Start

```python
import sys
import os
sys.path.append(os.path.dirname(__file__))
from core import src
if __name__ == '__main__':
    src.run()
```

core(src, admin, student, teacher)

core(src)

```python
from core import admin, student, teacher
func_dic = {
    '1': admin.admin_view,
    '2': student.student_view,
    '3': teacher.teacher_view,
}
def run():
    while True:
        print('''
        1.管理员视图

        2.学生视图

        3.老师视图

        q.退出
        ''')
        choice = input('请选择视图:').strip()
        if choice == 'q':
            break
        if choice not in func_dic:
            print('选择有误!')
            continue
        func_dic.get(choice)()
```

core(admin.py)

```python
from interface import admin_interface
from interface import common_interface
from lib import common
admin_info = {
    'user': None
}
def register():
    while True:
        username = input('请输入用户名:').strip()
        password = input('请输入密码:').strip()

        re_password = input('请确认密码:').strip()

        if password == re_password:
            flag, msg = admin_interface.register_interface(username,
password)
            if flag:
                print(msg)
                break
            else:
                print(msg)
        else:

            print('两次密码不一致!')

def login():
    while True:

        username = input('请输入用户名').strip()

        password = input('请输入密码').strip()

        flag, msg = common_interface.login_interface(username, password,
user_type='admin')
        if flag:
            print(msg)
            admin_info['user'] = username
            break
        else:
            print(msg)

@common.login_auth('admin')
def create_school():
    while True:

        # 学校名\学校地址

        school_name = input('请输入学校名:').strip()

        school_addr = input('请输入学校地址:').strip()

        flag, msg = admin_interface.create_school_interface(
            admin_info.get('user'), school_name, school_addr)
        if flag:
            print(msg)
            break
        else:
            print(msg)

@common.login_auth('admin')
```

```python
def create_teacher():
    while True:
        teacher_name = input('请输入老师的用户名:').strip()
        flag, msg = admin_interface.create_teacher_interface(
            admin_info.get('user'), teacher_name)
        if flag:
            print(msg)
            break
        else:
            print(msg)

@common.login_auth('admin')
def create_course():
    while True:
        # 1.获取所有的学校
        # [s1, s2...]   or   None
        school_list = common_interface.get_school_interface()
        if not school_list:
            print('没有学校,请去创建!')
            break
        for index, school in enumerate(school_list):
            print(index, school)
        # 2.选择学校
        choice = input('请选择学校编号:').strip()
        if not choice.isdigit():
            print('请输入数字')
            continue
        choice = int(choice)
        if choice not in range(len(school_list)):
            print('输入有误!')
            continue
        # 3.添加课程给学校
        school_name = school_list[choice]
        course_name = input('请输入课程名称: ').strip()
        flag, msg = admin_interface.create_course_interface(
            admin_info.get('user'), school_name, course_name
        )
        if flag:
            print(msg)
            break
        else:
            print(msg)


func_dic = {
    '1': register,
    '2': login,
    '3': create_school,
    '4': create_teacher,
    '5': create_course,
}

def admin_view():
    while True:
        print('''
        1.注册
        2.登录
        3.创建学校
        4.创建老师
        5.创建课程
        q.退出
        ''')

        choice = input('请选择管理员功能:').strip()

        if choice == 'q':
            break

        if choice not in func_dic:
            print('选择有误!')
            continue

        func_dic.get(choice)()
```

