大作业项目总结报告

题目： A

小组成员：

[已隐藏成员信息]

2023 年 7 月

一、功能简介（简述项目拥有的功能）

1. 支持注册登录功能，并且使得每个用户仪表盘所展示的图谱不同。
2. 支持多个课程来源。
3. 支持预览视频封面图片。
4. 支持课程按照不同指标进行排序。
5. 支持搜索结果缓存，使得下次搜索相同的节点不会再次加载。
6. 支持节点收藏，修改节点名、等级。
7. 支持上传下载图谱（json 格式）。
8. 支持图谱收藏（书签功能）。

二、已完成任务

必做任务完成情况（4/4）

1. 开发用于从其他课程学习网站获取课程资源的网络爬虫工具。
2. 导入 neo4j 知识图谱文件并可视化它。同时，提供一种删除现有知识图谱节点的方法。
3. 点击知识图谱中的知识节点应显示相应的课程资源，并且这些资源应根据不同的指标进行排序。
4. 实现包括注册、登录和用户仪表板在内的用户系统。知识图谱资源应根据不同用户进行区分。
5. 选做任务完成情况（3/3）

1. 在其他课程来源上进行网络爬虫。额外的课程源为：网易云课堂（study163）、学堂在线（xuetangx）、MOOC 中国（cmooc）。
2. 实现知识图谱的 CRUD（创建、读取、更新、删除）操作和书签功能。

3. 支持搜索结果缓存，使得下次搜索相同的节点不会再次加载。

4. 支持预览视频封面图片。

三、总体设计方案（请详述具体功能的实现逻辑、核心代码及具体实现图）

1.项目整体架构

本项目采用前后端分离方式进行开发，前端使用 vue.js 开发，后端使用 Django 编写。后端为前端提供对应 api 执行对数据表的操作，前端在页面操作中完成对 api 的调用。当前端用户调用爬虫数据时，后端的爬虫部分会爬取对应的网页信息，整理格式后通过 api 发送至前端进行展示。

2.爬取网页信息

根据网页请求规则来获取网页信息，具体可以分为三种：网页给予搜索 api（bilibili）、get 请求以及 post 请求。根据不同网页的规则来构造请求头以及 paras、data 等向网页发送请求，获得内容。

返回内容可能为 html 或者是 json 等文件，若为 html 则使用 BeautifulSoup 库来解析 html 格式信息从而获取课程信息，而 json 文件等则直接分析请求的返回的数据结构获取信息即可。

在实现上为了排序，我们规定所返回的都是字典类型变量。字典的 key 为排序方式(str)，value 为课程的具体数据(list)。其中 key 为 default 表示爬虫默认所获取的顺序。

在这里仅以 bilibili 网页的爬虫为代码例，其他网页的爬虫源码可以参考项目

文件：/backend/database/webe.py

```
# backend/database/bilibili.py

from database import exception
import requests
from requests import RequestException
from pypinyin import pinyin, Style

cookies = {
    'buvid3': 'CDA2F1DD-2DE9-4775-93BD-38D866B37261167611infoc'
}


def bilibili_search(keyword):
    params = {
        'keyword': keyword
    }
    url = 'https://api.bilibili.com/x/web-interface/search/all/v2'
    bilibili_default = []
    try:
        response = requests.get(url, params=params, cookies=cookies)
        html = response.json()
        content_list = html['data']['result'][11]['data']
        for data in content_list:
            courseName = str(str(data['title']).replace("<em class=\"keyword\">",
"")).replace("</em>", "")
            play = int(data['play'])
            duration = data['duration']
            author = data['author']
            imgUrl = 'http:' + data['pic']
            courseUrl = data['arcurl']
            # print(courseName, play, duration, author, courseUrl)
            bilibili_default.append({'name': courseName,
                                     'author': author,
                                     'length': duration,
                                     'plays': play,
                                     'img': imgUrl,
                                     'url': courseUrl})

    except RequestException as e:
        exception.do_exception("bilibili", keyword)
    bilibili = {
        'default': bilibili_default,
```

```
        'name': sorted(bilibili_default, key=lambda x: [pinyin(i, style=Style.TONE3) for i in x['name']]),
        'plays': sorted(bilibili_default, key=lambda x: x['plays'], reverse=True),
    }
    return bilibili


if __name__ == '__main__':
    dic = bilibili_search('计算机科学与技术')
    # Sort the results based on 'play' count in descending order
    for key in dic.keys():
        print(key + ':')
        for data in dic[key]:
            print(data)
```

3.后端数据架构

项目后端使用 Django 框架进行编写，数据表中存储的信息包含用户、知识节点、知识关联线和收藏图信息四类，共创建四个数据表，对前后端使用到的所有数据进行管理。

```
# backend/database/views.py

  class User(models.Model):
    username = models.CharField(max_length=20)
    password = models.CharField(max_length=26)


class NodeInfo(models.Model):
    knowledgeName = models.CharField(max_length=32)
    relation = models.IntegerField()
    user = models.CharField(max_length=20, null=True)
    favourite = models.IntegerField()
    time = models.DateTimeField(auto_now=True, null=True)


class Link(models.Model):
    source = models.CharField(max_length=26)
    target = models.CharField(max_length=26)
    name = models.CharField(max_length=26)
    user = models.CharField(max_length=20, null=True)
```

```
75.
76.
77. class Favourite(models.Model):
78.     favourite = models.CharField(max_length=114)
79.     username = models.CharField(max_length=514)
80.     nodecount = models.IntegerField()
81.     time = models.DateTimeField(auto_now=True, null=True)
```

