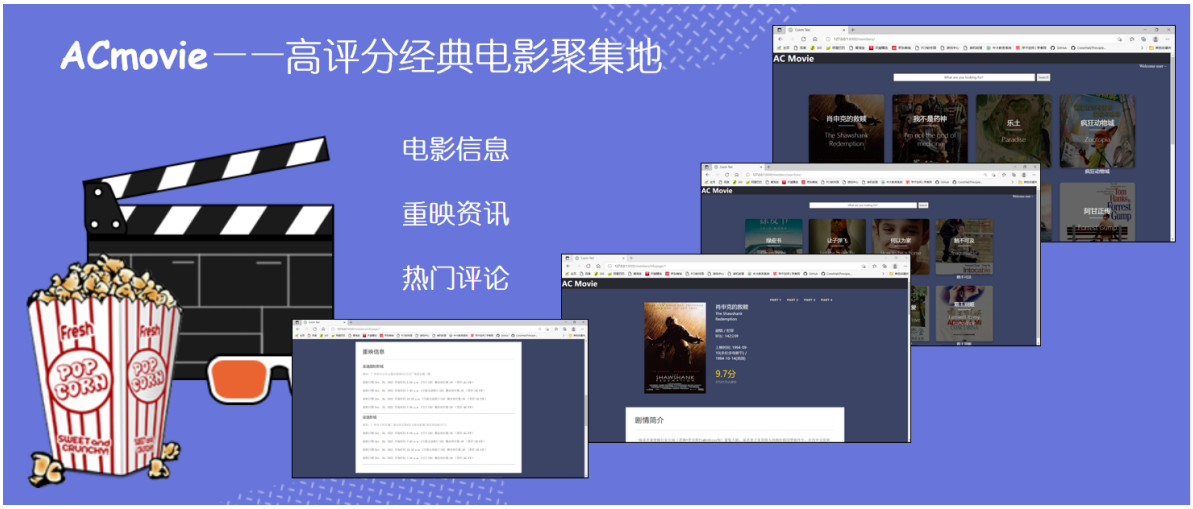
数据库期末大作业报告

姓名	学号	负责的工作
崔璨明（组长）	20337025	产品设计、数据库设计、前端网站设计、编写实验报告
付恒宇	20337029	爬取数据、参与产品设计

1 应用需求介绍

我们小组实现的是一个经典电影数据库应用系统，该应用系统基于数据库管理系统openGauss实现，包含8个以上的数据库表，并在该数据库的基础上搭建了Web界面，实现了一个经典电影信息网站。（展示视频和所有代码见附件）

我们小组在讨论选题时考虑了许多因素，如网站的实用性和创新性、数据库的利用率等等，并查找了大量的资料和调查，最终选择设计一个电影网站。但该电影网站和市面上的猫眼电影等平台不同，我们的网站专注于收集高评分的经典电影，并提供给用户查询影片信息、观影方式、与观影爱好者们一同评论电影等服务。总的来说，**我们设计的是一个欣赏经典电影的平台——AC Movie** 。



应用背景及设计目标：
我们调查到由于疫情的影响，许多电影院都采取了重映经典电影的方式来缓解经济压力，这对电影爱好者们来说无疑是好消息。但许多爱好者并不知道哪些影院有重映电影、该电影的具体信息、用户的评论和自己关注的电影是否重映等信息，而我们的网站便针对这一用户需求而设计。

我们旨在设计一个经典电影的网站，可以让人们了解到经典电影的背景、剧情、主演、导演等信息，并提供观看电影的方式。这对于电影爱好者来说是非常有意义的，因为他们可以在网站上轻松找到自己感兴趣的电影，并了解到更多的信息。

此外，经典电影网站还可以为广大电影爱好者提供交流的平台，让他们可以分享自己的看法和心得，与其他电影爱好者一起探讨电影的魅力所在。

开发平台
由于需要设计和实现一个数据库应用系统，并能过较为方便地演示，于是我们采用了搭建网站的方法。为此学习了html和css的语法和使用，以及Django的基础用法。Django是高水准的Python编程语言驱动的一个开源模型，视图，控制器风格的Web应用程序框架，它起源于开源社区。使用这种架构，我们可以方便、快捷地创建高品质、易维护、数据库驱动的应用程序。这也正是我们采用这种架构进行连接网站与后端数据库的主要原因。开发过程中主要使用的工具如下：