core(student.py)

```
from interface import student_interface
from interface import common_interface
from lib import common
student_info = {
    'user': None
}

def register():
```

```python
    while True:
        username = input('请输入用户名:').strip()
        password = input('请输入密码:').strip()
        re_password = input('请确认密码:').strip()
        if password == re_password:
            flag, msg = student_interface.register_interface(username, 
password)
            if flag:
                print(msg)
                break
            else:
                print(msg)
        else:
            print('两次密码不一致!')


def login():
    while True:
        username = input('请输入用户名').strip()
        password = input('请输入密码').strip()
        flag, msg = common_interface.login_interface(username, password, 
user_type='student')
        if flag:
            print(msg)
            student_info['user'] = username
            break
        else:
            print(msg)

@common.login_auth('student')
def choose_school():
    while True:
        school_list = common_interface.get_school_interface()
        for index, school in enumerate(school_list):
            print(index, school)
        choice = input('请输入选择的学校编号:').strip()
        # 如果不是数字

        if not choice.isdigit():
            print('必须是数字!')
            continue
        choice = int(choice)
        if choice not in range(len(school_list)):
            print('必须输入正确学校编号!')

            continue
        school_name = school_list[choice]
        flag, msg = student_interface.choose_school_interface(
            student_info.get('user'), school_name)
        if flag:
            print(msg)
            break
        else:
            print(msg)

@common.login_auth('student')
def choose_course():
    while True:
        # 1.获取学生下学校所有的课程

        flag, course_list_or_msg = student_interface.get_course_interface(
            student_info.get('user'))

        if not flag:
            print(course_list_or_msg)
            break

        if not course_list_or_msg:
            print('没有课程')

            break

        for index, course in enumerate(course_list_or_msg):
            print(index, course)

        choice = input('请选择课程编号:').strip()

        if not choice.isdigit():
            print('请输入数字!')

            continue

        choice = int(choice)

        if choice not in range(len(course_list_or_msg)):
            print('请选择正确编号')

            continue
        course_name = course_list_or_msg[choice]
        flag, msg = student_interface.choose_course_interface(
            student_info.get('user'), course_name)
        if flag:
                print(msg)
                break
            else:
                print(msg)
        pass

@common.login_auth('student')
def check_score():
    score_dic = 
student_interface.check_score_interface(student_info.get('user'))
    print(score_dic)
func_dic = {
    '1': register,
    '2': login,
    '3': choose_school,
    '4': choose_course,
    '5': check_score,
}

def student_view():
    while True:
        print('''

        1.注册

        2.登录

        3.选择学校

        4.选择课程

        5.查看成绩

        q.退出
        ''')

        choice = input('请选择学生功能:').strip()

        if choice == 'q':
            break
        if choice not in func_dic:

            print('选择有误!')

            continue
        func_dic.get(choice)()
```

core(teacher.py)

```python
from interface import teacher_interface
from interface import common_interface
from lib import common
teacher_info = {
    'user': None
}
def login():
    while True:
        username = input('请输入用户名:').strip()
        password = input('请输入密码:').strip()
        flag, msg = common_interface.login_interface(username, password, 
user_type='teacher')
        if flag:
            teacher_info['user'] = username
            print(msg)
            break
        else:
            print(msg)

# 查看教授课程
@common.login_auth('teacher')
def check_course():
    flag, course_list_or_msg = teacher_interface.check_course_interface(
        teacher_info.get('user'))
    if flag:
        print(course_list_or_msg)
    else:
        print(course_list_or_msg)
# 选择教授课程
@common.login_auth('teacher')
def choose_course():
    while True:
        # 1.查看所有课程
        course_list = common_interface.get_courses_interface()
        if not course_list:
            print('没有课程')
            break
        # 2.打印所有课程,并选择
        for index, course in enumerate(course_list):
            print(index, course)
        choice = input('请输入课程编号:').strip()
        if not choice.isdigit():
            print('必须是数字')
            continue
        choice = int(choice)
        if choice not in range(len(course_list)):
            print('请选择正确编号!')
            continue
        course_name = course_list[choice]
        flag, msg = teacher_interface.choose_course_interface(
            teacher_info.get('user'), course_name)
        if flag:
            print(msg)
            break
        else:
            print(msg)

# 查看课程下学生
@common.login_auth('teacher')
def check_student():
    while True:
        # 1.获取老师下所有课程
        flag, course_list_or_msg = teacher_interface.check_course_interface(
            teacher_info.get('user'))
        if not flag:
            print('没有课程')
            break
        for index, course in enumerate(course_list_or_msg):
            print(index, course)
        choice = input('请输入课程编号:').strip()
        if not choice.isdigit():
            continue
        choice = int(choice)
        if choice not in range(len(course_list_or_msg)):
            continue
        course_name = course_list_or_msg[choice]
        # 调用查看课程下学生接口
        flag, student_list_or_msg =
teacher_interface.check_student_interface(
            teacher_info.get('user'), course_name)
        if flag:
            print(student_list_or_msg)
            break
        else:
            print(student_list_or_msg)
            break

# 修改学生分数
```

```python
@common.login_auth('teacher')
def change_score():
    while True:
        # 1.获取当前老师下所有的课程
        flag, course_list_or_msg = teacher_interface.check_course_interface(
            teacher_info.get('user'))
        if not flag:
            print('老师下没有课程')
            break
        for index, course in enumerate(course_list_or_msg):
            print(index, course)
        choice = input('请选择课程编号:').strip()
        if not choice.isdigit():
            continue
        choice = int(choice)
        if choice not in range(len(course_list_or_msg)):
            continue
        course_name = course_list_or_msg[choice]
        flag, student_list_or_msg =
teacher_interface.check_student_interface(
            teacher_info.get('user'), course_name)
        # 若有学生,则循环打印学生列表,让老师选择学生
        if not flag:
            print(student_list_or_msg)
            break
        for index, student in enumerate(student_list_or_msg):
            print(index, student)
        choice2 = input('请选择学生编号:').strip()
        if not choice2.isdigit():
            continue
        choice2 = int(choice2)
        if choice2 not in range(len(student_list_or_msg)):
            continue
        student_name = student_list_or_msg[choice2]
        # 输入修改学生的成绩
        score = input('请输入修改的成绩:').strip()
        flag, msg = teacher_interface.change_score_interface(
            teacher_info.get('user'), course_name, student_name, score
        )
        if flag:
            print(msg)
            break
func_dic = {
    '1': login,
    '2': check_course,
    '3': choose_course,
    '4': check_student,
    '5': change_score,
}

def teacher_view():
    while True:
        print('''
        1.登录
        2.查看教授课程
        3.选择教授课程
        4.查看课程学生
        5.修改学生成绩
        q.退出
        ''')
        choice = input('请选择老师功能:').strip()
        if choice == 'q':
            break
        if choice not in func_dic:
            print('选择有误!')
            continue
        func_dic.get(choice)()
```