4.前后端交互接口

前后端交互方面，后端将改变数据表的操作封装为函数，向前端提供可用的 api，前端通过 axios 发送 ajax 请求，对 api 进行调用，实现前后端交互。

后端提供的 url：

```
1.  # backend/database/urls.py
2.
3.  from django.conf.urls import url
4.  from database import views
5.
6.  urlpatterns = [
7.      url(r'^add_node/', views.add_node),
8.      url(r'^show_nodes/', views.show_nodes),
9.      url(r'^login_in/', views.login_in),
10.     url(r'^register/', views.register),
11.     url(r'^change_password/', views.change_password),
12.     url(r'^add_relation/', views.add_relation),
13.     url(r'^read_graph/', views.read_graph),
14.     url(r'^del_node/', views.del_node),
15.     url(r'^del_line/', views.del_line),
16.     url(r'^get_creep_content/', views.get_creep_content),
17.     url(r'^download_graph/', views.download_graph),
18.     url(r'^upload_graph/', views.upload_graph),
19.     url(r'^change_node/', views.change_nodename),
20.     url(r'^change_favourite/', views.change_favourite),
21.     url(r'^get_favourite/', views.get_favourite),
22.     url(r'^favourite_graph/', views.favourite_graph),
23.     url(r'^get_favourite_list/', views.get_favourite_list),
24.     url(r'^delete_favourite_graph/', views.delete_favourite_graph),
25.     url(r'^load_favourite/', views.load_favourite),
26.     url(r'^init_graph/', views.init_graph),
27. ]
```

前端对接口实现的调用示例：

```
// frontend/src/components/Base/Login.vue

API({
        method: 'get',
        url: '/download_graph/',
        params: {
          username: this.username
        }
      }).then(res => {
        console.log(res.data);
      })
```

5.前端项目结构

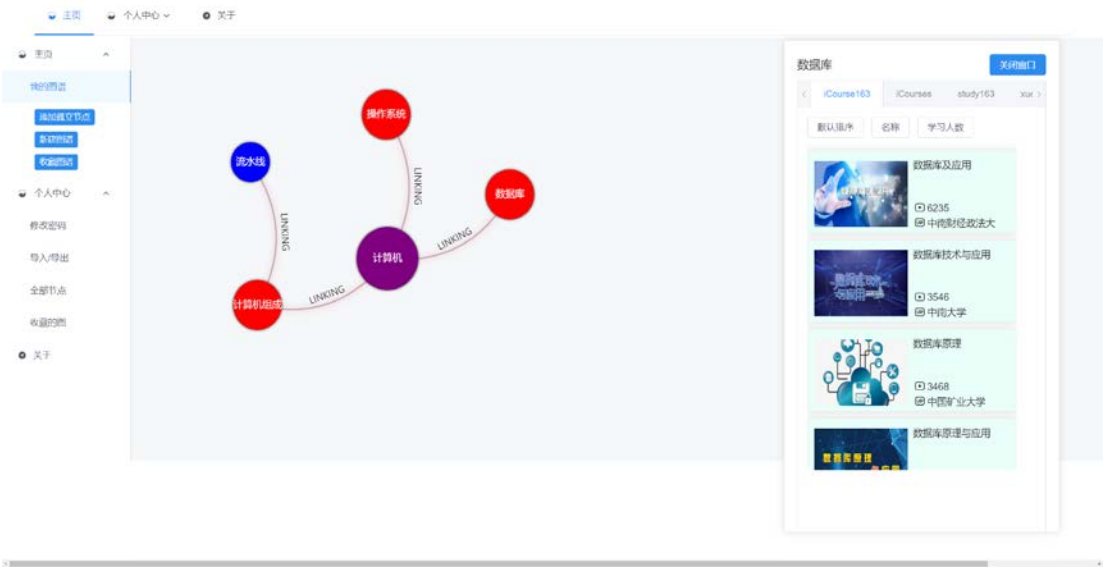
前端使用 vue.js 开发，编写多个组件页面，并通过 vue.router 进行连接，实现页面之间的跳转，跳转时传递用户名作为参数参与页面展示的过程。

```
/
├──login/          ------ 登陆页面
├──home/           ------ 主页页面
├──self
|   ├──change/     ------ 修改密码
|   ├──load/       ------ 上传下载页面
|   ├──favourite/  ------ 查看图谱节点信息
|   └──graph/      ------ 查看收藏图谱信息
└──about/          ------ 关于页面
```

