写在最前：本项目是一个选课智能编辑器，也即只能给出选课建议，用户仍需要在规定时间前往选课网进行选课；由于按当前时间进行“初选”、“补退选”等阶段划分不利于录制展示视频，我们可以通过在时间表中点击对应时段直接进入“初选”、“补退选”等阶段

一、 程序功能实现

(一) 智能推荐课程

- **基于神经网络推荐：**初选阶段，基于用户的个性化偏好，通过神经网络为其推荐课表，并生成推荐的备选列表

- **接入 Deepseek 智能推荐：**补退选阶段接入 Deepseek 的 api，支持更精细的选课推荐

请输入具体选课要求，如：希望选一门能练习听口能力，给分还好的英语课

- 支持**学分上限**设置
- 支持**个性化偏好**设置

（二） 自定义课表编辑

- 发挥**图形交互**的优势，**直观编辑课表**

- 支持同时编辑多张课表及课表增删
- 支持搜索课程，搜索授课教师，收藏课程，删除课程等操作
- 可以便捷查看课程详情

（三） 庞大的数据库

- 包含全校春季课程及详细信息，比如授课教师、授课地点、学分学时、任务量等等

（四） 投点建议

- 可根据输入的限选/已选人数生成投点建议

（五） 时间跟进

- 读取系统时间，预选阶段结束时自动切换至补退选阶段

二、 各模块和类细节

（一） 数据库

- 从教务网站爬取近 3000 条课程数据，形成数据库
- 爬取通用人工智能等专业的培养方案
- 后端通过 api 获取数据库中数据，将每条数据对象化，分类存入 course_list

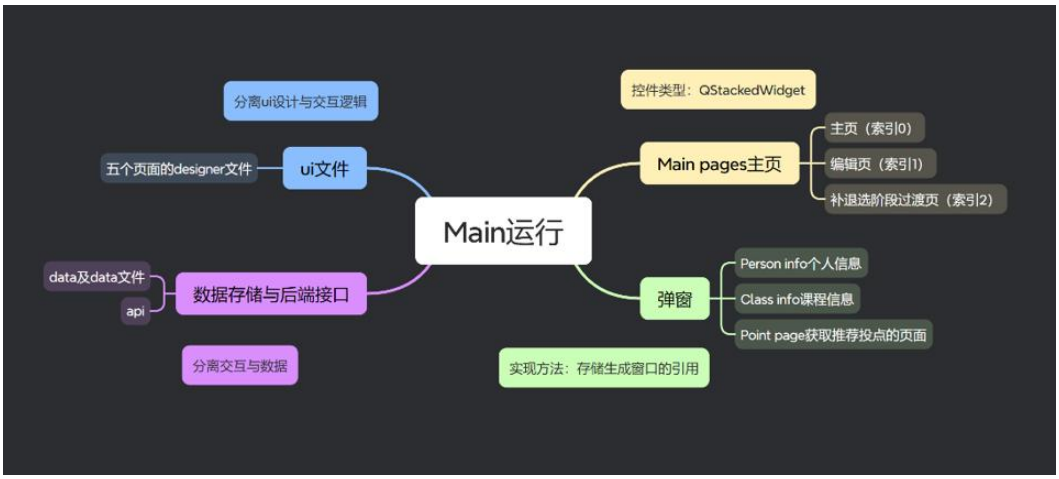
```python
# load the data
import requests
BASE_URL="http://10.7.21.186:8000"
API_PATH="/courses"
response=requests.get(f"{BASE_URL}{API_PATH}")
if response.status_code==200:
    data=response.json()

# load the curriculum, and add extra information to the existing courses
API_PATH1="/courses2"
response1=requests.get(f"{BASE_URL}{API_PATH1}") #导入培养方案课程
if response1.status_code==200:
    data1=response1.json()
```

```
course_list={'所有课程':[],'专业必修':[],'专业任选':[],'思想政治':[],'大学英语':[],'
'全校公选课':[],'通选课':[],'体育':[],'专业限选':[]}
```

