demo

传送门

联系我：微信 1257309054

点我添加微信

基于多维向量夹角求解用户同过滤推荐算法

算法讲解传送门

一、搭建项目

python解释器版本使用3.7.8。

1、创建虚拟环境

在d盘下创建一个文件夹`my_work`，然后在里面创建两个文件夹：`pro`和`venv`。

`win+R`输入`cmd`进入文件夹`venv`，然后执行以下命令创建虚拟环境：

```
python -m venv course_manager
```

激活虚拟环境:

```
cd course_manager
cd Scripts
activate
```

导入django:

```
pip install Django==3.0.7 -i https://pypi.mirrors.ustc.edu.cn/simple/
pip install PyMySQL==0.9.2 -i https://pypi.mirrors.ustc.edu.cn/simple/
pip install xadmin-django==3.0.2 -i https://pypi.mirrors.ustc.edu.cn/simple/
pip install mysqlclient==2.0.1 -i https://pypi.mirrors.ustc.edu.cn/simple/
pip install numpy==1.21.6 -i https://pypi.mirrors.ustc.edu.cn/simple/

```

然后进入`pro`目录

```
cd ..
cd ..
cd ..
cd pro
```

2、创建项目

执行命令：

```
django-admin startproject  course_manager
```

3、创建子应用

切换到项目根目录：

```
cd course_manager
```

创建子应用:

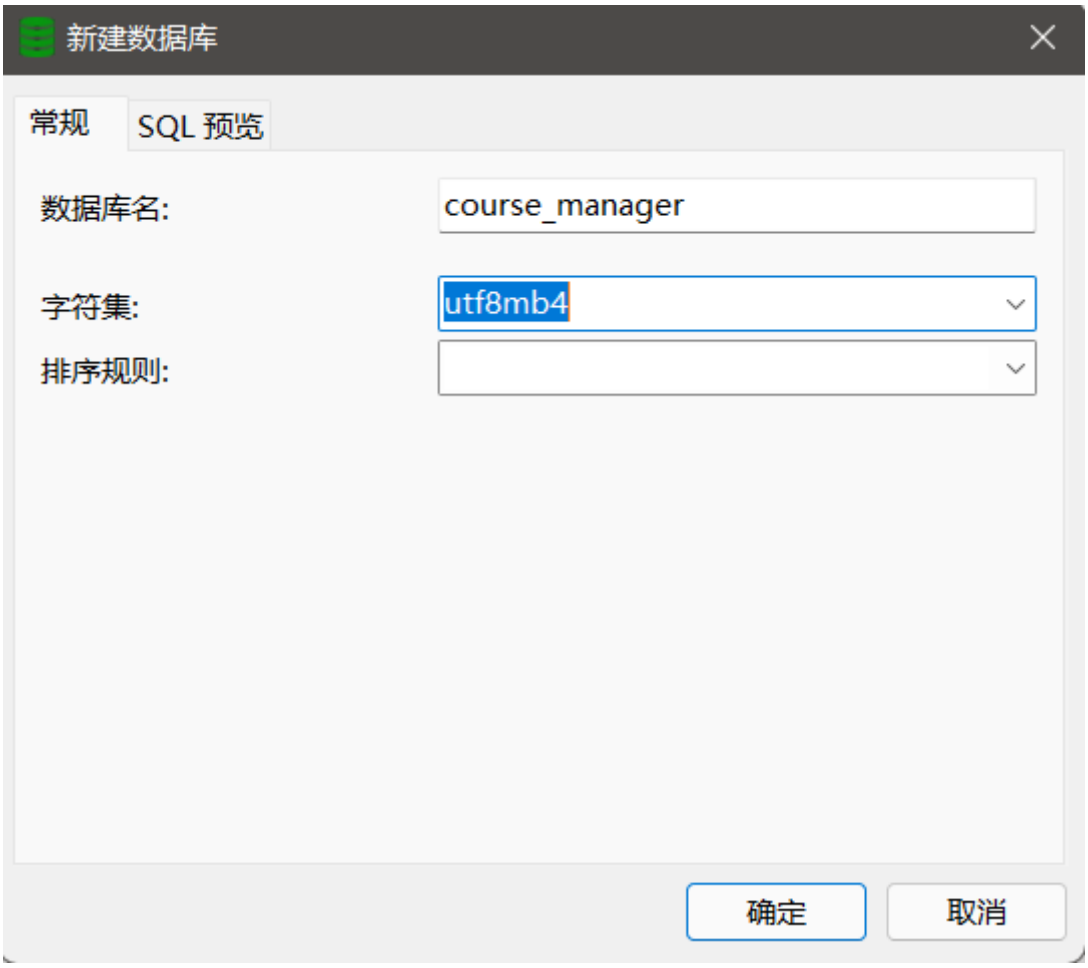
```
python manage.py startapp course
```

自此项目创建完成。

二、settings.py配置

1、创建数据库

使用MySQL可视化工具创建一个数据库`course_manager`。



2、PyCharm打开项目

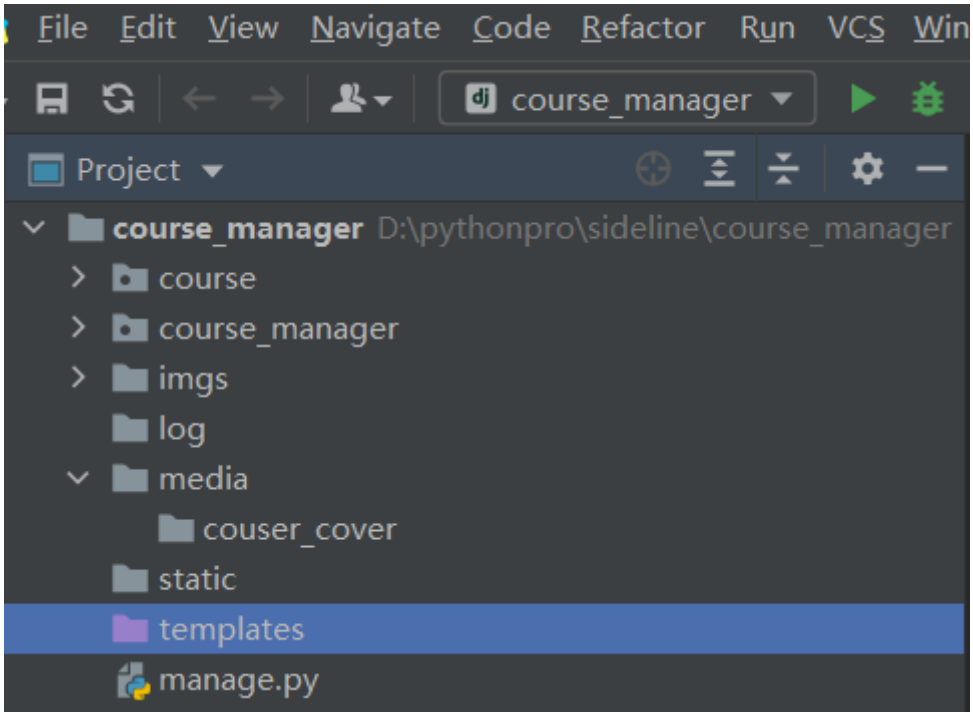
使用PyCharm打开项目：`file->open`.