- Django 4.1.4
- openguass
- data studio
- python 3.10.6

---

2 应用系统设计

2.1 功能设计

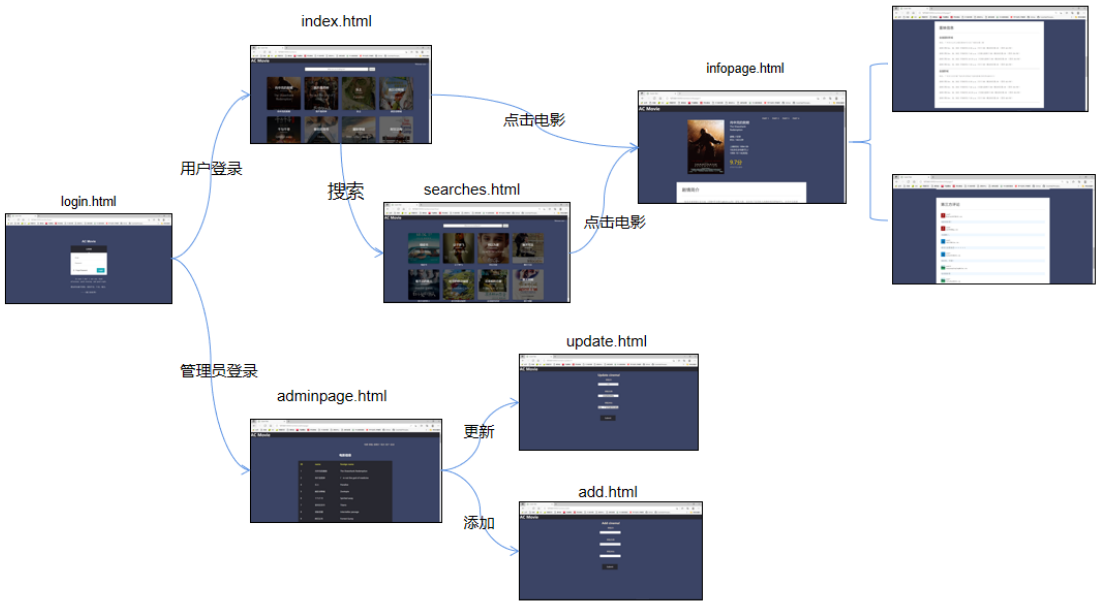
我们首先对该应用系统的进行了功能设计，经过讨论和收集相似类型的网站的资料，我们为我们的数据库应用系统设计了以下功能：

- **用户管理**：网站的数据库会储存每个用户的账号和密码，只有用户输入了正确的用户名（邮箱）和对应的密码后才能进入网站的主页，若输入的账号和密码不一致则会返回登录页面。且管理员无法查看和修改用户的账号和密码，这具有保护用户隐私的现实意义。
- **身份访问**：我们为网站的用户设计了两种身份：普通用户和管理员。两种身份的用户看到的网站的界面是不一样的，用户看到的简洁、提供信息的界面，可以正常使用网站提供的服务。而管理员看到的是便于管理和修改的数据表形式的页面，管理员可以通过相关的接口来修改数据库中的部分数据。
- **电影信息**：用户可以在网站中看到经典电影的信息，包括电影中英文名、上映日期、上映国家、演职员表、内容简介、用户评分及评分人数、重映的影院的地址、重映场次、每个场次的票价、对应影院的放映厅、放映厅的剩余座位数、来自第三方的用户评论等信息。
- **搜索功能**：我们的网站内置有搜索引擎，可以输入电影名字来查询相关的电影，在搜索框内输入信息后，会筛选出符合信息的电影。
- **增删数据**：当使用管理员身份登录网站时，便会跳转到数据修改页面，管理员可以增加数据库中的数据。例如当有新的影院重映某一部电影时，管理员可以通过该接口增加电影场次、上映影院等信息。当然，当场次结束时，管理员也可以删除相关的信息。我们采取的是立刻提交的方式，因此修改后的内容便会立即在用户界面和后端数据库中更新。
- **更新数据**：当使用管理员身份登录网站时，管理员可以更新数据库中的数据。例如当影院更名或放映厅更名时，管理员可以更新其记录。
- **随机语录**：为舒畅用户心情，用户每次打开登录界面时，我们都会从所记录的电影名言中抽取一条进行展示。

2.2 前端网站开发

接着进行Web界面的设计，在设计网页时，我们秉持简约的设计风格，以藏青色、白色和黑色为主要色调，考虑到电影的海报是吸引用户和美化网页的重要一环，因此我们在电影的海报设计上增加了鼠标聚焦的动画，使页面更加生动。具体设计的代码在.css文件中可以看到，内容较多且非重要内容，在此就不做过多介绍。

我们的数据库应用系统在前端共有六类页面：登录界面`login.html`、网站主页`index.html`，展示电影信息的页面`infopage.html`、展示搜索结果的页面`searches.html`、管理员页面`adminpage.html`、添加记录的页面`add.html`、更新记录的页面`update.html`以及错误界面`faultpage.html`，页面跳转逻辑如下图所示：



网页设计这部分代码较多，在此就不进行详细的介绍，所有前端代码可以在附件或该仓库中找到：https://github.com/91Mrcui/database_final_web.git

2.3 前后端的交互

我们采用了Django来进行前端网站与后端openguass数据库的连接。Django为我们提供了很方便的视图函数，视图函数是处理Web请求的函数。同时，我们还需要编写URLconf，URLconf是一组映射关系，用于将URL映射到视图函数。我用我本地的电脑作为数据库的服务器，在Django的`views.y`文件里面进行数据库的连接，只有当连接上本地的数据库后，网站才可以正常使用。Django为用户提供了很方便的接口，我们只需要在`views.py`用python代码编写对数据库操作的函数，然后将操作的结果通过特定的函数以字典的方式传到html中即可。`views.py`中函数较多且都写有注释，在此就不详细介绍，只以部分代码为例进行关键原理的介绍。