(二) 前端

- 运用 pyside6 图形界面库，用 designer 工具进行界面主体设计



- 界面的显示：对每个页面创建一个类（开始页，主页，详情页，投点页），实现槽函数并连接控件

```python
class Main_pages:

    def __init__(self,app):
        self.app=app
        uiLoader = QUiLoader()
        self.ui = uiLoader.load('ui/main_pages.ui')
```

- 各种功能实现：对常用控件（时间，表格，分类树）封装方法类

```
class TimeContent:
    def __init__(self,timetable):…
    def switchStage(self):…
    def showTime(self):…

class tableMethods:
    def __init__(self,app):…
    def listToTable(self,classList,classTable,creditTag):…
    def combineCells(self,table):…
    def deCombine(self,table):…
    def goClassInformation(self,row,col,t):…

class treeMethods:
    def __init__(self):…
    def loadClassList(self,tree):…
    def filterItems(self, text,tree):…
    def filterTreeItem(self, item, text):…
```

- 学生、课程等数据：打包 person 类和 class 类

```
class Person:
    def __init__(self):…
user=Person()
user_send={}

class Class:
    def __init__(self,classbag):…
```

（三） 后端

- 课程 Course 类：包含原始字符串、上课时间、各维度评分（比如给分、任务量）、限选和已选人数等（部分数据比较敏感，因此评分、已选限选人数等采用随机生成）

```
class Course:
    def __init__(self,course_information):
        self.original=course_information #给原始字典留档

            self.time_list=[]

    self.available=100
    self.chosen=random.randint(20,400)
    self.probability=min(1,self.available/self.chosen) #选课概率
    self.evaluated=[random.randint(60,100),random.randint(40,100)] #百分制
    # 分别是给分 任务量 早八厌恶 选课难度 水分
    if [1,1] in self.time_list or [2,1] in self.time_list or [3,1] in self.time_list
```

```
or [4,1] in self.time_list or [5,1] in self.time_list or [6,1] in self.time_list or
        self.evaluated.append(-8)
    else: self.evaluated.append(0) #对早八的厌恶程度
    self.evaluated.append(1/self.probability) #选课难度
    self.evaluated.append(random.randint(40,100)) #水分
```

（四） 神经网络设计

- Step1. 根据用户专业和年级计算本学期（2024-2025 春季学期）可选的全部课程
- Step2. 使用神经网络计算本学期可选的所有课程的评分，对其进行排序
- Step3. 根据用户希望的学分上限、每类课程选择的门数上限进行筛选，并注意保证时间不冲突，最终得出推荐课表。

```python
def recommend_with_NN(preferences:Dict[str,float],must_take:List[Dict],must_not_take:List[Dict],available_courses:List[Course],expected_num:Dict[str,int]):
    priority_list=calculate_with_NN(preferences,must_take,must_not_take,available_courses)
    recommended=[]
    nums={'总学分':0,'专业任选':0,'思想政治':0,'大学英语':0,'全校公选课':0,'通选课':0,'体育':0,'专业限选':0}
    name_list=[]
    occupied_times=[] # 简单起见，只要求其他课程不能与专业必修课时间冲突
    for score,course in priority_list: #专业必修课
        if course.name in name_list: continue
        if course.type!='专业必修':
            continue
        if course.original in must_not_take:
            continue
        if nums['总学分']+course.credit>expected_num['总学分']:
            continue
        for time in course.time_list:
            if time in occupied_times:
                continue
        occupied_times.extend(course.time_list)
        name_list.append(course.name)
        recommended.append((score,course))
        nums['总学分']+=course.credit
    for score,course in priority_list: #其它课程
        if course.name in name_list: continue
        if course.credit==0: continue
        if course.type not in nums.keys():
            continue
        if nums[course.type]>=expected_num[course.type] or nums['总学分']+course.credit>expected_num['总学分']:
            continue
        for time in course.time_list:
            if time in occupied_times:
                continue
        name_list.append(course.name)
        recommended.append((score,course))
        nums['总学分']+=course.credit
        nums[course.type]+=1
    return recommended,priority_list
```