在项目根目录下创建以下文件夹：

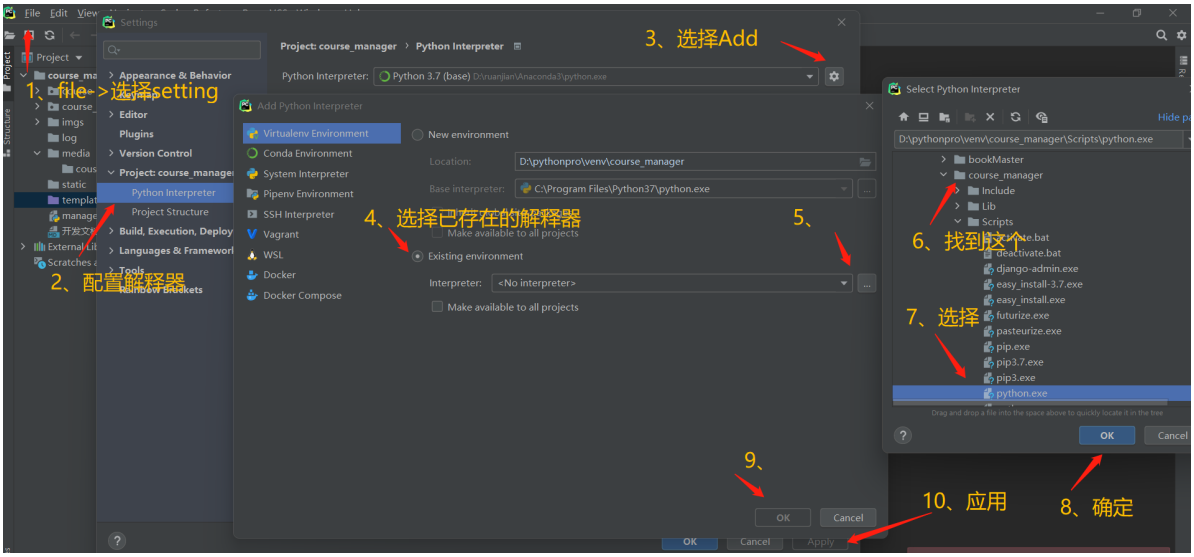
`static、media、imgs、log、templates`。

其中`media`中再创建一个文件夹`course_cover`存放课程封面。

选中`templates`->右键->`Make Directory as`->`Template Folder`.



3、配置项目虚拟环境



4、允许所有网站访问

在`course_manager\settings.py`中做修改：

```
ALLOWED_HOSTS = ['*']
```

5、添加子应用

在`course_manager\settings.py`中的`INSTALLED_APPS`加入子应用`course`，如下：

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'xadmin',
    'crispy_forms',
    'course'
]
```

6、添加templates目录

在`course_manager\settings.py`中的TEMPLATES中`DIRS`改为：

```
'DIRS': [os.path.join(BASE_DIR, 'templates')],
```

7、使用mysql数据库

把在`course_manager\settings.py`中的DATABASES注释掉，改为：

```
ip = '127.0.0.1'
DATABASE_NAME = 'course_manager'  # mysql数据库名称
DATABASE_USER = 'root'  # mysql数据库用户名
DATABASE_PASS = 'ldc-root'  # mysql数据库密码
DATABASE_HOST = ip  # mysql数据库IP
DATABASE_PORT = 3306  # mysql数据库端口

# 配置数据库
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',  # 修改数据库为MySQL，并进行配置
        'NAME': DATABASE_NAME,  #
        'USER': DATABASE_USER,  # 用户名
        'PASSWORD': DATABASE_PASS,  # 密码
        'HOST': DATABASE_HOST,
        'PORT': DATABASE_PORT,
        'OPTIONS': {'charset': 'utf8mb4', }
    }
```

```
}
```

8、使用中文

把`course_manager\settings.py`的LANGUAGE_CODE、TIME_ZONE和USE_TZ改为：

```
LANGUAGE_CODE = 'zh-hans'  # 使用中国时区

TIME_ZONE = 'Asia/Shanghai'

USE_I18N = True

USE_L10N = True

USE_TZ = False
```

9、配置静态文件路由

把`course_manager\settings.py`的STATIC_URL改为：

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'static'),
]
# STATIC_ROOT = os.path.join(BASE_DIR, 'static')  # 收集静态文件时打开，然后关闭 STATICFILES_DIRS

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_USER_ICON = os.path.join(BASE_DIR, 'media/user_icon')
```

三、models.py数据表

在`course\models.py`中创建用户表、课程类别表、课程表、用户选课表、评分表、收藏表、点赞表、评论表：

```
from django.db import models

# 用户表
class User(models.Model):
    username = models.CharField(max_length=255, unique=True, verbose_name="账号")
    password = models.CharField(default='123456', max_length=32, verbose_name="密码")

    class Meta:
        db_table = 'user'
        verbose_name_plural = "用户"
        verbose_name = "用户"

```

```
    def __str__(self):
        return self.username

# 课程类别表
class Tags(models.Model):
    name = models.CharField(max_length=200, verbose_name="类别")

    class Meta:
        db_table = 'tags'
        verbose_name = "课程类别"
        verbose_name_plural = "课程类别"

    def __str__(self):
        return self.name

# 课程表
class CourseInfo(models.Model):
    name = models.CharField(verbose_name="课程名", max_length=255)
    course_code = models.CharField(unique=True, verbose_name="课程编号", max_length=255)
    teacher = models.CharField(verbose_name="讲师", max_length=255)
    about = models.TextField(verbose_name="描述")
    pic = models.FileField(verbose_name="封面图片", max_length=64, upload_to='course_cover')
    tags = models.ForeignKey(
        Tags,
        on_delete=models.CASCADE,
        verbose_name="类别",
        related_name="tags",
        blank=True,
        null=True,
    )
    select_num = models.IntegerField(verbose_name="选修人数", default=0)
    look_num = models.IntegerField(verbose_name="浏览人数", default=0)
    collect_num = models.IntegerField(verbose_name="收藏人数", default=0)
    course_url = models.TextField(verbose_name="课程网址", default='https://www.xuetangx.com/')
    add_time = models.DateTimeField(verbose_name="创建时间", auto_now_add=True)

    class Meta:
        db_table = 'course_info'
        verbose_name = "课程"
        verbose_name_plural = "课程"

    def __str__(self):
        return self.name

# 用户选课表
class UserCourse(models.Model):
    course = models.ForeignKey(
        CourseInfo, to_field='course_code', on_delete=models.CASCADE, blank=True, null=True, verbose_name="课程id"
    )
    user = models.ForeignKey(
        User, on_delete=models.CASCADE, blank=True, null=True, verbose_name="用户id",
    )
```

```
    enroll_time = models.DateTimeField(verbose_name="选修时间", auto_now_add=True)

    class Meta:
        db_table = 'user_course'
        verbose_name = "用户选课信息"
        verbose_name_plural = verbose_name

# 评分表
class RateCourse(models.Model):
    course = models.ForeignKey(
        CourseInfo, to_field='course_code', on_delete=models.CASCADE, related_name='rate_course', blank=True, null=True,
        verbose_name="课程id"
    )
    user = models.ForeignKey(User, on_delete=models.CASCADE, blank=True, null=True, verbose_name="用户id")
    mark = models.FloatField(verbose_name="评分")
    create_time = models.DateTimeField(verbose_name="添加时间", auto_now_add=True)

    class Meta:
        db_table = 'rate_course'
        verbose_name = "评分信息"
        verbose_name_plural = verbose_name

# 收藏表
class CollectCourse(models.Model):
    course = models.ForeignKey(
        CourseInfo, to_field='course_code', on_delete=models.CASCADE, related_name='collect_course', blank=True,
        null=True,
        verbose_name="课程id"
    )
    user = models.ForeignKey(
        User, on_delete=models.CASCADE, blank=True, null=True, verbose_name="用户id",
    )
    is_delete = models.BooleanField(default=False, verbose_name='是否取消')
    create_time = models.DateTimeField(verbose_name="收藏时间", auto_now_add=True)

    class Meta:
        db_table = 'collect_course'
        verbose_name = "课程收藏"
        verbose_name_plural = verbose_name

# 评论表
class CommentCourse(models.Model):
    course = models.ForeignKey(
        CourseInfo, to_field='course_code', on_delete=models.CASCADE, related_name='comment_course', blank=True,
        null=True,
        verbose_name="课程id"
    )
    user = models.ForeignKey(User, on_delete=models.CASCADE, blank=True, null=True, verbose_name="用户")
    content = models.TextField(verbose_name="评论内容")
```

```
    create_time = models.DateTimeField(verbose_name='评论时间', auto_now_add=True)
    like_num = models.IntegerField(verbose_name="点赞数", default=0)
    like_users = models.TextField(null=True, blank=True, default=None, verbose_name="点赞用户")
    is_show = models.BooleanField(default=True, verbose_name='是否显示')

    class Meta:
        db_table = 'comment_course'
        verbose_name = "课程评论"
        verbose_name_plural = verbose_name
```