如下是查询指定表的函数，该函数接受查询的表名，将会以字典列表的形式返回查询的表，若查询的表不存在则会报错：

```
def search_all(table_name):
    # openguass属于psycopg2数据库，因此采用psycopg2。connect方法连接
    conn=psycopg2.connect(database='movie_databse',
                          user='gaussdb',password='2001CCMqwe@',
                          host='172.24.160.1',port='15432')
    # 创建cursor对象：
    cursor = conn.cursor()
    # 使用cursor对象来执行SQL语句。
    cursor.execute("select * from "+table_name)
    res=[]
    # 获取语句执行结果
    tmp=cursor.fetchall()
    # 提交语句进行保存
    conn.commit()
    # 获取列名以创建字典。
    cursor.execute("select column_name from information_schema.columns  \
                    where table_schema='public' and
table_name='"+table_name+"';")
    tmp_idx=cursor.fetchall()
    conn.commit()
    conn.close()
    # 创建符合html中格式的字典列表
```

```
    for i in range(len(tmp)):
        res.append({})
        for j in range(len(tmp[i])):
            res[i][tmp_idx[j][0]]=tmp[i][j]
    # 返回结果
    return res
```

从数据库获取到数据，将数据储存在`context`字典列表中，然后可以通过`template.render(context,request))`语句传入html，这样便完成了数据库到html的数据传输，在此以查询信息为例：

```
# 查找页面
def searches(request):
    # 通过POST方法来接受用户在html中输入到搜索框中的信息
    name=request.POST['search']
    print('name:',name)
    # 调用函数来查找
    allmov=search_all('movie')
    match=[]
    # 英文名或中文名
    # 匹配到名字则返回
    for m in allmov:
        # 支持模糊查询
        if name in m['name'] or name in m['foreignname']:
            match.append(m)
    # 该方法用于加载该函数对应的网页（模板）
    template=loader.get_template('searches.html')
    # 以字典的方式传到html
    context={'matches':match,}
    return HttpResponse(template.render(context,request))
```

从html获取数据到后端主要采用的方法有两种：一是通过post方法，在后端函数中获取用户在输入框的输入,如上述代码所示；第二种是通过链接的方法，用户点击链接将链接对应的信息作为参数传入到函数中，例如通过在每个电影下面使用该语句，可以将该电影的id带到infopage页面中，然后根据id显示对应的内容：

```
<div class="tag" ><a href="infopage/{{x.movie_id}}">{{x.name}}</a></div>。
```

在infopage页面对应的函数中可以直接接收该参数：

```
def infopage(request,id):
```

通过这两种方法便可以建立html到数据库的数据传输，因此该数据库应用系统便实现了双向连接。

有了前后端交互的方法，又有了数据库操作的函数，我们就可以处理前端网页发来的请求，返回需要的数据， 或者编写复杂的算法处理前台的逻辑并返回结果。

2.4 数据库设计

该部分为数据库期末大作业的重要内容，因此我将花较多的篇幅来进行介绍。介绍的顺序采用最直观的方法，依次对重要的数据库表进行介绍，并会在介绍中讲解该表的作用和其他表的联系。最后便会给出整个数据库的ER图。

按照作业要求，采用了**openguass数据库**，因此有些语法和平常的sql语法可能不一致。

电影表`movies`

我们在电影信息页面中展示以下信息：

- 电影中英文名
- 电影类型、时长
- 上映时间、地区
- 评分及评分人数
- 剧情简介
- 重映信息（影院名称、影院地址、上映场次信息等）
- 用户评论

因此为了实现展示电影信息的服务，我们需要用一张表来存储电影的基本信息。该表的建立如下，主键设置为`movie_id`，以标识每个电影。字段的类型大多设置为字符串类型。用下列SQL语句创建：

```
CREATE TABLE movie (
 movie_id INTEGER(10) NOT NULL, -- 电影ID
 name varchar(30) NOT NULL, -- 电影名称
 staring varchar(1000) DEFAULT NULL, -- 电影主演
 detail varchar(500) NOT NULL, -- 电影详情
 duration varchar(50) DEFAULT NULL,  -- 电影时长
 type varchar(50) NOT NULL ,-- 电影类型
 score varchar(50) DEFAULT NULL ,-- 评分
 picture varchar(100) NOT NULL ,-- 海报
 commentsCount varchar(30) DEFAULT NULL, -- 参评人数
 releaseDate varchar(100) DEFAULT NULL,-- 上映时间
 foreignName varchar(50) DEFAULT NULL ,-- 电影的外国名
 commentsUnit INTEGER(50) DEFAULT NULL,
 PRIMARY KEY (movie_id)
)
```