（五） 投点建议设计

- 可对需要投点的课程给出投点建议

```python
def distribute_stake(courses):
    bets=[97,61,31,2,0]
    tot=99
    ret=[]
    for available,chosen in courses:
        if tot<=5:
            ret.append(tot)
            tot=0
            continue
        probability=min(1,available/chosen)
        if probability<0.1 or probability>=0.95:
            ret.append(0)
        elif 0.1<=probability<0.4:
            for i in range(5):
                if tot>=bets[i]:
                    ret.append(bets[i])
                    tot-=bets[i]
                    break
        elif 0.4<=probability<0.7:
            for i in range(1,5):
                if tot>=bets[i]:
                    ret.append(bets[i])
                    tot-=bets[i]
                    break
        elif 0.7<=probability<0.95:
            for i in range(2,5):
                if tot>=bets[i]:
                    ret.append(bets[i])
                    tot-=bets[i]
                    break
    return ret
```

（六） Deepseek 接入

- 此阶段接入 Deepseek 的 api，让其基于已选上的课程和用户个性化偏好的文字为用户推荐补选课程。Prompt 如下

```
prompt = f"""
你是一位智能课程推荐助手。根据以下信息，从可选课程列表和可选专业课列表中为用户推荐合适的课程。

具体来说，用户在各个维度的偏好数值越高表示对该方面要求越高；

每门课的数据结构形如[记录课程信息的原始字典,[上课时间],[对应维度的课程评分],[限选人数，已选人数]]，其中对应维度的课程评分越高说明该维度表现越好。

务必注意：

推荐的课程课名不能和已选课程有重合；

推荐的课程不能和已有课程产生时间冲突；推荐的课程之间不能有时间冲突；

可选专业课列表中的课程，相同课名的能且只能选择一门；

推荐课程和已选课程学分加起来不能超过25分

可选课程列表：
{json.dumps(available_courses, indent=2)}

可选专业课列表：
{json.dumps(must_take_courses, indent=2)}

已选课程列表：
{json.dumps(chosen_courses, indent=2)}

用户个人偏好：
{json.dumps(preferences, indent=2)}

请返回推荐的课程列表，每个课程的数据结构形如：[记录课程信息的原始字典,[上课时间],[对应维度的课程评分],-1,[限选人数，已选人数],-1]
注意：不要含有其它任何信息！包括推荐理由、“根据您的需求，我将推荐以下课程”等。要求能够被解析为json格式。
"""
prompt=prompt+your_prompts
```

三、 小组成员分工（按姓名首字母排序）

董弋非：后端设计、部分报告制作

李佳燚：前端设计、展示视频录制、部分报告制作

王奕文：数据爬取与数据库制作、报告整合

四、 项目总结与反思

总结：

本项目旨在开发一个智能选课编辑器，通过神经网络与外部 API（如 Deepseek）为用户提供个性化的选课推荐。系统涵盖了课程推荐、课表编辑、课程搜索与收藏、以及投点建议等功能，结合后端数据处理与前端图形交互，提升用户在选课过程中的效率与体验。

在开发过程中，项目各部分较为完整地协作完成。后端通过爬取教务网

数据构建了包含近 3000 条课程记录的数据库，涵盖课程信息、教师、学分等关键字段。前端借助 PySide6 与 Qt Designer 构建图形界面，实现了课表展示与编辑功能，同时通过封装的类对数据和控件进行有效管理。推荐逻辑方面，神经网络能够结合用户专业、年级与偏好进行初筛，并保证时间不冲突；补退选阶段则接入 Deepseek 的 API，进一步提升推荐精度。

反思：

要在开始设计前和设计中及时明确数据库、前端、后端之间的接口格式，明确分工，以避免后期调整带来的重复工作。此外，课程评分与人数等敏感数据采用了随机生成，神经网络的训练数据量也不够庞大，这在真实场景下可能影响推荐准确性，后续若能获取更真实的交互数据，推荐模型可进一步优化。

总的来说，项目基本实现了预期功能，也提升了团队在数据处理、界面开发和 AI 应用方面的综合能力，为今后的系统设计积累了宝贵经验。