四、urls.py路由配置

1、修改course_manager\urls.py

在`course`下创建一个`urls.py`，并让`course_manager\urls.py`分配路由，

其中`course_manager\urls.py`改为：

```
import xadmin
from django.urls import path, re_path, include
from django.views.generic import RedirectView
from django.views.static import serve
from django.conf import settings

urlpatterns = [
    path('xadmin/', xadmin.site.urls),
    path("", include("course.urls")),
    # favicon.cio
    re_path(r'^favicon\.ico$', RedirectView.as_view(url=r'media/favicon.ico')),
    re_path(r'^media/(?P<path>.*)$', serve, {'document_root': settings.MEDIA_ROOT}),
    # re_path(r'^static/(?P<path>.*)$', serve, {'document_root': settings.STATICFILES_DIRS}), # 收集静态文件时关闭
    path(r'^static/(?P<path>.*)$', serve, {'document_root': settings.STATIC_ROOT}), # 收集静态文件时打开，然后关闭STATICFILES_DIRS
]

```

2、修改course\urls.py

先改为：

```
from django.urls import path, re_path

from course import views

urlpatterns = [

]
```

后面会添加各种路由。

3、数据迁移

在pycharm左下角的Terminal里执行数据迁移命令

```
python manage.py makemigrations
python manage.py migrate
```

4、创建缓存表

```
python manage.py createcachetable
```

5、收集静态文件

先把`course_manager\settings.py`中的静态文件路由改为：

```
STATIC_URL = '/static/'
# STATICFILES_DIRS = [
#     os.path.join(BASE_DIR, 'static'),
# ]
STATIC_ROOT = os.path.join(BASE_DIR, 'static')  # 收集静态文件时打开，然后关闭STATICFILES_DIRS
```

然后执行：

```
python manage.py collectstatic
```

执行成功后，把`course_manager\settings.py`中的静态文件路由改为：

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'static'),
]
# STATIC_ROOT = os.path.join(BASE_DIR, 'static')  # 收集静态文件时打开，然后关闭STATICFILES_DIRS
```

把`course_manager\urls.py`改为：

```
import xadmin
from django.urls import path, re_path, include
from django.views.generic import RedirectView
from django.views.static import serve
from django.conf import settings

urlpatterns = [
    path('xadmin/', xadmin.site.urls),
    path("", include("course.urls")),
```

```
    # favicon.cio
    re_path(r'^favicon\.ico$',
RedirectView.as_view(url=r'media/favicon.ico')),
    re_path(r'^media/(?P<path>.*)$', serve, {'document_root':
settings.MEDIA_ROOT}),
    re_path(r'^static/(?P<path>.*)$', serve, {'document_root':
settings.STATICFILES_DIRS}), # 收集静态文件时关闭
    # path(r'^static/(?P<path>.*)$', serve, {'document_root':
settings.STATIC_ROOT}), # 收集静态文件时打开，然后关闭STATICFILES_DIRS
]
```

6、创建后台管理员

```
python manage.py createsuperuser
```

```
设置账号为 root
邮箱为 1@qq.com
密码为 course-root
```

五、导入基础数据

把根目录下的`course.sql`在mysql可视化工具中执行即可。

六、核心代码

1、static创建文件夹

在`static`目录下创建三个文件夹`image`、`css`、`js`和`fonts`用来存放前端需要使用到的文件。

2、base.html前端框架

在目录`templates`下创建前端页面框架`base.html`，代码如下：

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- 上述3个meta标签*必须*放在最前面，任何其他内容都*必须*跟随其后！ -->
    <meta name="description" content="">
    <meta name="author" content="">
    <link rel="icon" href="/media/course.png">
    <title>在线课程推荐系统</title>
    {% block style %}
    {% endblock %}
    <!-- Bootstrap core CSS -->
    <link href="/static/css/bootstrap.min.css" rel="stylesheet">
```

```
    <!-- Custom styles for this template -->
    <link href="/static/css/dashboard.css" rel="stylesheet">
    <link href="/static/css/custom.css" rel="stylesheet">
    {% block extrastyle %}
    {% endblock %}
    <!-- Just for debugging purposes. Don't actually copy these 2 lines! -->
    <!--[if lt IE 9]>
    <script src="../../assets/js/ie8-responsive-file-warning.js"></script>
<![endif]-->
    <script src="/static/js/ie-emulation-modes-warning.js"></script>
    <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->
    <!--[if lt IE 9]>
    <script src="/static/js/html5shiv.min.js"></script>
    <script src="/static/js/respond.min.js"></script>
    <![endif]-->

</head>

<body>
<nav class="navbar navbar-inverse navbar-fixed-top">
    <div class="container-fluid">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar"
                    aria-expanded="false" aria-controls="navbar">
                <span class="sr-only">Toggle navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="/">课程推荐系统</a>
        </div>
        <div id="navbar" class="navbar-collapse collapse">
            <ul class="nav navbar-nav navbar-right">
                {% if request.session.login_in == True %}
                    <li><a href="{% url 'personal' %}">{{ request.session.name }}</a></li>
                    <li><a href="{% url 'logout' %}">退出</a></li>
                {% else %}
                    <li><a href="{% url 'login' %}">登录</a></li>
                    <li><a href="{% url 'register' %}">注册</a></li>
                {% endif %}
            </ul>
            <form class="navbar-form navbar-right" action="{% url 'search' %}" method='post'>
                {% csrf_token %}
                <label for="search"></label>
                <input id="search" type="text" class="form-control" name="search" placeholder="输入关键字"/>
                <button class="btn btn-default" type="submit">提交</button>
            </form>
        </div>
    </div>
</nav>
{% block content-nav %}{% endblock %}
<div class="container-fluid">
```

```
    <div class="row" >
        <div class="col-sm-3 col-md-2 sidebar">
            <ul class="nav nav-sidebar">
                <li class="active"><a href="{% url 'all_course' %}">全部课程
<span class="sr-only">(current)</span></a></li>
                <li><a href="{% url 'new_course' %}">新课速递</a></li>
                <li><a href="{% url 'hot_course' %}">热门课程</a></li>
                <li><a href="{% url 'sort_course' %}">课程分类</a></li>
                <li><a href="{% url 'recommend_course' %}">猜你喜欢</a></li>
                <li><a href="{% url 'personal' %}">个人中心</a></li>
            </ul>
        </div>
        <div class="col-sm-9 col-sm-offset-3 col-md-10 col-md-offset-2
main" style="margin-right:0;padding-right:0;">
            {% block right-panel-content %}
            {% endblock %}
        </div>
    </div>
</div>


<script src="/static/js/jquery-2.1.1.min.js"></script>
<script src="/static/js/jquery.min.js"></script>
<script src="/static/js/bootstrap.min.js"></script>
<script src="/static/js/ie10-viewport-bug-workaround.js"></script>
<script src="/static/js/custom.js"></script>
<script src="/static/js/plugins/highstock/js/highstock.js"></script>
<script src="/static/js/plugins/highstock/js/modules/exporting.js">
</script>
<script type="text/javascript">
    window.__user_media_prefix__ = "/media/";
    window.__user_path_prefix__ = "";
    window.__user_language_code__ = "";
    $(function ($) {
        {#    导航栏按钮渲染#}
        $(".sidebar").find("li").each(function () {
            var a = $(this).find("a:first")[0];
            if ($(a).attr("href") === location.pathname) {
                $(this).addClass("active");
            } else {
                $(this).removeClass("active");
            }
        });
    });
</script>
{% block bottom-js %}
{% endblock %}
</body>
</html>
```