影院表`cinema`

为了给出某部电影重映的影院名称、地址等信息，我们需要建立一个影院表，该表记录所有影院的信息，主键设置为`cinema_id`，用于标识每个影院，表中还包含有名称、地址字段。用下列SQL语句创建：

```
CREATE TABLE CINEMA(
CINEMA_ID INTEGER(10) NOT NULL,-- 影院id
NAME VARCHAR(50) NOT NULL,-- 影院名称
ADDRESS VARCHAR(120),-- 影院地址
PRIMARY KEY(CINEMA_ID)
);
```

评论表`comments`

我们需要在电影下面显示第三方用户的评论，因此需要建立一个评论表，该表的主键设置为`comments_id`，外键为`movie_id`，是从movie的`movie_id`的外键引用，用下列SQL语句创建：

```
CREATE TABLE COMMENTS_(
comments_id INTEGER(10) NOT NULL,-- 评论id
user_id INTEGER(10) NOT NULL,-- 用户id
comments_ varchar(300) NOT NULL,-- 评论内容
movie_id INTEGER(10) NOT NULL,-- 对应的电影id
PRIMARY KEY (comments_id)
);

ALTER TABLE `COMMENTS_` ADD CONSTRAINT `COMMENTS_1` FOREIGN KEY
(`movie_id`) REFERENCES `movie` (`movie_id`) ON DELETE CASCADE ON UPDATE
CASCADE;
```

放映厅表`hall`

放映厅表记录每个影院的放映厅，因此`cinema_id`为外键，从影院的id引用。主键为`hall_id`，表中还包含放映厅名字、容量等字段。用下列SQL语句创建：

```
CREATE TABLE hall (
hall_id INTEGER(10) NOT NULL,-- 放映厅id
name varchar(20) NOT NULL,-- 放映厅名称
cinema_id INTEGER(10) NOT NULL,-- 对应的影院
capacity INTEGER(50) NOT NULL,-- 容量
PRIMARY KEY(hall_id)
)

ALTER TABLE `hall` ADD CONSTRAINT `hall_1` FOREIGN KEY
(`cinema_id`) REFERENCES `cinema` (`cinema_id`) ON DELETE CASCADE ON UPDATE
CASCADE;
```

用户表`user_`

该表记录网站的所有用户信息，包括用户名称、邮箱、密码、用户身份（管理员或普通用户）以及用户的图像路径（我们为每个用户分配了随机头像），该表用下列SQL语句创建：

```
CREATE TABLE user_ (
user_id INTEGER(10) NOT NULL ,-- 用户id
uname varchar(30) NOT NULL,-- 用户名
pass_word varchar(30) NOT NULL ,-- 密码
email varchar(30) NOT NULL ,-- 邮箱
role_ INTEGER(10) NOT NULL ,-- 角色(1表示管理员, 0 表示用户)
headImg varchar(30) DEFAULT NULL,-- 头像路径
PRIMARY KEY (user_id)
)
```

语录表`words`

该表主要用于在登录页面显示经典台词，用下列SQL语句创建：

```
CREATE table words(
english VARCHAR(300), -- 英文
chinese VARCHAR(100), -- 中文
from_ VARCHAR(50) -- 出自哪部电影
)
```

播放场次表`session`

我们需要给用户展示某个电影在重映的影院的播放场次信息，因此需要建立场次表`session`，该表包括场次id、场厅id、电影院id、电影id、放映日期、开场时间、票价、剩余座位数等信息。其中`hall_id`是从放映厅的id引用的外键，`movie_id`是从电影的id引用的外键，`cinema_id`是从影院的id引用的外键。用下列SQL语句创建：

