

-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ0ZXN0IiwiZXhwIjoxNjU2MTczOTY4fQ.hJqKttVsN0UDF-0EdcymxrjA5qAPrZb2eS95OjRw57Q' \

-d ''

**4.获取用户选择的单词：**

curl -X 'POST' \

'http://127.0.0.1:8000/user/get\_words' \

-H 'accept: application/json' \

-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ0ZXN0IiwiZXhwIjoxNjU2MTczOTY4fQ.hJqKttVsN0UDF-0EdcymxrjA5qAPrZb2eS95OjRw57Q' \

-d ''

Request URL：

<http://127.0.0.1:8000/user/get_words>

**5.删除用户选择的单词：**

curl -X 'POST' \

'http://127.0.0.1:8000/user/delete\_word?word=nickel' \

-H 'accept: application/json' \

-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ0ZXN0IiwiZXhwIjoxNjU2MTczOTY4fQ.hJqKttVsN0UDF-0EdcymxrjA5qAPrZb2eS95OjRw57Q' \

-d ''

Request URL：

<http://127.0.0.1:8000/user/delete_word?word=nickel>

**6.添加感兴趣的文章：**

curl -X 'POST' \

'http://127.0.0.1:8000/user/add\_article?title=EU%20Criticizes%20Iran%20Over%20Nuclear%20Violations' \

-H 'accept: application/json' \

-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ0ZXN0IiwiZXhwIjoxNjU2MTczOTY4fQ.hJqKttVsN0UDF-0EdcymxrjA5qAPrZb2eS95OjRw57Q' \

-d ''

Request URL：

<http://127.0.0.1:8000/user/add_article?title=EU%20Criticizes%20Iran%20Over%20Nuclear%20Violations>

**7.获取用户选择过的文章列表：**

curl -X 'POST' \

'http://127.0.0.1:8000/user/get\_articles' \

-H 'accept: application/json' \

-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ0ZXN0IiwiZXhwIjoxNjU2MTczOTY4fQ.hJqKttVsN0UDF-0EdcymxrjA5qAPrZb2eS95OjRw57Q' \

-d ''

Request URL：

<http://127.0.0.1:8000/user/get_articles>

**8.根据文章标题获取文章详细信息：**

curl -X 'GET' \

'http://127.0.0.1:8000/article/get\_article\_info?title=EU%20Criticizes%20Iran%20Over%20Nuclear%20Violations' \

-H 'accept: application/json'

Request URL：

<http://127.0.0.1:8000/article/get_article_info?title=EU%20Criticizes%20Iran%20Over%20Nuclear%20Violations>

**9.删除感兴趣的文章：**

curl -X 'POST' \

'http://127.0.0.1:8000/user/delete\_article?title=EU%20Criticizes%20Iran%20Over%20Nuclear%20Violations' \

-H 'accept: application/json' \

-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ0ZXN0IiwiZXhwIjoxNjU2MTczOTY4fQ.hJqKttVsN0UDF-0EdcymxrjA5qAPrZb2eS95OjRw57Q' \

-d ''

Request URL：

<http://127.0.0.1:8000/user/delete_article?title=EU%20Criticizes%20Iran%20Over%20Nuclear%20Violations>

**10.未登录状态下随机获取文章：**

curl -X 'GET' \

'http://127.0.0.1:8000/article/get\_random\_articles?amount=10' \

-H 'accept: application/json'

Request URL：

<http://127.0.0.1:8000/article/get_random_articles?amount=10>

**11.已登录状态下获取文章：**

curl -X 'GET' \

'http://127.0.0.1:8000/article/get\_recommend\_articles?amount=10' \

-H 'accept: application/json' \

-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ0ZXN0IiwiZXhwIjoxNjU2MTczOTY4fQ.hJqKttVsN0UDF-0EdcymxrjA5qAPrZb2eS95OjRw57Q'

Request URL：

<http://127.0.0.1:8000/article/get_recommend_articles?amount=10>

**12.刷新token：**

curl -X 'POST' \

'http://127.0.0.1:8000/auth/refresh' \

-H 'accept: application/json' \

-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ0ZXN0IiwiZXhwIjoxNjU2MTczOTY4fQ.hJqKttVsN0UDF-0EdcymxrjA5qAPrZb2eS95OjRw57Q' \

-d ''