db(db_handler.py , models.py)

db(db_handler.py)

```
from conf import settings
import os
import pickle
def db_select(cls, username):
    # 1.获取当前用户文件夹
    class_name = cls.__name__
    dir_path = os.path.join(settings.DB_PATH, class_name)
    # 判断文件夹是否存在
    if os.path.isdir(dir_path):
        user_path = os.path.join(dir_path, username)
```

```python
        # 判断文件是否存在
        if os.path.exists(user_path):
            # 把对象从 pickle 文件中读出，若不存在，则默认返回 None
            with open(user_path, 'rb') as f:
                obj = pickle.load(f)
                return obj

# 保存数据
def db_save(obj):
    # Admin
    class_name = obj.__class__.__name__
    # 获取保存文件目录
    dir_path = os.path.join(settings.DB_PATH, class_name)
    # 判断文件夹是否存在,不存在则创建
    if not os.path.isdir(dir_path):
        os.mkdir(dir_path)
    # 拼接 pickle 文件路径
    user_path = os.path.join(dir_path, obj.name)
    with open(user_path, 'wb') as f:
        pickle.dump(obj, f)
        f.flush()
```

db(models.py)

```python
from db import db_handler
class Base:
    # 对象的保存数据方法
    def save(self):
        db_handler.db_save(self)
    # 对象的查询方法
    @classmethod
    def select(cls, username):
        obj = db_handler.db_select(cls, username)
        return obj
# 管理员类
class Admin(Base):
    def __init__(self, name, pwd):
        self.name = name
        self.pwd = pwd
        self.save()
    # 管理员创建学校方法
    def create_school(self, school_name, school_addr):
        # 实例化学校类,创建学校
        School(school_name, school_addr)
    # 管理员创建老师方法
    def create_teacher(self, teacher_name, teacher_pwd):
        # 实例化 Teacher 保存老师对象
        Teacher(teacher_name, teacher_pwd)
    # 管理员创建课程方法
    def create_course(self, school_name, course_name):
        # 1.给学校添加课程

        # 获取学校对象的课程列表
        school_obj = School.select(school_name)
        # 实例化课程类创建课程
        Course(course_name)
        # 把课程绑定给学校
        school_obj.add_course(course_name)


# 学生类
class Student(Base):
    def __init__(self, student_name, student_pwd):
        self.name = student_name
        self.pwd = student_pwd
        self.school = None
        self.student_course_list = []
        # 学生的所有分数
        self.score = {}  # score[course_name] = score
        self.save()
    # 学生选择学校
    def choose_school(self, school_name):
        self.school = school_name
        self.save()
    # 学生选择课程
    def choose_course(self, course_name):
        self.student_course_list.append(course_name)
        # 1.学生选择课程并初始化该课程分数
        self.score[course_name] = 0
        self.save()
        # 2.让课程也选择学生
        course_obj = Course.select(course_name)
        course_obj.add_student(self.name)

    # 学生查看成绩
    def check_score(self):
        return self.score

# 课程类
class Course(Base):
    def __init__(self, course_name):
        self.name = course_name
        self.student_list = []
        self.save()
    def add_student(self, student_name):
        self.student_list.append(student_name)
        self.save()

# 老师类
class Teacher(Base):
    def __init__(self, teacher_name, teacher_pwd):
        self.name = teacher_name
        self.pwd = teacher_pwd

        # 一个老师可以有多个课程
        self.teacher_course_list = []
        self.save()

    # 老师查看教授课程方法
    def check_course(self):
        return self.teacher_course_list

    # 老师选择教授课程方法
    def choose_course(self, course_name):
        self.teacher_course_list.append(course_name)
        self.save()

    # 老师查看课程下学生方法
    def check_student(self, course_name):

        # 1.获取课程对象
        course_obj = Course.select(course_name)
        return course_obj.student_list

    # 老师修改成绩方法
    def change_score(self, course_name, student_name, score):
        student_obj = Student.select(student_name)
        student_obj.score[course_name] = score
        student_obj.save()
```

```python
# 学校类
class School(Base):
    def __init__(self, school_name, school_addr):
        self.name = school_name
        self.addr = school_addr
        # 一所学校可以有多个课程
        self.school_course_list = []
        self.save()
    def add_course(self, course_name):
        self.school_course_list.append(course_name)
        self.save()

