



# 上海海事大学

SHANGHAI MARITIME UNIVERSITY

## 电影推荐系统

### Movie recommendation system

课程名称：\_\_\_\_\_ 软件项目管理 \_\_\_\_\_

项目名称：\_\_\_\_\_ 电影推荐系统 \_\_\_\_\_

指导教师：\_\_\_\_\_ 刘 晋 \_\_\_\_\_

成 员：\_\_\_\_\_ 肖 品 202030310304 \_\_\_\_\_

成 员：\_\_\_\_\_ 王子箐 202030310190 \_\_\_\_\_

成 员：\_\_\_\_\_ 邓盼盼 202030310178 \_\_\_\_\_

日 期：\_\_\_\_\_ 2021 年 6 月 16 日 \_\_\_\_\_

# 目 录

1 项目展示 .....	5
1.1 项目设计 .....	5
1.1.1 功能模块介绍 .....	5
1.1.2 算法设计介绍 .....	5
1.1.3 数据流图 .....	7
1.1.4 数据库设计 .....	7
1.2 项目代码展示 .....	8
1.2.1 用户注册登录 .....	8
1.2.2 用户评分 .....	9
1.2.3 推荐算法 .....	9
2 项目执行 .....	11
2.1 人员分工情况 .....	11
2.2 WBS .....	11
2.3 里程碑与交付物 .....	11
2.4 计划执行折线图 .....	13
3 测试计划 .....	14
3.1 人工测试结果 .....	14
3.2 自动化测试结果 .....	15

4 课程学习 .....	18
4.1 项目计划 .....	18
4.2 项目开发 .....	18
5 总结 .....	19
6 未来规划 .....	20
6.1 电影模块完善 .....	20
6.2 推荐模块完善 .....	20
6.3 用户模块完善 .....	20
6.4 其他 .....	20

## 摘 要

我们进行电影推荐系统设计的原因是目前处于信息爆炸的时代，成千上万的网络信息充斥在我们的生活方方面面。面对海量的数据，用户往往感到束手无策、无处下手。如何帮助用户从海量的信息中获取用户最感兴趣的信息逐渐成为当今的热门研究之一。通过个性化推荐系统，系统分析用户的历史数据对用户的兴趣爱好进行建模。在用户使用系统的过程中，记录并学习每个用户的兴趣，及时更新。用户无需特意填写大量的兴趣调查信息，可以减轻用户负担，提高用户认可度。此报告包含最终项目的展示、项目执行情况、测试用例及测试代码、课程学习与总结、下一个周期的计划等。

# 1 项目展示

## 1.1 项目设计

对功能模块、算法、数据流及数据库设计进行介绍。

### 1.1.1 功能模块介绍

电影推荐系统分为电影模块、推荐系统模块、用户模块，三个功能模块。

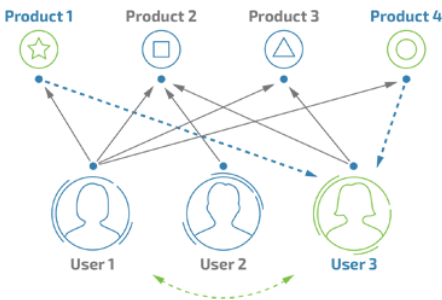
- (1) 电影模块功能包括：上新电影，按类别、标签查看电影，按名称搜索电影。
- (2) 推荐系统模块功能包括：推荐算法的运行，后台信息更新，热门及高分排行榜单，相似电影推荐。
- (3) 用户模块功能包括：用户注册、登录，用户评论，用户收藏，用户浏览电影，添加兴趣标签。