Request URL：

<http://127.0.0.1:8000/auth/refresh>

**13.从redis中根据key查询：**

curl -X 'GET' \

'http://127.0.0.1:8000/word/nickel/lookup' \

-H 'accept: application/json'

Request URL

http://127.0.0.1:8000/word/nickel/lookup

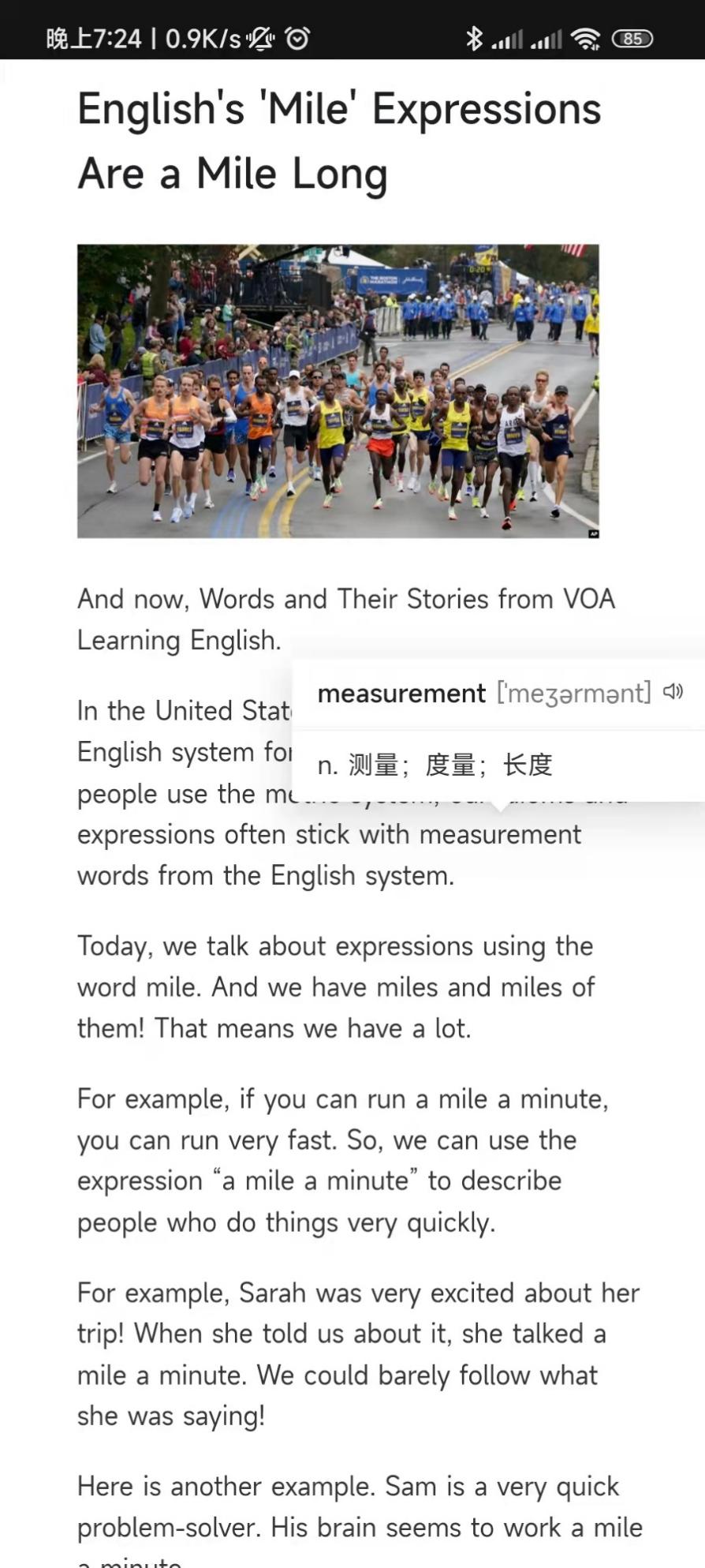
## 5.3测试结果

## 5.3.1 前端测试

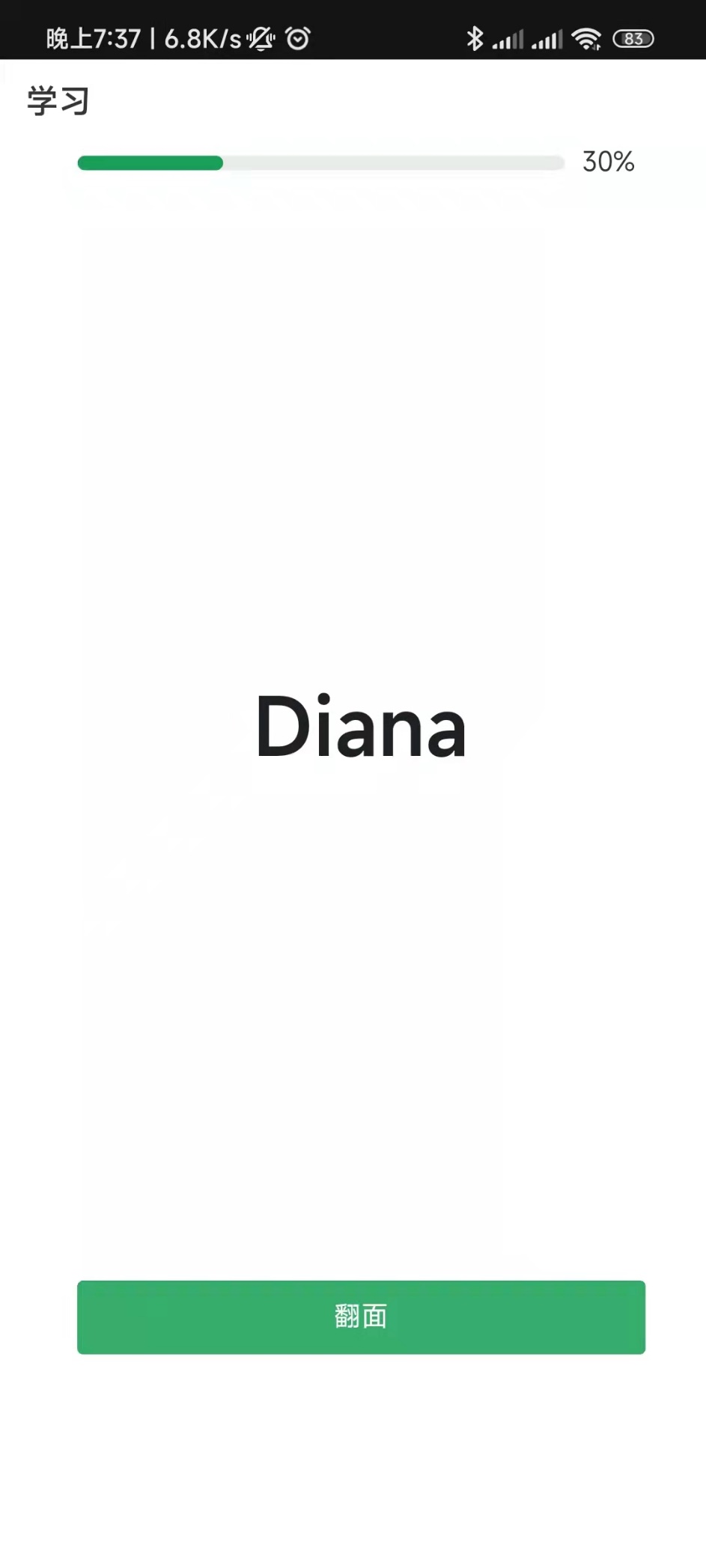
**1．登录页面**

****

**2．浏览文章并在文章内点击单词来查词**

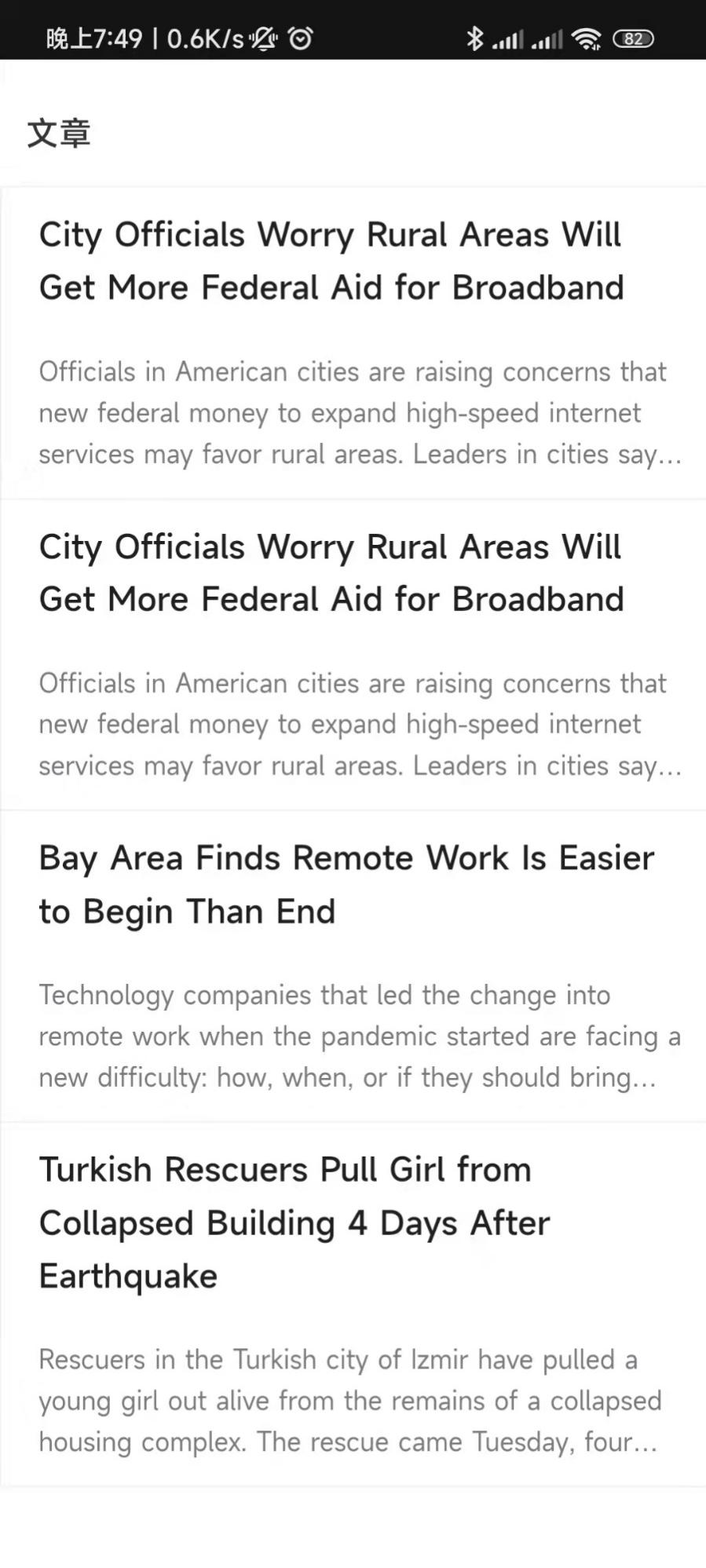