3、course/urls.py创建基础路由

`course/urls.py`创建搜索、全部课程、新课速递、热门课程、课程分类、猜你喜欢、个人中心等路由，代码如下：

```
# !/usr/bin/python
# -*- coding: utf-8 -*-

from django.urls import path

from course import views

urlpatterns = [
    path("", views.index, name="index"),  # 首页
    path("login/", views.login, name="login"),  # 登录
    path("register/", views.register, name="register"),  # 注册
    path("logout/", views.logout, name="logout"),  # 退出
    path("modify_pwd/", views.modify_pwd, name="modify_pwd"),  # 修改密码
    path("search/", views.search, name="search"),  # 搜索
    path("all_course/", views.all_course, name="all_course"),  # 所有课程
    path("course/<int:course_id>/", views.course, name="course"),  # 具体的课程
    path("select/<int:course_id>/", views.select_course, name="select"),  # 选修课程
    path("score/<int:course_id>/", views.score, name="score"),  # 评分
    path("comment/<int:course_id>/", views.comment, name="comment"),  # 评论
    path("comment_like/<int:comment_id>/", views.comment_like, name="comment_like"),  # 给评论点赞
    path("collect/<int:course_id>/", views.collect, name="collect"),  # 收藏
    path("new_course/", views.new_course, name="new_course"),  # 新课速递
    path("hot_course/", views.hot_course, name="hot_course"),  # 热门课程
    path("sort_course/", views.sort_course, name="sort_course"),  # 课程分类
    path("recommend_course/", views.recommend_course, name="recommend_course"),  # 猜你喜欢
    path("personal/", views.personal, name="personal"),  # 个人中心
    path("my_select/", views.my_select, name="my_select"),  # 获取我的选修
    path("my_collect/", views.my_collect, name="my_collect"),  # 获取我的收藏
    path("my_rate/", views.my_rate, name="my_rate"),  # 我打分过的课程
    path("my_comments/", views.my_comments, name="my_comments"),  # 我的评论
    path("delete_rate/<int:rate_id>", views.delete_rate, name="delete_rate"),  # 取消评分
    path("delete_comment/<int:comment_id>", views.delete_comment, name="delete_comment"),  # 取消评论

]
```

在course\views.py中为每个路由创建响应。

4、登录

```
def login(request):
    if request.method == "GET":
        # get请求说明是用户进入登录界面
        form = Login()
        return render(request, "login.html", {"form": form})
```

```
    # post请求说明是用户输入账号与密码，发起登录请求
    form = Login(request.POST)
    if form.is_valid():
        # 验证用户的登录form是否有效
        username = form.cleaned_data["username"]  # 账号
        password = form.cleaned_data["password"]  # 密码
        result = User.objects.filter(username=username)  # 通过账号获取用户信息
        if result:
            user = result.first()
            # 验证密码是否正确
            if user.password == password:
                request.session["login_in"] = True
                request.session["user_id"] = user.id  # 把用户Id写进浏览器session
                return redirect(reverse("all_course"))  # 跳转到首页
            else:
                return render(request, "login.html", {"form": form, "error": "账号或密码错误"})
        else:
            return render(request, "login.html", {"form": form, "error": "账号不存在"})
```

5、注册

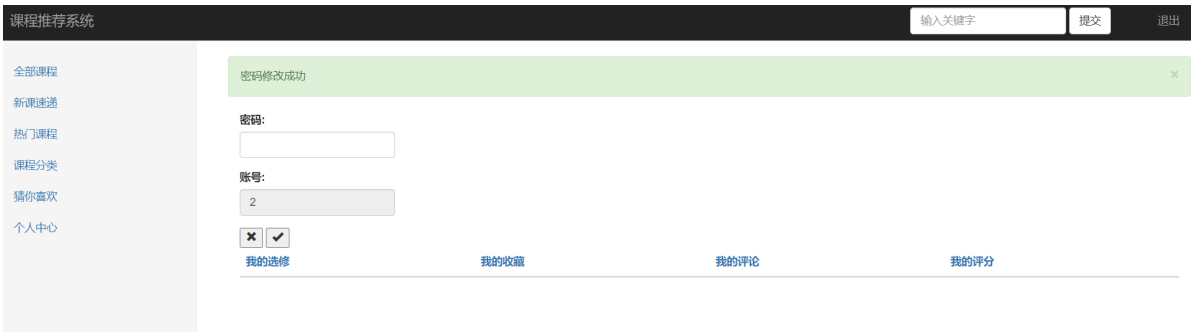
```
def register(request):
    if request.method == "GET":
        # get请求说明用户进入了注册页面
        form = RegisterForm()
        return render(request, "register.html", {"form": form})
    # post请求说明用户输入了账号、密码发起了注册请求
    form = RegisterForm(request.POST)
    error = None
    if form.is_valid():
        # 验证账号是否已经存在、两次密码是否一致
        username = form.cleaned_data["username"]
        password = form.cleaned_data["password2"]
        User.objects.create(username=username, password=password) # 创建用户
        return redirect(reverse("login"))  # 跳转到登录界面
    else:
        return render(request, "register.html", {"form": form, "error": error})  # 表单验证失败返回一个空表单到注册页面
```

6、登出

```
def logout(request):
    if not request.session.get("login_in", None):  # 不在登录状态跳转回首页
        return redirect(reverse("index"))
    request.session.flush()  # 清除session信息
    return redirect(reverse("index"))
```

7、修改密码

```
@login_in
def modify_pwd(request):
    # 获取我的信息
    user = User.objects.get(id=request.session.get("user_id"))
    if request.method != "POST":
        return render(request, '404.html')
    form = Edit(instance=user, data=request.POST)
    if form.is_valid():
        form.save()
        return render(request, "personal.html", {"inform_message": "密码修改成功", "inform_type": "success", "form": form})
    else:
        return render(request, "personal.html", {"inform_message": "密码修改失败", "inform_type": "danger", "form": form})
```



8、搜索

```python
def search(request):  # 搜索
    if request.method == "POST":  # 如果搜索界面
        key = request.POST["search"]
        request.session["search"] = key  # 记录搜索关键词解决跳页问题
    else:
        key = request.session.get("search")  # 得到关键词
    # 课程名称、简介、老师模糊搜索
    courses = CourseInfo.objects.filter(Q(name__icontains=key) | Q(about__icontains=key) | Q(teacher__icontains=key))
    page_num = request.GET.get("page", 1)
    courses = courses_paginator(courses, page_num)
    return render(request, "all_course.html", {"courses": courses})
```