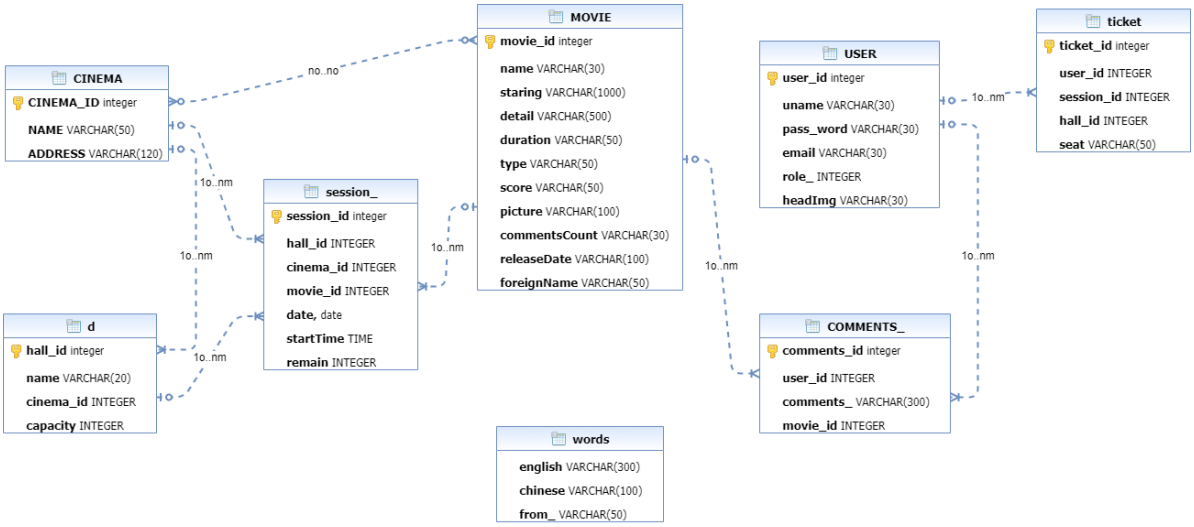
```
CREATE TABLE session_ (
session_id INTEGER(10) NOT NULL, --'场次ID',
hall_id INTEGER(10) NOT NULL, --'场厅ID',
cinema_id INTEGER(10) NOT NULL, --'电影院ID',
movie_id INTEGER(10) NOT NULL, --'电影ID',
date_ date NOT NULL, --'放映日期',
startTime time DEFAULT NULL,  --'开场时间',
price float8 NOT NULL, -- 票价
remain INTEGER(50) NOT NULL,--'剩余座位数',
PRIMARY KEY (session_id)
)

ALTER TABLE `session_` ADD CONSTRAINT `session_1` FOREIGN KEY
(`cinema_id`) REFERENCES `cinema` (`cinema_id`) ON DELETE CASCADE ON UPDATE
CASCADE;
ALTER TABLE `session_` ADD CONSTRAINT `session_2` FOREIGN KEY
(`movie`) REFERENCES `movie` (`movie_id`) ON DELETE CASCADE ON UPDATE CASCADE;
ALTER TABLE `session_` ADD CONSTRAINT `session_3` FOREIGN KEY
(`hall_id`) REFERENCES `hall` (`hall_id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

电影票表`ticket`

该表储存每个用户的电影票信息，由于这部分功能并不是很实用，因此在最后实现时我们删掉了这部分内容，但在此还是介绍一下。用下列SQL语句创建：

```
CREATE TABLE ticket (
ticket_id INTEGER(10) NOT NULL, -- 电影票ID
user_id INTEGER(10) NOT NULL, -- 所属用户
session_id INTEGER(10) NOT NULL, -- 场次ID
hall_id INTEGER(10) NOT NULL, -- 场厅ID
seat varchar(50) NOT NULL, -- 座位号
PRIMARY KEY (ticket_id)
)
ALTER TABLE `session_` ADD CONSTRAINT `ticket_1` FOREIGN KEY
(`user_id`) REFERENCES `user_` (`user_id`) ON DELETE CASCADE ON UPDATE CASCADE;
ALTER TABLE `session_` ADD CONSTRAINT `ticket_2` FOREIGN KEY
(`session_id`) REFERENCES `session` (`session_id`) ON DELETE CASCADE ON UPDATE
CASCADE;
ALTER TABLE `session_` ADD CONSTRAINT `ticket_3` FOREIGN KEY
(`hall_id`) REFERENCES `hall` (`hall_id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

以上便是数据库中所有表的信息、每张表的作用以及表之间的联系。**最后我们整合所有的数据表关系，绘制的ER图模型如下**：



3 数据来源

本次数据库应用系统的数据大部分是通过合法的方式从网上爬取得到，如电影信息是从豆瓣经典电影榜单和高分电影榜单中爬取，电影的评论信息是从猫眼电影的评论区中爬取，影院信息是从美团的附近影院中爬取得到，但为了避免不必要的纠纷，我们在爬取到的信息上做了一定的修改。

除此之外，还有小部分信息是我们收集资料再输入到数据库中的，如电影的经典台词等等。

以爬取电影信息为例，我们通过爬取主页面的方式先获得了所需要网站的页面信息，即各个热门电影的网页信息，然后我们将这部分信息储存到mongodb中。

第一部分爬取到的信息存储在数据库中如下：

```
_id: ObjectId('63955707dd256f14408ebb74')
id: "36171755"
url: "https://movie.douban.com/subject/36171755/"
名称: "爱在空气中 番外篇"
海报: "https://img1.doubanio.com/view/photo/s_ratio_poster/public/p2884479088…"
评分: "8.7"
```

然后就是通过get请求访问每一个页面并对get后的页面进行解析即可获得所需的页面信息进行解析即可。解析代码如下：

```
response=session.get(url=url,headers=headers).text
            response=BeautifulSoup(response,'html.parser')
            list=response.find('div',id='info')
            list1=list
            #print(list1.text)
            ans=list1.text
            ans1=ans.replace('\n',',')
            data['部分信息']=ans1[1:-1]
            data['详情']=response.find('div',id='link-report-
intra').find_all('span')[0].text.replace('
','').replace('\n','').replace('\u3000','')
            data['参评人数']=response.find('span',property='v:votes').text
            #print(data)
            fun=list1.find('br')
```