****

**3．单词复习**



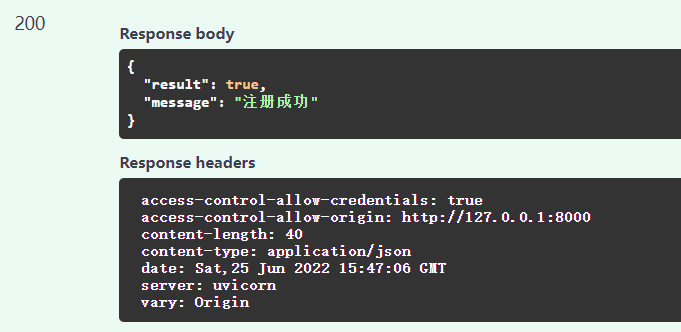


**4．文章列表**

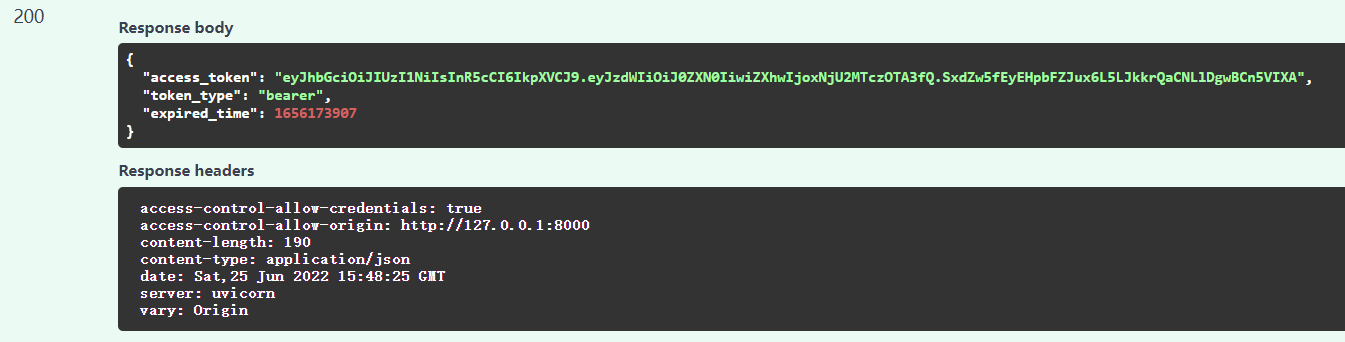


## 5.3.2 后端测试

1.注册账号：测试通过



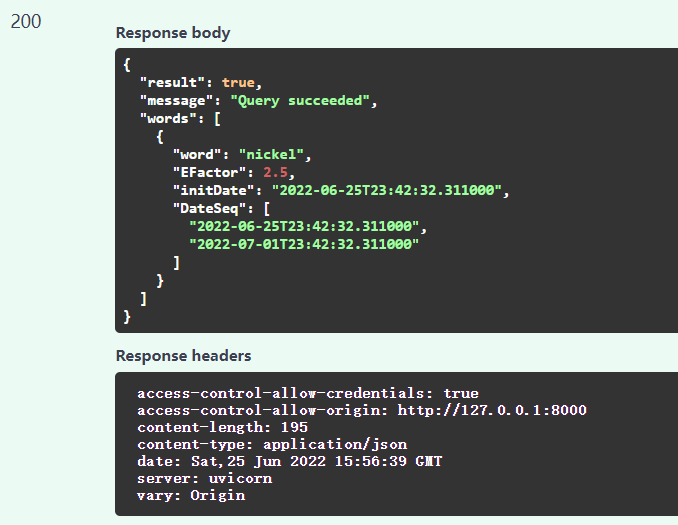
2.登陆账号：测试通过



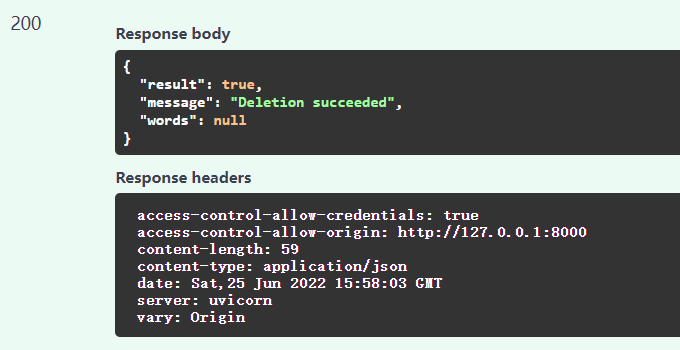
3.添加单词：测试通过



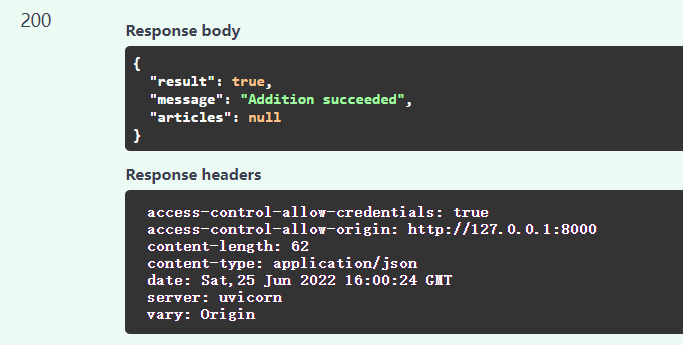
4.获取用户选择的单词：测试通过



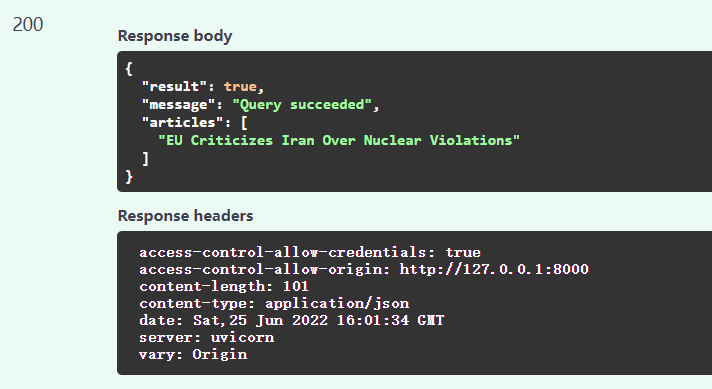
5.删除用户选择的单词：测试通过



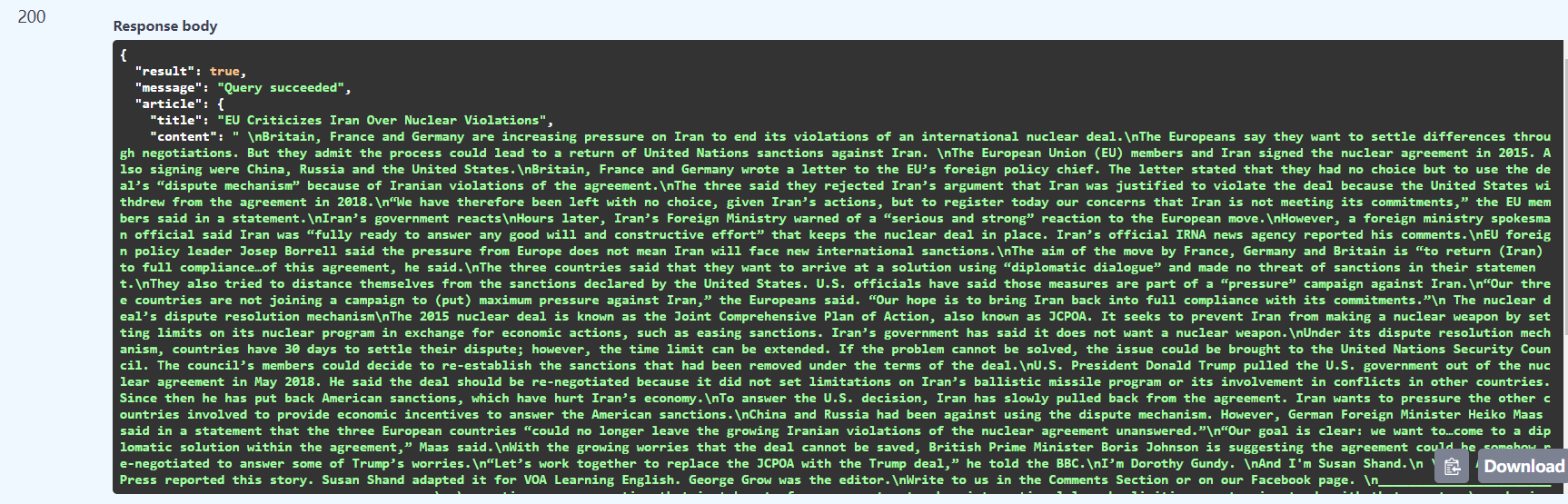
6.添加感兴趣的文章：测试通过



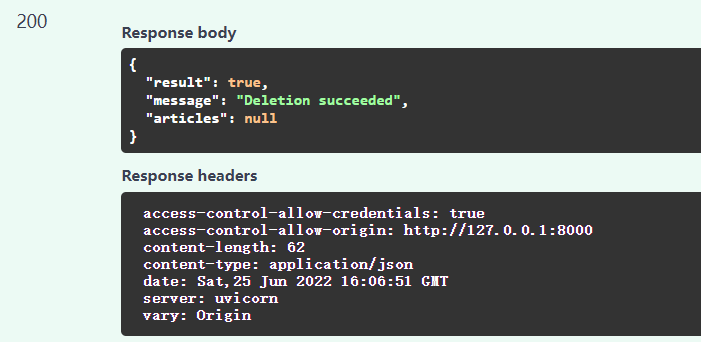