### 1.1.2 算法设计介绍

为在计划实现内按时交付，本系统采用易于实现且具有成效的推荐算法，即协同过滤算法。

#### (1) 协同过滤算法

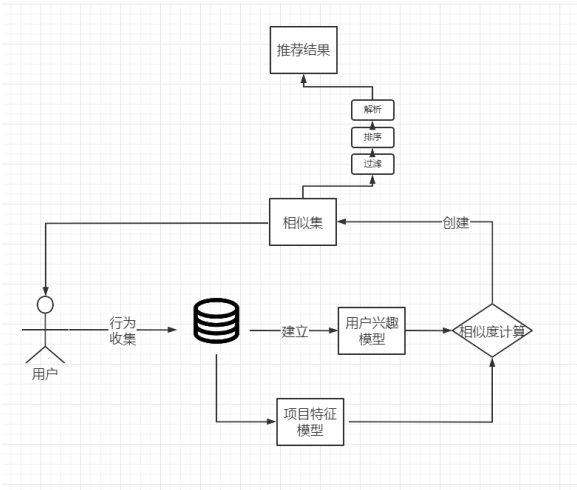
在本电影推荐系统中，基于用户的协同过滤算法是指通过用户的历史行为数据发现用户对 item 或 content 的喜欢，如收藏，评分等行为。根据不同用户对相同电影或内容的偏好程度计算用户之间的关系。在有相同喜好的用户间进行相同电影推荐。简单的说就是如果 User1, User2 两个用户都收藏了 Product1、Product2、Product3 相同的三部电影，并且给出了 5 星的评价。那么 A 和 B 就属于同一类用户。可以将 User1 看过的 Product1 也推荐给 User2。同样的，可以对于电影也进行一个协同过滤的操作，之后进行推荐。



图一 基于用户协同过滤算法的示意图

协同过滤算法主要分为两个步骤：①寻找相似的用户集合；②寻找集合中用

户喜欢的且目标用户没有的进行推荐。



图二 推荐算法具体设计图

(2) 用户行为

要从用户的行为和偏好中发现规律，并基于此给予推荐，如何收集用户的偏好信息成为系统推荐效果最基础的决定因素。用户有很多方式向系统提供自己的偏好信息，而且不同的应用也可能大不相同，此电影推荐系统以以下（根据可靠程度降序）：

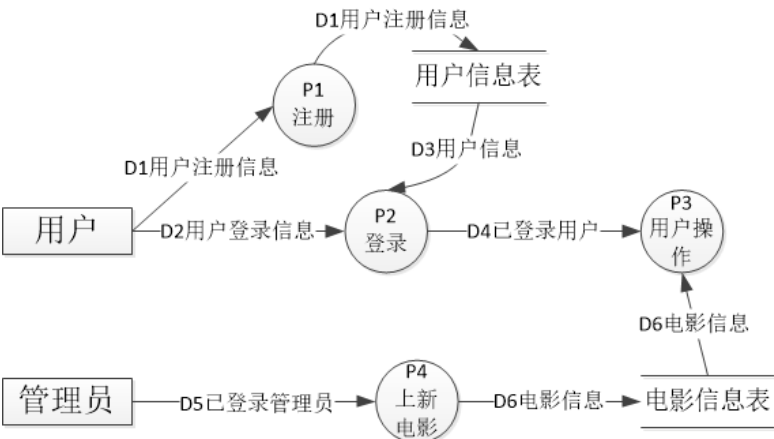
表一 用户行为数据

用户行为	类型	特征	作用
提交评分	显式	整数量化的偏好，取值是 [0, 5] ；	通过用户对电影的评分, 可以尤其精确的得到用户的偏好
收藏	显式	布尔量化的偏好，取值是 0 或 1	通过用户对物品的投票, 可以较精确的得到用户的偏好
前往观看	隐式	布尔量化的偏好，取值是 0 或 1	用户的购买是很明确的说明这个项目它感兴趣。
点击量 ( 查看 )	隐式	一组用户的点击, 用户对物品感兴趣, 需要进行分析, 得到偏好	用户的点击一定程度上反映了用户的注意力, 所以它也可以从一定程度上反映用户的喜好。

评论	显示	一段文字，需要进行文本分析，得到偏好	通过分析用户的评论，可以得到用户的情感：喜欢还是讨厌
----	----	--------------------	----------------------------