9、所有课程

```python
def all_course(request):
    # 按收藏数进行排序
    courses = CourseInfo.objects.all().order_by('-collect_num') # 获取所有课程，按收藏量进行降序排序
    paginator = Paginator(courses, 9) # 对查询结果进行分页，一页显示9门课程
    current_page = request.GET.get("page", 1)
    courses = paginator.page(current_page)
    return render(request, "all_course.html", {"courses": courses, "title": "所有课程"})
```

10、具体的课程

```python
def course(request, course_id):
    # 获取具体的课程
    user_id = request.session.get("user_id")
    course = CourseInfo.objects.get(pk=course_id)
    course.look_num += 1  # 增加浏览量
    course.save()
    comments = course.comment_course.filter(is_show=True).order_by("-create_time")  # 获取可展示的评论，按时间降序
    rate = RateCourse.objects.filter(course=course).aggregate(Avg("mark")).get("mark__avg", 0)  # 获取该课程的平均评分
```

```
    course_rate = round(rate, 2) if rate else 0  # 获取评分
    if user_id:
        # 用户已经登录
        user = User.objects.get(pk=user_id)
        is_collect = True if course.collect_course.filter(user_id=user_id, is_delete=False) else False  # 判断用户是否收藏
        is_rate = RateCourse.objects.filter(course=course, user=user).first()  # 判断用户是否评分
        # 判断用户是否选修
        is_select = True if UserCourse.objects.filter(user_id=user_id, course_id=course.course_code) else False
    return render(request, "course.html", locals())

```

11、选修课程

```
@login_in
def select_course(request, course_id):
    # 用户选修课程
    course = CourseInfo.objects.get(pk=course_id)
    user_id = request.session.get("user_id")
    user = User.objects.get(pk=user_id)
    if not UserCourse.objects.filter(user_id=user_id, course_id=course.course_code):
        # 创建一条用户选修课程记录
        UserCourse.objects.create(course=course, user=user)
        course.select_num += 1  # 增加选修人数
        course.save()
    return redirect(course.course_url)  # 跳转到课程观看页面
```

12、评分

```
@login_in
def score(request, course_id):
    user_id = request.session.get("user_id")
    user = User.objects.get(id=user_id)

```

```
    course = CourseInfo.objects.get(id=course_id)
    score = float(request.POST.get("score", 0))  # 获取评分
    is_rate = RateCourse.objects.filter(course=course, user=user).first()
    if not is_rate:
        # 用户未评分则创建一条评分记录
        RateCourse.objects.get_or_create(user=user, course=course, defaults={"mark": score})
        is_rate = {'mark': score}
    comments = course.comment_course.filter(is_show=True).order_by("-create_time")  # 获取可展示的评论，按时间降序
    rate = RateCourse.objects.filter(course=course).aggregate(Avg("mark")).get("mark__avg", 0)  # 获取平均评分
    course_rate = round(rate, 2) if rate else 0  # 取评分两位小数
    is_collect = True if course.collect_course.filter(user_id=user_id, is_delete=False) else False  # 判断是否收藏
    # 判断是否选修
    is_select = True if UserCourse.objects.filter(user_id=user_id, course_id=course.course_code) else False
    return render(request, "course.html", locals())
```

13、评论

```
@login_in
def comment(request, course_id):
    # 评论
    user_id = request.session.get("user_id")
    user = User.objects.get(id=user_id)
    course = CourseInfo.objects.get(id=course_id)
    comment = request.POST.get("comment", "")  # 获取评论内容
    CommentCourse.objects.create(user=user, course=course, content=comment)  # 创建评论记录
    comments = course.comment_course.filter(is_show=True).order_by("-create_time")  # 获取可展示的评论，按时间降序
    rate = RateCourse.objects.filter(course=course).aggregate(Avg("mark")).get("mark__avg", 0)  # 获取平均评分
    course_rate = round(rate, 2) if rate else 0  # 取评分两位小数
    is_collect = True if course.collect_course.filter(user_id=user_id, is_delete=False) else False  # 判断是否收藏
    is_rate = RateCourse.objects.filter(course=course, user=user).first()  # 判断是否评分过
    # 判断是否选修
    is_select = True if UserCourse.objects.filter(user_id=user_id, course_id=course.course_code) else False
    return render(request, "course.html", locals())
```

14、给评论点赞

```python
@login_in
def comment_like(request, comment_id):
    user_id = request.session.get("user_id")
    user = User.objects.get(id=user_id)
    comment = CommentCourse.objects.get(id=comment_id)
    if not comment.like_users:
        # 还没有用户点过赞
        comment.like_users = '{},'.format(user_id)  # 添加用户点赞记录
        comment.like_num += 1
    elif str(user_id) not in comment.like_users.split(','):
        # 用户未给该评论点过赞
        comment.like_users += '{},'.format(user_id)  # 添加用户点赞记录
        comment.like_num += 1
    else:
        pass

    comment.save()
    course = comment.course
    comments = course.comment_course.filter(is_show=True).order_by("-create_time")  # 获取可展示的评论，按时间降序
    rate = RateCourse.objects.filter(course=course).aggregate(Avg("mark")).get("mark__avg", 0)  # 获取平均评分
    course_rate = round(rate, 2) if rate else 0  # 取评分两位小数
    is_collect = True if course.collect_course.filter(user_id=user_id, is_delete=False) else False  # 判断是否收藏
    is_rate = RateCourse.objects.filter(course=course, user=user).first()  # 判断是否评分过
    # 判断是否选修
    is_select = True if UserCourse.objects.filter(user_id=user_id, course_id=course.course_code) else False
    return render(request, "course.html", locals())
```

15、收藏

```python
@login_in
def collect(request, course_id):
    user_id = request.session.get("user_id")
    user = User.objects.get(id=user_id)
    course = CourseInfo.objects.get(id=course_id)
    collects = course.collect_course.filter(user_id=user_id)
    # 判断用户是否已收藏
    if collects:
        collect_ = collects.first()
        if collect_.is_delete:
            # 已取消收藏，再次点击改为已收藏
            is_collect = True
            collect_.is_delete = False
            collect_num_ = 1
        else:
            # 已收藏，再次点击改为取消收藏
            is_collect = False
            collect_.is_delete = True
            collect_num_ = -1
        collect_.save()
    else:
        # 未存在收藏书籍，创建收藏记录
        CollectCourse.objects.create(course=course, user=user)
        is_collect = True
        collect_num_ = 1

    course.collect_num += collect_num_  # 修改收藏人数
    course.save()
    comments = course.comment_course.filter(is_show=True).order_by("-create_time")  # 获取可展示的评论，按时间降序
    rate = RateCourse.objects.filter(course=course).aggregate(Avg("mark")).get("mark__avg", 0)  # 获取平均评分
    course_rate = round(rate, 2) if rate else 0  # 取评分两位小数
    is_collect = True if course.collect_course.filter(user_id=user_id, is_delete=False) else False  # 判断是否收藏
    is_rate = RateCourse.objects.filter(course=course, user=user).first()  # 判断是否评分过
    # 判断是否选修
    is_select = True if UserCourse.objects.filter(user_id=user_id, course_id=course.course_code) else False
    return render(request, "course.html", locals())
```

16、新课速递

```python
def new_course(request):
    page_number = request.GET.get("page", 1)
    # 按创建时间排序，取前10门课程
    courses = courses_paginator(CourseInfo.objects.all().order_by("-add_time")[:10], page_number)
    return render(request, "all_course.html", {"courses": courses, "title": "新课速递"})
```

17、热门课程

```
def hot_course(request):
    page_number = request.GET.get("page", 1)
    # 按收藏量查询，降序，取前10门课程
    courses = CourseInfo.objects.all().order_by('-collect_num')[:10]
    courses = courses_paginator(courses, page_number)
    return render(request, "all_course.html", {"courses": courses, "title":
"热门课程"})
```