7.获取用户选择过的文章列表：测试通过



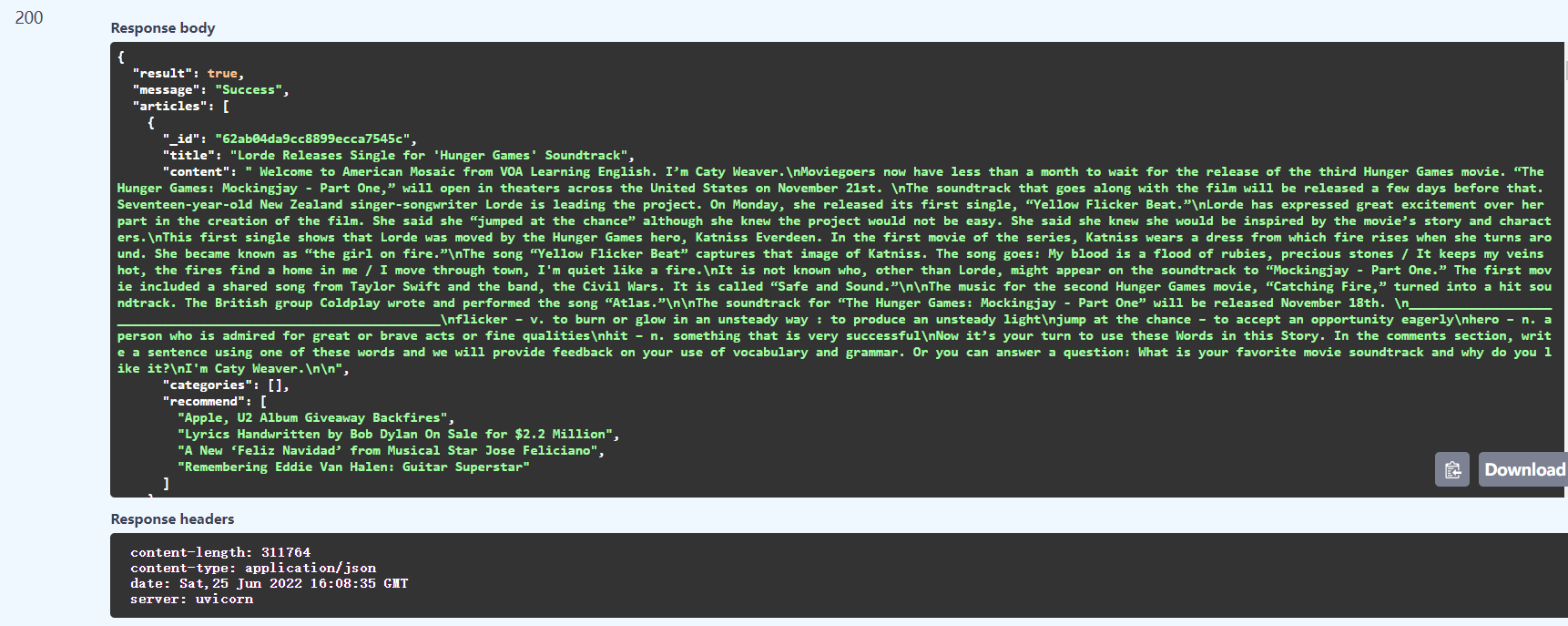
8.根据文章标题获取文章详细信息：测试通过



9.删除感兴趣的文章：测试通过



10.未登录状态下随机获取文章：测试通过



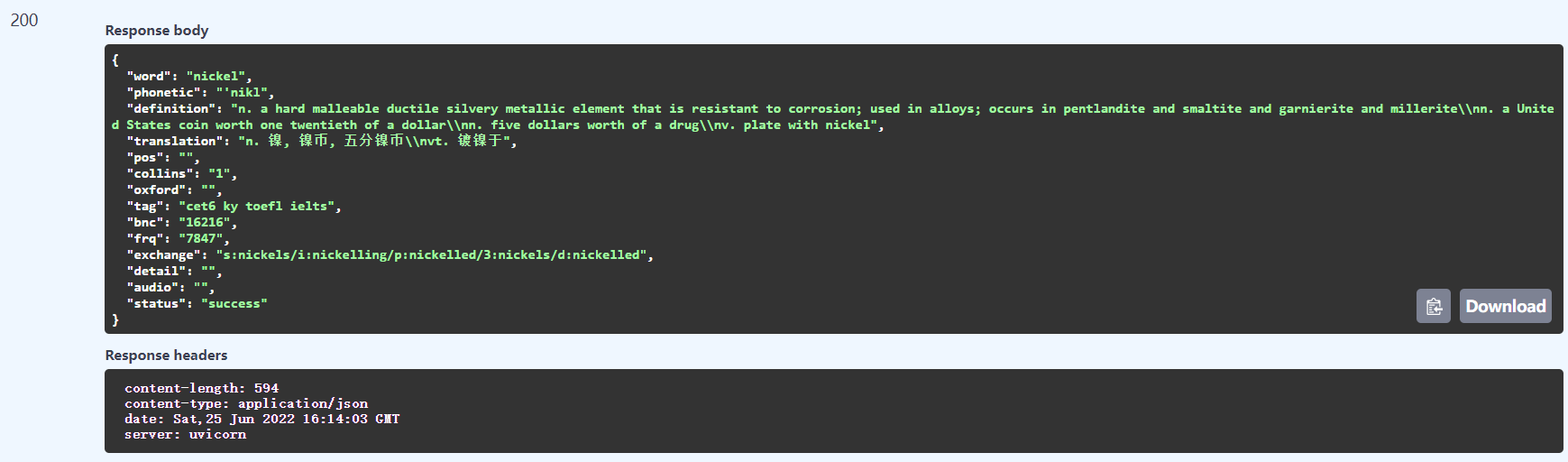
11.已登录状态下获取文章：测试通过



12.刷新Token：测试通过



13.从redis中根据key查询：测试通过



# 6 结论与心得

起初我在本地主机上进行了伪分布式部署，可本地主机性能有限，只能进行一些验证性的试验工作，爬虫效率没有得到提升；如果购买多台弹性计算服务器，对于这些短时间执行的工作，将会带来较高的成本。

后来我从AWS（亚马逊云）的教程里了解到了Serverless（无服务器计算，FaaS，功能即服务），Serverless十分适合本项目这种短时间执行的计算任务（如每天爬半个小时）。最终我们选择了对学生提供较高免费额度的腾讯云“云函数”。