然后同上将爬取到的页面详细信息储存起来。

因为网页读取问题，将部分网页信息存储为详情这个标签。

网页具体信息存储到数据库中如下：

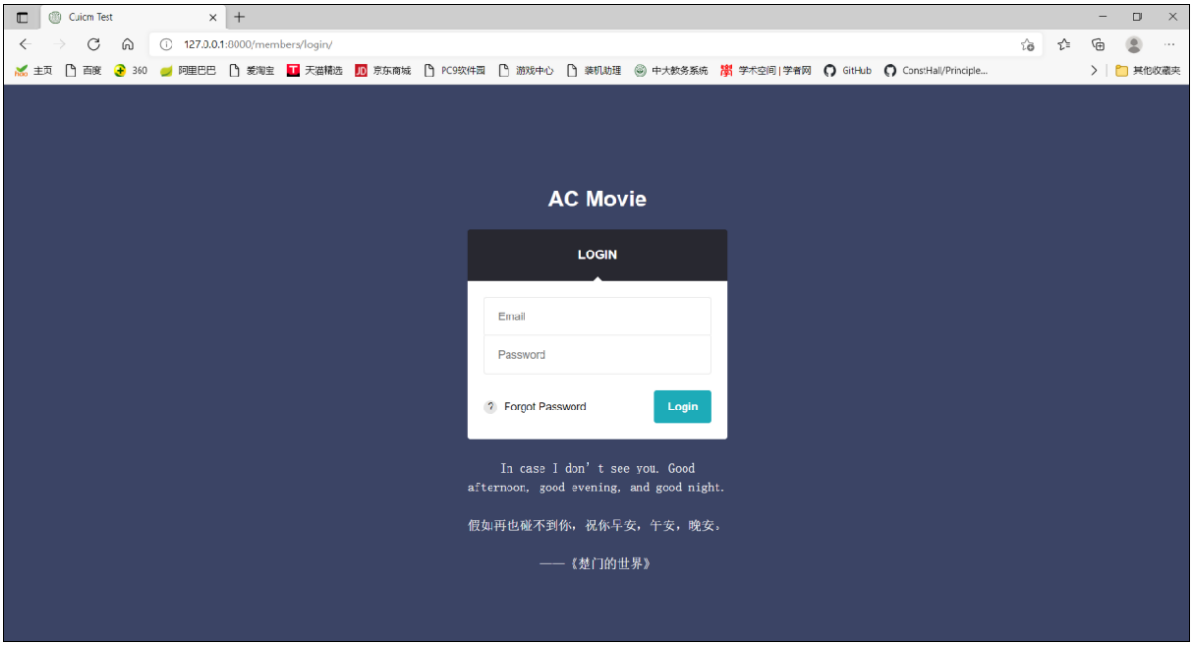
以上就是爬虫相关部分的内容，在爬取之中还是遇到了一些反爬的机制上的问题，但最终通过`time.sleep()`,以及及时更新时间戳等方式解决了爬取方面的问题。同时可以将数据库内容导出为excel方便读取和使用。

4 实现效果展示

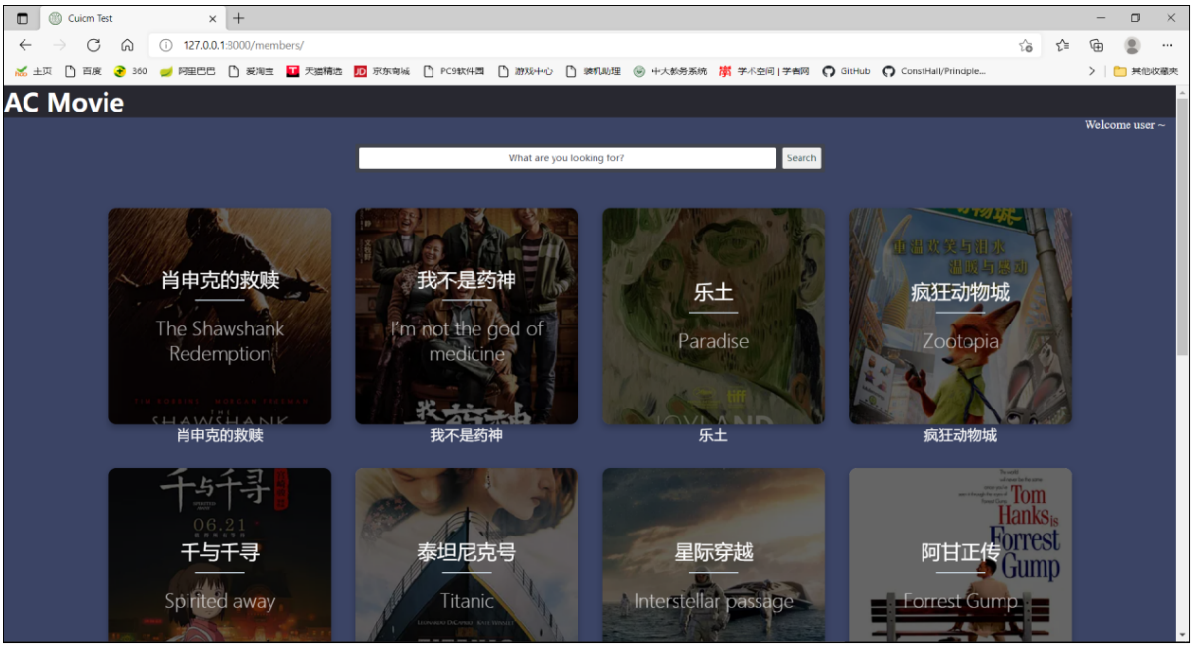
最后的实现效果演示可以在附件中的演示视频中看到。由于网站连接的是我本地的数据库，因此要我在我的电脑上开着数据库才可以进行连接，故难以生成一个随时可以访问的网址，因此我们采用了录制视频的方式来展示最后的结果。