```

conf(settings.py)

```python
import os
BASE_PATH = os.path.dirname( os.path.dirname(__file__))
DB_PATH = os.path.join(BASE_PATH, 'db')
```

interface(admin_interface , student_interface, teacher_interface, common_interface)

interface(admin_interface.py)

```python
from db import models
def register_interface(username, password):
    # 1.判断用户是否存在
    # 不合理：不要使用
    # obj = models.Admin(username, password)
    # obj.select(username)
    # 合理：推荐使用
    admin_obj = models.Admin.select(username)
    if admin_obj:
        return False, '用户已存在!'
    # 若存在则返回给用户，用户已存在
    # 若不存在去保存用户数据
    # 方式一：
    # admin_obj = models.Admin(username, password)
    # admin_obj.save()
    # 方式二：
    # 保存用户数据
    models.Admin(username, password)
    return True, f'{username}---注册成功'

# def login_interface(username, password):
#     admin_obj = models.Admin.select(username)
#     if admin_obj:
#         if admin_obj.pwd == password:
#             return True, f'{username}---登录成功'
#
#         else:
#             return False, '密码错误'
#     else:
#         return False, '用户不存在!'


# 创建学校接口
def create_school_interface(admin_name, school_name, school_addr):
    # 1.判断学校是否存在
    school_obj = models.School.select(school_name)
    if school_obj:
        return False, '学校已存在!'
    # 2.若学校不存在则保存学校
    # 获取管理员对象,让管理员来创建学校
    admin_obj = models.Admin.select(admin_name)
    admin_obj.create_school(
        school_name, school_addr)
    return True, f'{school_name}--学校创建成功!'

# 创建老师接口
def create_teacher_interface(admin_name, teacher_name, teacher_pwd='123'):
    teacher_obj = models.Teacher.select(teacher_name)
    if teacher_obj:
        return False, '老师已存在!'
    # 通过管理员对象, 创建老师
    admin_obj = models.Admin.select(admin_name)
    admin_obj.create_teacher(teacher_name, teacher_pwd)
    return True, f'{teacher_name}---创建成功!'

# 创建课程接口
def create_course_interface(admin_name, school_name, course_name):
    # 1.获取学校对象中的课程列表，判断当前课程是否存在列表中
    school_obj = models.School.select(school_name)
    if course_name in school_obj.school_course_list:
        return False, '该学校已存在此课程!'
    # 2.由管理员创建课程
    admin_obj = models.Admin.select(admin_name)
    admin_obj.create_course(school_name, course_name)
    return True, f'{course_name}---课程创建成功!'
```