18、课程分类

```
def sort_course(request):
    sort_id = request.GET.get('sort_id', '1')  # 获取选择的分类菜单栏id
    tracks = []  # 分类菜单栏
    title = None
    for tag in Tags.objects.all():
        if int(sort_id) == tag.id:
            title = tag.name
            class_name = 'btn_select_track'
        else:
            class_name = 'btn_grey_track'
        tracks.append({
            'name': tag.name,
            'href': '/sort_course/?sort_id={}'.format(tag.id),
            'class_name': class_name
        })
    # 按收藏量进行排序
    courses = CourseInfo.objects.filter(tags_id=sort_id).order_by('-
collect_num')
    paginator = Paginator(courses, 9)
    current_page = request.GET.get("page", 1)
    courses = paginator.page(current_page)
    return render(request, "course_sort.html", {"courses": courses, "title":
"{}类课程".format(title), "tracks": tracks})
```

19、猜你喜欢

```
@login_in
def recommend_course(request):
    page = request.GET.get("page", 1)
    courses =
courses_paginator(recommend_by_user_id(request.session.get("user_id")), page)
    path = request.path
    title = "猜你喜欢"
    return render(request, "all_course.html", {"courses": courses, "path":
path, "title": title})
```

20、推荐算法

在项目根目录下的`recommend_courses.py`文件中：

```
# -*-coding:utf-8-*-
import math
import os
import django
import operator
import numpy as np
from course.models import *


os.environ["DJANGO_SETTINGS_MODULE"] = "course_master.settings"
django.setup()


class UserCf:
    # 基于用户协同算法来获取推荐列表
    """
    利用用户的群体行为来计算用户的相关性。
    计算用户相关性的时候我们就是通过对比他们选修过多少相同的课程相关度来计算的
    举例：
    --------+--------+--------+--------+--------+
            |   X    |    Y   |    Z   |    R   |
    --------+--------+--------+--------+--------+
        a   |   1    |    1   |    1   |    0   |
    --------+--------+--------+--------+--------+
        b   |   1    |    0   |    1   |    0   |
    --------+--------+--------+--------+--------+
```

```
        c   |   1   |   1   |   0   |   1   |
    --------+-------+-------+-------+-------+

    a用户选修了：X、Y、Z
    b用户选修了：X、Z
    c用户选修了：X、Y、R

    那么很容易看到a用户和b、c用户非常相似，给a用户推荐课程R，
    给b用户推荐课程Y
    给c用户推荐课程Z
    这就是基于用户的协同过滤。
    a用户向量为(1,1,1,0)
    b用户向量为(1,0,1,0)
    c用户向量为(1,1,0,1)
    找a用户的相似用户，则计算a向量与其他向量的夹角即可，夹角越小则说明越相近
    利用求高维空间向量的夹角,可以估计两组数据的吻合程度
    """

    # 获得初始化数据
    def __init__(self, data):
        self.data = data

    # 计算N维向量的夹角
    def calc_vector_cos(self, a, b):
        '''
        cos=(ab的内积)/(|a||b|)
        :param a: 向量a
        :param b: 向量b
        :return: 夹角值
        '''
        a_n = np.array(a)
        b_n = np.array(b)
        if any(b_n) == 0:
            return 0
        cos_ab = a_n.dot(b_n) / (np.linalg.norm(a_n) * np.linalg.norm(b_n))
        return round(cos_ab, 2)


    # 计算与当前用户的距离，获得最临近的用户
    def nearest_user(self, username, n=1):
        distances = {}
        # 用户，相似度
        # 遍历整个数据集
        for user, rate_set in self.data.items():
            # 非当前的用户
            if user != username:
                vector_a = tuple(self.data[username].values())
                vector_b = tuple(self.data[user].values())
                distance = self.calc_vector_cos(vector_a, vector_b)
                # 计算两个用户的相似度
                distances[user] = distance
        # 排序，按向量夹角由小到到排序
        closest_distance = sorted(distances.items(),
key=operator.itemgetter(1), reverse=True)
        # 最相似的N个用户
        # print("closest user:", closest_distance[:n])
        return closest_distance[:n]

```

```
    # 给用户推荐课程
    def recommend(self, username, n=1):
        recommend = set()
        nearest_user = self.nearest_user(username, n) # 获取最相近的n个用户
        for user_id, _ in nearest_user:
            for usercourse in UserCourse.objects.filter(user_id=user_id):
                if usercourse.course.id not in self.data[username].keys():
                    recommend.add(usercourse.course.id)
        return recommend

def recommend_by_user_id(user_id):
    # 通过用户协同算法来进行推荐
    current_user = User.objects.get(id=user_id)
    # 如果当前用户没有选修过课程，则按照收藏量降序返回
    if current_user.usercourse_set.count() == 0:
        courses = CourseInfo.objects.all().order_by('-collect_num')
        if courses.count() > 30:
            courses = courses[:30]
        return courses
    data = {}
    course_ids = []
    other_user_ids = set()
    # 把该用户选修过的课程变成向量字典：{'用户id': {'课程1id': 1, '课程2id': 1...}}
    for u_course in current_user.usercourse_set.all():
        # 遍历用户选修过的课程
        if not data:
            data[current_user.id] = {u_course.course.id: 1}  # 已选课程，设置值为1
        else:
            data[current_user.id][u_course.course.id] = 1
        course_ids.append(u_course.course)
        # 获取其他选修过该课程的用户id
        for usercourse in UserCourse.objects.filter(course=u_course.course):
            if usercourse.user.id != current_user.id:
                other_user_ids.add(usercourse.user.id)

    # 把选修过其中课程的用户选修过的课程变成向量字典：{'用户2id': {'课程1id': 0, '课程2id': 1...}}
    for other_user in User.objects.filter(pk__in=other_user_ids):
        other_user_id = other_user.id
        for i in range(len(course_ids)):
            course = course_ids[i]
            if UserCourse.objects.filter(user_id=other_user_id, course=course):
                is_select = 1
            else:
                is_select = 0
            if other_user_id not in data:
                data[other_user_id] = {course.id: is_select}  # 已选课程，设置值为1,未选课程设置为0
            else:
                data[other_user_id][course.id] = is_select

    user_cf = UserCf(data=data)
    recommend_ids = user_cf.recommend(current_user.id, 1)

    if not recommend_ids:
```

```
        # 如果没有找到相似用户则按照收藏量降序返回
        courses = CourseInfo.objects.all().order_by('-collect_num')
        if courses.count() > 30:
            courses = courses[:30]
        return courses

    return CourseInfo.objects.filter(id__in=recommend_ids).order_by('-select_num')

```

21、个人中心

```
@login_in
def personal(request):
    user = User.objects.get(id=request.session.get("user_id"))
    form = Edit(instance=user)
    return render(request, "personal.html", {"form": form})
```

22、我的选修

```
@login_in
def my_select(request):
    user = User.objects.get(id=request.session.get("user_id"))
    courses = user.usercourse_set.all()
    return render(request, "my_select.html", {"courses": courses, "last_url": "my_select"})
```

23、我的收藏

```
@login_in
def my_collect(request):
    collect_courses = CollectCourse.objects.filter(user_id=request.session.get("user_id"), is_delete=False)
    return render(request, "my_collect.html", {"collect_courses": collect_courses})
```

24、我的评分

```
@login_in
def my_rate(request):
    rate_courses = RateCourse.objects.filter(user_id=request.session.get("user_id"))
    return render(request, "my_rate.html", {"rate_courses": rate_courses})
```