主要运行界面

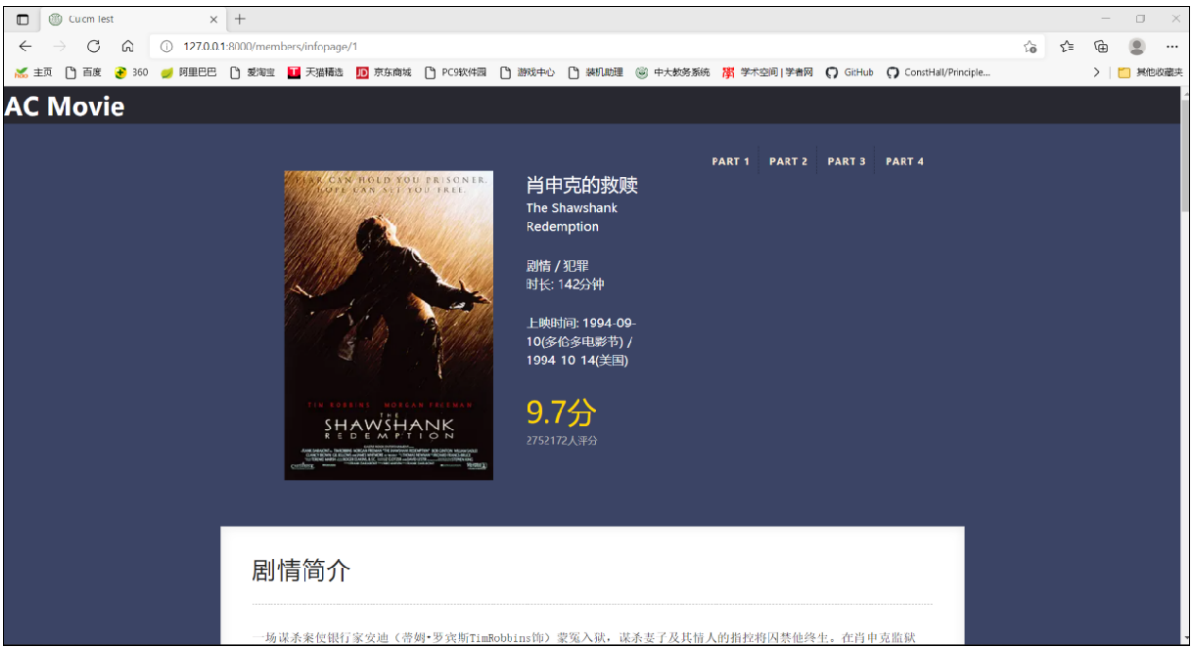
登录页面，输入正确的邮箱和密码后才可以登录，并且会根据用户的身份跳转到特定的操作页面：



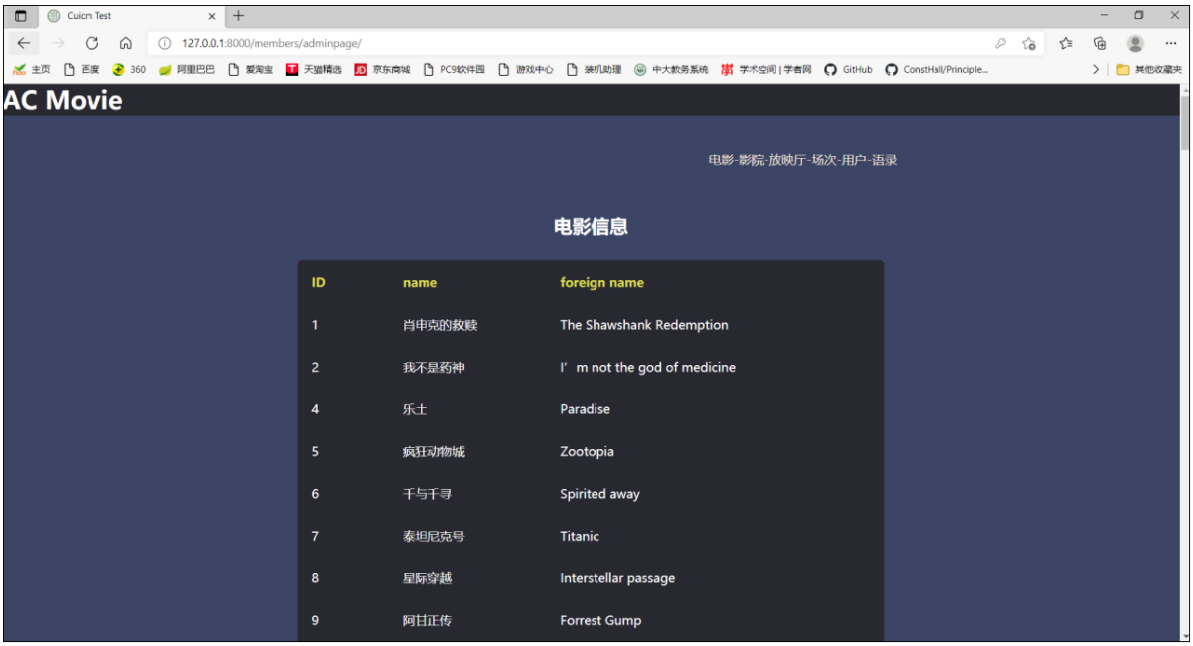
普通用户的主页，用户可以查看电影的海报、预览信息，点击电影则会进入对应的信息页面，在搜索框中可以搜索想了解的电影：