interface(student_interface)

```
from db import models
def register_interface(username, password):
    # 1.判断学生是否存在
    student_obj = models.Student.select(username)
    if student_obj:
        return False, '学生已存在'
    models.Student(username, password)
    return True, f'{username}---学生创建成功!'
# def login_interface(username, password):
#     # 1.获取学生对象,判断学生是否存在
#     student_obj = models.Student.select(username)
#     if student_obj:
#         # 判断密码是否一致
#         if student_obj.pwd == password:
#             return True, '登录成功!'
#         else:
#             return  False, '密码错误!'
#     else:
#         return False , '用户不存在!'
# 学生选择学校接口
def choose_school_interface(student_name, school_name):
    # 1.判断学生是否拥有学校
    student_obj = models.Student.select(student_name)
    if student_obj.school:
        return False, '学生已选择学校'
```

```python
    # 2.让学生对象选择学校
    student_obj.choose_school(school_name)
    return True, '选择学校成功!'

# 获取学校下所有课程接口
def get_course_interface(student_name):
    student_obj = models.Student.select(student_name)
    if student_obj.school:
        school_name = student_obj.school
        school_obj = models.School.select(school_name)
        return True, school_obj.school_course_list
    else:
        return False, '请先选择学校!'


# 学生选择课程接口
def choose_course_interface(student_name, course_name):
    # 1.判断该课程是否存在学生课程列表中
    student_obj = models.Student.select(student_name)
    if course_name in student_obj.student_course_list:
        return False, '该课程已经选择过了!'
    student_obj.choose_course(course_name)
    return True, f'{course_name}---课程选择成功!'

# 学生查看成绩接口
def check_score_interface(student_name):
    student_obj = models.Student.select(student_name)
    score_dic = student_obj.check_score()
    return score_dic
```

interface(teacher_interface)

```python
from db import models
# def login_interface(username, password):
#     # 1.判断用户名是否存在
#     teacher_obj = models.Teacher.select(username)
#     if teacher_obj:
#         if teacher_obj.pwd == password:
#             return True, '登录成功!'
#         else:
#             return False, '密码错误'
#     else:
```

```python
#         return  False, '用户不存在!'

# 老师查看教授课程接口
def check_course_interface(teacher_name):
    teacher_obj = models.Teacher.select(teacher_name)
    # 老师去获取教授课程数据
    course_list = teacher_obj.check_course()
    if course_list:
        return True, course_list
    return False, '没有课程'

# 老师选择教授课程接口
def choose_course_interface(teacher_name, course_name):
    # 1.判断课程是否在老师教授课程列表中
    teacher_obj = models.Teacher.select(teacher_name)
    # 若存在,则返回课程已存在
    if course_name in teacher_obj.teacher_course_list:
        return False, '课程已存在'
    # 若不存在,则添加
    teacher_obj.choose_course(course_name)
    return True, f'{course_name}---课程添加成功!'

# 老师查看课程下学生接口
def check_student_interface(teacher_name, course_name):
    # 1.获取老师对象
    teacher_obj = models.Teacher.select(teacher_name)
    # 2.让老师对象,去查看课程下学生
    student_list = teacher_obj.check_student(course_name)
    if student_list:
        return True, student_list
    return False, '课程下没有学生'


# 老师修改学生成绩
def change_score_interface(teacher_name, course_name, student_name, score):
    # 1.获取老师对象
    teacher_obj = models.Teacher.select(teacher_name)
    # 2.让老师去修改成绩
    teacher_obj.change_score(course_name, student_name, score)
    return True, '修改成绩成功!'
```

interface(common_interface)

```
from conf import settings
from db import models
import os
# 获取所有学校,返回一个列表
def get_school_interface():
    school_path = os.path.join(
        settings.DB_PATH, 'School')
    if os.path.exists(school_path):
        school_list = os.listdir(school_path)
        return school_list
# 获取所有课程接口
def get_courses_interface():
    # 1.拼接课程文件路径
    course_path = os.path.join(
        settings.DB_PATH, 'Course'
    )
    if os.path.exists(course_path):
        return os.listdir(course_path)

def login_interface(username, password, user_type):  # user_type --> admin
    if user_type == 'admin':
        # 1.判断用户名是否存在
        obj = models.Admin.select(username)
    elif user_type == 'student':
        obj = models.Student.select(username)
    elif user_type == 'teacher':
        obj = models.Teacher.select(username)
    else:
        return False, '没有权限'
    # 判断用户是否存在 (admin_obj student_obj teacher_obj)
    if not obj:
        return False, '用户不存在!'
    if obj.pwd == password:
        return True, f'{username}---登录成功!'
```

```python
    else:
        return False, '密码错误'
```

lib(common.py)

```python
def login_auth(role):
    def auth(func):
        from core import admin, student, teacher
        def inner(*args, **kwargs):
            if role == 'admin':
                if admin.admin_info.get('user'):
                    res = func(*args, **kwargs)
                    return res
                else:
                    admin.login()
            elif role == 'student':
                if student.student_info.get('user'):
                    res = func(*args, **kwargs)
                    return res
                else:
                    student.login()
            elif role == 'teacher':
                if teacher.teacher_info.get('user'):
                    res = func(*args, **kwargs)
                    return res
                else:
                    teacher.login()
            else:
                print('权限不足!')
        return inner
    return auth
```