6.前端图谱展示

前端使用 echarts.js 对后端返回的图谱进行展示。通过对 echarts 的设置进行标定修改其展示方式，实现不同级别知识点大小、颜色的区分，实现节点的层次化。通过接管页面的左键、右键单击事件、获取点击事件的位置信息和节点信息，实现对知识点课程的爬取+展示，或将右键独立菜单展示在合适的位置。

7.实现图



四、项目运行过程

依赖安装

```
$ pip install -r requirements
$ cd frontend
$ npm install
```

本地运行

```
$ start cmd /K "cd frontend && npm run dev" && python
backend\manage.py runserver
```

操作数据库（http://127.0.0.1:8000/admin/）

```
$ python manage.py createsuperuser
```

五、项目总结

总体完成的不错，我们先分工学习爬虫的知识并以此实践，根据 django 后端作为数据库以及爬虫功能的运行，vue 前端来实现 gui 网页，并借助 echarts 完成知识图谱的可视化。在设计上尽可能在满足功能要求的前提下让网页简洁、直观、好看。

六、课程学习总结

1、课程收获和难点分析（小组成员是否有 Python 或大作业要求的基础，做完这个大作业自我感觉是否有提高等其他收获，本次项目感觉最困难的地方在哪里）

课程收获：

小组成员基本没有 Python 的基础，不过部分成员曾有过有 django、vue 的基础。通过 Python 暑期课程以及完成本次大作业，在课程以外：我们掌握了 github 的使用技巧、团队协调工作的方法、以及提高了自己的信息获取能力。而在课程以内：我们学习到了 Python 基础的语法知识，Python 面向对象的设计构造的编程方法，Python 多线程、Python 图形库的使用方法、如何使用 django 作为后端，vue 作为前端构建一个完整的项目、以及使用 Python 爬取网站的一般信息等等，收获很多。

难点分析：

在刚接触爬虫的时候对于其规则和逻辑不是很熟悉，前期试错的过程很多，直到阅读了很多其他作者分享的爬虫实战以及知道了网站：https://curlconverter.com/可以快捷构造请求之后，才感觉逐渐对于爬虫熟悉了起来，不过仅仅以来 Python 还是无法解决所有的爬虫问题，针对于需要 JavaScript 的网页，还需要有一定的 JavaScript 知识才能解决。

首先项目选用前后端分离方式进行构建，第一个要解决的就是跨域问题。本项目采用在后端引入中间层 django-cors-headers 取消 ACAH 请求头的限制，从而使得前端可以通过直接发送 ajax 请求进行通信。

其次在后端数据表的结构上，项目起初考虑用户采用一对多的模式构建数据自己账户的节点，后期由于项目规模和实现问题采用了数据表只包含单层数据的扁平处理。但是这样的处理方式在后期功能增多时（尤其是多用户分别进行数据修改、交互时）出现了众多 bug，这说明了分离各用户数据，防止数据污染的重要性。同时随着项目功能的不断完善，最初的数据表字段已经不能满足原本的需求，添加新字段时出现了需要重构数据表和所有相关的 api 函数的问题。

前端最核心的部分是图谱展示，最开始本项目采用 d3.js 进行图谱构建，但是在处理力导图参数时设置过于复杂，难以调整出合适的表现形式，最后换用了 echarts.js 进行处理。echarts 对图表的响应事件接管上处理更直观，参数也更简练容易调节。

最后是前端 css 样式的调试问题。前端绕不开的一个问题就是项目的 css 样式表示，由于是第一次用 vue 同时开发数个页面，对工程组件的层次化处理没有把控好，某些不必要的部分独立拆分为组件后，父组件对其 css 格式的调整非常困难，同时为了用户的阅览体验也不能随意使用 absolute 进行定位，调试 css 位置也是困难的问题。

2、教师授课评价（老师上课过程的一些建议，以及希望老师之后能够介绍一些什么东西）

老师上课讲得比较细致，同时在下课时间也会用中文向我们提问是否有讲得不细致、大家不明白的地方。而授课内容我认为比较完整，同学们也能借此对 Python 的动态类型、解释型等特点有了一定的了解，暂时没有别的希望老师能够介绍的了。

3、助教评价

助教很负责，在关于作业要求实现上出现疑惑的时候，都愿意为我们详细地解答，我认为助教的工作做的是比较完善的。

4、当前课程教授内容评价与课程进一步改进建议

课程总体安排很合理，编程小作业锻炼了学生使用 Python 编程的基本功，大作业也锻炼了学生团队协作、信息获取、项目实践等能力，总的来说是一门非常不错的课程。

唯一可以提出来的建议就是对于课上作业的填空、选择题我认为题目的来源并不是很合理，有些题目还是存在着不严谨、审题不足等问题，因此我建议下学期能够针对题库进行一定的优化、改进等工作。

七、主要参考资料

https://github.com/SocialSisterYi/bilibili-API-collect

https://blog.csdn.net/qq_44862120/article/details/109002396

https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Access-Control-Allow-Headers

https://www.runoob.com/

八、项目功能实际展示视频（不超过 5min）

展示视频请查看压缩包内同级文件“演示视频-2023Python 大作业-A 题-05 组.mp4”