Graphical user interface, text

Description automatically generated Graphical user interface, text, application

Description automatically generated

腾讯云的Serverless支持单实例多并发，在一个实例被远程调用所创建后，执行完计算任务后不会立刻销毁，而会保持一定的存活时间。通过这一特性，我可以实现对运算实例的复用，减少实例创建的时间，且可以通过动态地调整实例数量来调整爬取性能。而且腾讯云Serverless实例可以创建在不同的地区，这可以天然地替代代理池，提供不同地域的IP，减少风控的情况。

**Graphical user interface, application

Description automatically generated**

腾讯云的Serverless支持单实例多并发，在一个实例被远程调用所创建后，执行完计算任务后不会立刻销毁

# 参考文献

\* Inst.eecs.berkeley.edu. 2022. Gram matrix. [online] Available at: <https://inst.eecs.berkeley.edu/~ee127/sp21/livebook/def\_Gram\_matrix.html> [Accessed 20 June 2022].

\* Unger, H., A, Y. and Mamgain, S., 2022. CNN News Articles from 2011 to 2022 Dataset. [online] Kaggle.com. Available at: <https://www.kaggle.com/datasets/hadasu92/cnn-articles-after-basic-cleaning> [Accessed 22 June 2022].

\* Pariser, E. and Heyborne, K., 2011. The filter bubble. [Old Saybrook, Ct.]: Tantor Media, Inc.

\* Wozniak, P.A., 1990. Optimization of learning: A new approach and computer application. Unpublished master’s thesis, Poznan University of Technology. Poznan, Poland.

\* 知乎专栏. 2022. PyCM-多分类性能评估-混淆矩阵. [online] Available at: <https://zhuanlan.zhihu.com/p/138180751> [Accessed 25 June 2022].

\* 《系统设计入门》. 2022. system-design-primer/README-zh-Hans.md at master · donnemartin/system-design-primer. [online] Available at: <https://github.com/donnemartin/system-design-primer/blob/master/README-zh-Hans.md> [Accessed 26 June 2022].

\* Faqs.ankiweb.net. 2022. What spaced repetition algorithm does Anki use? - Frequently Asked Questions. [online] Available at: <https://faqs.ankiweb.net/what-spaced-repetition-algorithm.html> [Accessed 26 June 2022].