25、删除我的评分

```
@login_in
def delete_rate(request, rate_id):
    rate = RateCourse.objects.filter(pk=rate_id)
    if not rate:
        return render(request, "404.html")
    rate = rate.first()
    rate.delete()
    rate_courses = RateCourse.objects.filter(user_id=request.session.get("user_id"))
    return render(request, "my_rate.html", {"rate_courses": rate_courses})
```

26、我的评论

```
@login_in
def my_comments(request):
    comment_courses = CommentCourse.objects.filter(user_id=request.session.get("user_id"), is_show=True)
    return render(request, "my_comment.html", {"comment_courses": comment_courses})
```

27、删除我的评论

```python
@login_in
def delete_comment(request, comment_id):
    # 删除评论
    comment = CommentCourse.objects.get(pk=comment_id)
    comment.is_show = False
    comment.save()
    comment_courses = CommentCourse.objects.filter(user_id=request.session.get("user_id"), is_show=True)
    return render(request, "my_comment.html", {"comment_courses": comment_courses})
```

七、后台管理

1、创建adminx.py文件

在子应用`course`下创建一个`adminx.py`文件:

里面代码为:

```python
# !/usr/bin/python
# -*- coding: utf-8 -*-
import xadmin
from django.utils.safestring import mark_safe
from xadmin import views
from django.conf import settings
from .models import *


# https://fontawesome.dashgame.com/  图标字体网站
# 基础设置
class BaseSetting(object):
    enable_themes = True  # 使用主题
    use_bootswatch = True


# 全局设置
class GlobalSettings(object):
    site_title = '课程管理系统'  # 标题
    site_footer = mark_safe('课程推荐系统')  # 页尾
    site_url = '/'
    menu_style = 'accordion'  # 设置左侧菜单  折叠样式


# 用户管理
class UserAdmin(object):
    search_fields = ['username']  # 检索字段
    list_display = ['id', 'username', 'password']  # 要显示的字段
    list_per_page = 30  # 默认每页显示多少条记录，默认是100条
    model_icon = 'fa fa-users'  # 左侧小图标


# 标签管理
class TagsAdmin(object):
    search_fields = ['name']  # 检索字段
```

```
    list_display = ['id', 'name']
    list_filter = ['name']
    ordering = ('id',)
    model_icon = 'fa fa-tags'  # 左侧小图标


# 课程管理
class CourseInfoAdmin(object):
    search_fields = ['name', 'teacher', 'about']  # 检索字段
    list_display = ['id', 'show_pic', 'name', 'teacher', 'add_time',
                    'tags', 'look_num', 'select_num', 'collect_num',
'show_avg_mark']  # 要显示的字段
    list_filter = ['add_time', 'tags']  # 分组过滤的字段
    ordering = ('id',)  # 设置默认排序字段，负号表示降序排序
    list_per_page = 30  # 默认每页显示多少条记录，默认是100条
    model_icon = 'fa fa-book'  # 左侧小图标
    list_editable = ['name', 'teacher']  # 可编辑字段
    style_fields = {'tags': 'm2m_transfer', 'prizes': 'm2m_transfer'}  # 控
制字段的显示样式
    filter_horizontal = ('tags', 'prizes')  # 水平选择编辑多对多字段

    def show_pic(self, obj):
        # 显示书籍封面
        if obj.pic.name:
            text = """
                <style type="text/css">

                    #div1 img{
                      cursor: pointer;
                      transition: all 0.6s;
                    }
                    #div1 img:hover{
                      transform: scale(2);
                    }
                </style>
                <div id="div1">
                    <img src="%s" style="width:50px;"/>
                </div>
                """ % (self.request.build_absolute_uri('/') + 'media/' +
obj.pic.name)

            return mark_safe(text)
        return ''

    def show_avg_mark(self, obj):
        return obj.avg_mark

    def save_models(self):
        flag = self.org_obj is None and 'create' or 'change'
        if flag == 'create':
            if self.new_obj.pic.name:
                self.new_obj.pic.name = f"{self.new_obj.title}.
{self.new_obj.pic.name.split('.')[1]}"
        if flag == 'change' and 'pic' in self.change_message():
            if self.org_obj.pic.name:
                self.org_obj.pic.name = f"{self.org_obj.title}.
{self.org_obj.pic.name.split('.')[1]}"

```

```
        super().save_models()

    show_pic.short_description = '封面'
    show_avg_mark.short_description = '评分'

# 选课表
class UserCourseAdmin(object):
    search_fields = ['course__name', 'user__name', 'mark']  # 检索字段
    list_display = ['user', 'course', 'enroll_time']  # 要显示的字段
    list_filter = ['enroll_time']  # 分组过滤的字段
    ordering = ('id',)  # 设置默认排序字段，负号表示降序排序
    list_per_page = 30  # 默认每页显示多少条记录，默认是100条
    list_editable = []  # 可编辑字段
    fk_fields = ('course', 'user')  # 设置显示外键字段


# 书籍评分管理
class RateAdmin(object):
    search_fields = ['course__name', 'user__name', 'mark']  # 检索字段
    list_display = ['course', 'user', 'mark', 'create_time']  # 要显示的字段
    list_filter = ['mark', 'create_time']  # 分组过滤的字段
    ordering = ('id',)  # 设置默认排序字段，负号表示降序排序
    list_per_page = 30  # 默认每页显示多少条记录，默认是100条
    list_editable = []  # 可编辑字段
    fk_fields = ('course', 'user')  # 设置显示外键字段


# 书籍收藏管理
class CollectcourseAdmin(object):
    search_fields = ['course__title', 'user__name']  # 检索字段
    list_display = ['course', 'user', 'is_delete', 'create_time']  # 要显示
的字段
    list_filter = ['is_delete', 'create_time']  # 分组过滤的字段
    ordering = ('id',)  # 设置默认排序字段，负号表示降序排序
    list_per_page = 30  # 默认每页显示多少条记录，默认是100条
    list_editable = []  # 可编辑字段
    fk_fields = ('course', 'user')  # 设置显示外键字段


# 书籍评论管理
class CommentAdmin(object):
    search_fields = ['course__title', 'user__name']  # 检索字段
    list_display = ['user', 'course', 'show_content', 'like_num',
'is_show', 'create_time']  # 要显示的字段
    list_filter = ['course', 'is_show', 'create_time']  # 分组过滤的字段
    ordering = ('id',)  # 设置默认排序字段，负号表示降序排序
    list_per_page = 30  # 默认每页显示多少条记录，默认是100条
    list_editable = []  # 可编辑字段
    fk_fields = ('course', 'user')  # 设置显示外键字段

    def show_content(self, obj):
        # 显示评论内容
        if not obj.content:
            return mark_safe('')
        if len(obj.content) < 20:
            return mark_safe(obj.content)
        short_id = f'{obj._meta.db_table}_short_text_{obj.id}'
        short_text = obj.content[:len(obj.content) // 4] + '......'
```

```
        detail_id = f'{obj._meta.db_table}_detail_text_{obj.id}'
        detail_text = obj.content

        text = """<style type="text/css">
                        #%s,%s {padding:10px;border:1px solid green;}
                    </style>
                      <script type="text/javascript">

                      function openShutManager(oSourceObj,oTargetObj,shutAble,oOpenTip,oShutTip,oShortObj){
                        var sourceObj = typeof oSourceObj == "string" ? document.getElementById(oSourceObj) : oSourceObj;
                        var targetObj = typeof oTargetObj == "string" ? document.getElementById(oTargetObj) : oTargetObj;
                        var shortObj = typeof oShortObj == "string" ? document.getElementById(oShortObj) : oShortObj;
                        var openTip = oOpenTip || "";
                        var shutTip = oShutTip || "";
                        if(targetObj.style.display!="none"){
                           if(shutAble) return;
                           targetObj.style.display="none";
                           shortObj.style.display="block";
                           if(openTip  &&  shutTip){
                            sourceObj.innerHTML = shutTip;
                           }
                        } else {
                           targetObj.style.display="block";
                           shortObj.style.display="none";
                           if(openTip  &&  shutTip){
                            sourceObj.innerHTML = openTip;
                           }
                        }
                        }
                    </script>
                    <p id="%s">%s</p>
                    <p><a href="###" onclick="openShutManager(this,'%s',false,'点击关闭','点击展开','%s')">点击展开</a></p>

                    <p id="%s" style="display:none">
                       %s
                    </p>
                    """ % (short_id, detail_id, short_id, short_text, detail_id, short_id, detail_id, detail_text)
        return mark_safe(text)

    show_content.short_description = '评论内容'


xadmin.site.register(views.CommAdminView, GlobalSettings)
xadmin.site.register(views.BaseAdminView, BaseSetting)
xadmin.site.register(User, UserAdmin)
xadmin.site.register(Tags, TagsAdmin)
xadmin.site.register(CourseInfo, CourseInfoAdmin)
xadmin.site.register(UserCourse, UserCourseAdmin)
xadmin.site.register(RateCourse, RateAdmin)
xadmin.site.register(CollectCourse, CollectcourseAdmin)
```

```
xadmin.site.register(CommentCourse, CommentAdmin)