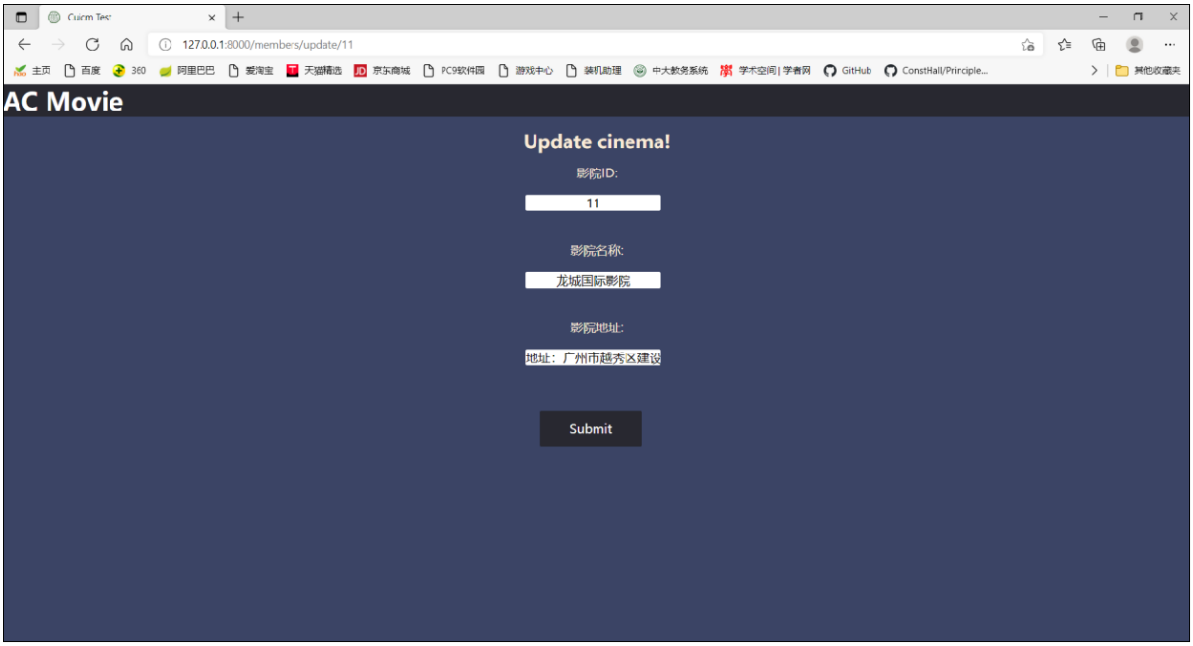
点击进入某个电影的信息页面，可以看到许多详细信息：



管理员页面：



增加信息页面：



5 总结&感想

这次的数据库期末大作业对我们来说是一次很有意义的挑战。最具挑战性的方面之一是设计合适的数据库架构，以有效地存储和组织网站的所有必要数据。仔细考虑不同实体之间的关系以及它们在架构中的表示方式也需要一些时间。

另一个挑战是实施网站功能所需的各种数据库操作。这包括插入和更新电影和电影信息等任务，以及查询数据库以检索要在网站上显示的数据。我们发现将这些任务可以分解为更小的步骤并一次完成一个步骤，这在我们在开发的过程中进行测试和调试很有帮助。

一个学期的数据库系统课程的学习结束了，在经过了这个学期的学习后，我对数据库有了更深入的理解和认识。在这门课程中，我学习了数据库的基本概念，包括数据库系统的架构和功能，以及数据库设计的基本原则。我还学习了 SQL 语言，这是一种用于访问和操纵数据库的标准语言。

我认识到数据库在许多方面都是非常重要的，例如在商业中用于存储客户信息和订单信息，在医疗信息系统中用于存储病人信息和医疗记录，在社交媒体中用于存储用户信息和内容。数据库还被广泛用于研究和数据分析。

在学习这门课程后，我认识到数据库的设计是非常重要的，因为它将直接影响数据库的性能和可用性。我还学习了数据库的优化方法，包括索引、分区和视图，这些方法可以帮助我们提高数据库的性能。

总的来说，学习数据库课程是一次非常有意义的经历，它让我对数据库有了更深入的理解，并且我相信这些知识将会为我的未来职业生涯提供很大的帮助。