```

2、修改course\apps.py

代码改为如下：

```
from django.apps import AppConfig


class CourseConfig(AppConfig):
    name = 'course'
    verbose_name = "课程推荐系统"
```

3、修改`course\__init__.py`

代码改为：

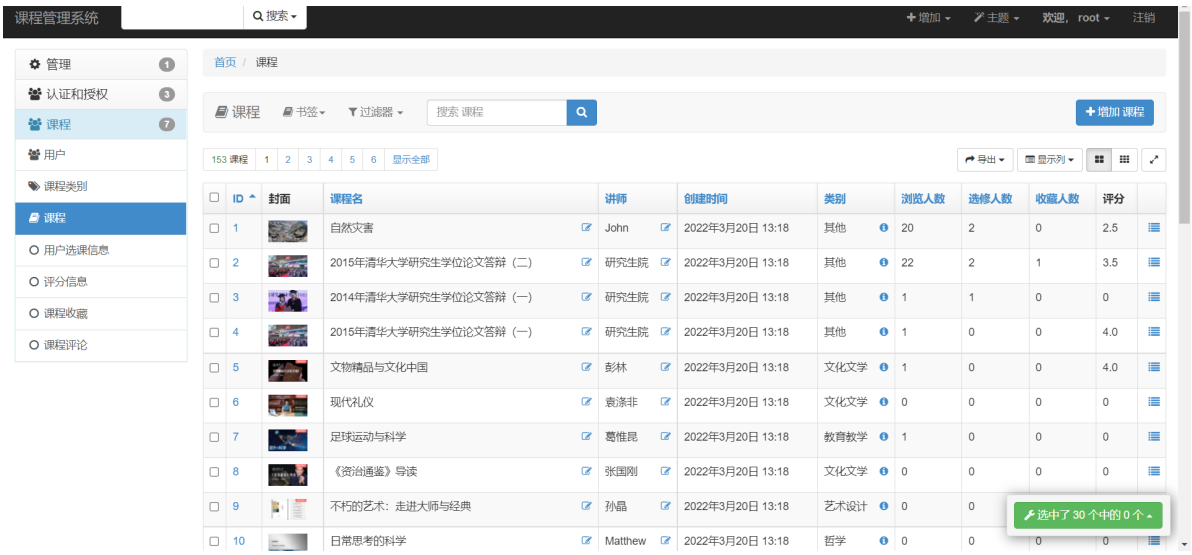
```
default_app_config = 'course.apps.CourseConfig'
```

4、浏览器登录

浏览器访问`http://127.0.0.1:8000/xadmin/`

输入账号：`root`

输入密码：`course-root`



八、部署到服务器

1、前期工作

部署到`ubuntu`服务器，把项目放到`home`目录下，创建数据库`course_manager`。

在服务器`/home/venv/`目录下创建一个虚拟环境`course_manager`：

```
cd ..
cd home/venv/
python3 -m venv course_manager
```

激活虚拟环境：

```
source course_manager/bin/activate
```

更新pip:

```
pip install --upgrade pip
```

退回项目根目录：

```
cd ..
cd course_manager
```

批量安装第三方库：

```
pip install -r requirements.txt  -i https://pypi.tuna.tsinghua.edu.cn/simple
```

数据迁移：

```
python manage.py migrate
```

创建缓存表：

```
python manage.py createcachetable
```

创建后台管理员

```
python manage.py createsuperuser
```

```
设置账号为 root
邮箱为 1@qq.com
密码为 book-root
```

数据库图形化工具执行`course.sql`

2、配置uwsgi.ini

`uwsgi.ini`内容如下：

```
[uwsgi]
# 使用nginx连接时 使用
socket=0.0.0.0:8102
# 直接作为web服务器使用
#http=127.0.0.1:8102
# 配置工程目录
chdir=/home/course_manager
# 配置项目的wsgi目录。相对于工程目录
wsgi-file=course_manager/wsgi.py
virtualenv =/home/venv/course_manager
#配置进程，线程信息
listen=1024
processes=2
threads=4
enable-threads=True
master=True
pidfile=uwsgi.pid
daemonize=uwsgi.log
#django项目修改完文件后自动重启
py-autoreload=1

```

把`uwsgi.ini`放到根目录下。

导入`uwsgi`：

```
pip install uwsgi
```

启动uwsgi:

```
uwsgi --ini uwsgi.ini
```

查看是否启动成功：

```
netstat -lnp|grep uwsgi
```

若出现类似：

```
tcp        0      0 0.0.0.0:8102            0.0.0.0:*               LISTEN
   927/uwsgi
```

则说明启动成功

2、配置nginx

创建一个`course_manager_nginx`文件，内容如下：

```
    #设定虚拟主机配置
        server {
            #侦听80端口

```

```
            listen 80;
            #listen 443 ssl;
            #定义使用 www.nginx.cn访问
            #ssl on;
            server_name  xxx.xxx.com;
            #server_name  xxx.xxx.xxx.30;
            #定义服务器的默认网站根目录位置
            root /home/course_manager;
            #ssl_session_timeout 5m;
            #ssl_certificate   /etc/nginx/cert/xxx.pem;
            #ssl_certificate_key  /etc/nginx/cert/xxx.key;
            #ssl_ciphers ECDHE-RSA-AES128-GCM-
SHA256:ECDHE:ECDH:AES:HIGH:!NULL:!aNULL:!MD5:!ADH:!RC4;
            #ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
            #ssl_prefer_server_ciphers on;
            #设定本虚拟主机的访问日志
            #access_log  logs/nginx.access.log  main;

            #默认请求
            location / {
                #倒入了uwsgi的配置
                include uwsgi_params;
                client_max_body_size    50m;
                #连接uwsgi的超时时间
               # uwsgi_connect_timeout 30;
            #设定了uwsig服务器位置
                uwsgi_pass 127.0.0.1:8102;
            }

            location /static{
            alias /home/course_manager/static;
            }
        location /media {
        alias /home/course_manager/media;
        }
    }

```

文件放到`/etc/nginx/sites-available`下面。

然后通过以下命令映射到`/etc/nginx/sites-enabled`

```
ln -s /etc/nginx/sites-available/course_manager_nginx /etc/nginx/sites-
enabled/course_manager_nginx
```

`nginx`重启:

```
nginx -s reload
```

在浏览器中访问网站就可以看到书籍了。