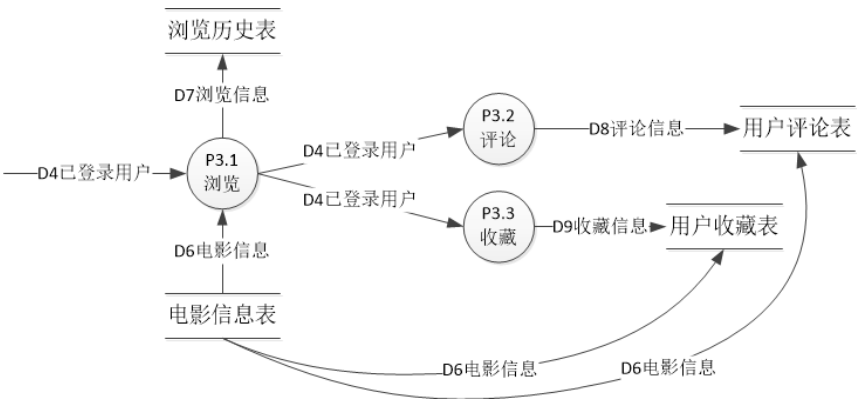
### 1.1.3 数据流图

电影推荐系统的顶层数据流图如下图所示：



图三 电影推荐系统顶层数据流图

操作 P3 的用户操作层数据流图如下图所示：



图四 用户操作数据流图

### 1.1.4 数据库设计

电影推荐系统的数据库设计如下图所示：





## 1.2.2 用户评分

用户在电影详情页面浏览后，可以基于电影内容、其他用户评论、个人想法等对此电影进行评分。系统在收集信息后，会对用户数据进行更新。部分代码如下：

```
def login_in
# 电影评分 在打分的时候清除缓存
def score(request, movie_id):
    user_id = request.session.get("user_id")
    user = User.objects.get(id=user_id)
    movie = Movie.objects.get(id=movie_id)
    score = float(request.POST.get("score"))
    get, created = Rate.objects.get_or_create(user_id=user_id, movie=movie, defaults={"mark": score})
    if created:
        for tag in movie.tags.all():
            prefer, created = UserTagPrefer.objects.get_or_create(user_id=user_id, tag=tag, defaults={"score": score})
            if not created:
                # 更新分数
                prefer.score += (score - 3)
                prefer.save()
            print('create data')
        # 清理缓存
        user_cache = USER_CACHE.format(user_id=user_id)
        item_cache = ITEM_CACHE.format(user_id=user_id)
        als_cache = ALS_CACHE.format(user_id=user_id)
        cache.delete(user_cache)
        cache.delete(item_cache)
        cache.delete(als_cache)
        print('cache deleted')
        update_item_movie_sim_matrix(movie_id, user_id)
        user_cf.update_all_user(user=user)
    return redirect(reverse("movie", args=(movie_id,)))
```

## 1.2.3 推荐算法

系统分别基于用户和电影实现推荐。

基于用户推荐会根据当前用户的评分、标签收藏、浏览度等进行推荐。部分代码如下：

```
class UserCf:
    # 获得初始化数据
    def __init__(self, all_user):...
    # 通过用户名获得商品列表，仅调试使用
    def getItems(self, username1, username2):...
    # 计算两个用户的皮尔逊相关系数
    def pearson(self, user1, user2):...
    # 计算与当前用户的距离，获得最临近的用户
    def nearest_user(self, current_user, n=1):...
    # 给用户推荐商品
    def recommend(self, username, n=3):...
    # 某个用户给电影打分后，更新all_user dict
    def update_all_user(self, user):...

# 入口函数
def recommend_by_user_id(user_id):
    user_prefer = UserTagPrefer.objects.filter(user_id=user_id).order_by('-score').values_list('tag_id', flat=True)
    current_user = User.objects.get(id=user_id)
    # 如果当前用户没有打分 则看是否选择过标签，选过的话，就从标签中挑
    # 没有的话，就按照浏览数量推荐15个
    if len(user_prefer) == 0:
        if len(current_user.rate_set.count()) != 0:
            movie_list = Movie.objects.filter(tags__in=user_prefer)[:15]
        else:
            movie_list = Movie.objects.order_by("-num")[:15]
        return movie_list
    import random
    recommend_list = [each[0] for each in user_cf.recommend(current_user.username, 15)]
    movie_list = list(Movie.objects.filter(id__in=recommend_list)[:15])
    random.shuffle(movie_list)
    other_length = 15 - len(movie_list)
    if other_length > 0:
        fix_list = Movie.objects.filter(~Q(rate__user_id=user_id)).order_by('-num')
        for fix in fix_list:
            if fix not in movie_list:
                movie_list.append(fix)
            if len(movie_list) >= 15:
                break
    return movie_list
```

基于电影推荐会根据电影的标签、电影相似度等进行推荐。代码如下：

```
def recommend_by_item(user_id, k=15):
    # 用户的tag
    user_prefer = UserTagPrefer.objects.filter(user_id=user_id).order_by('-score').values_list('tag_id', flat=True)
    user_prefer = list(user_prefer)[1:]
    current_user = User.objects.get(id=user_id)
    # 如果当前用户没有打分 则希望返回看过标签、看过的话，就从标签中找
    # 没有的话，就按照时间顺序找15个
    if current_user.rate_set.count() == 0:
        if len(user_prefer) != 0:
            movie_list = Movie.objects.filter(tags__in=user_prefer)[:15]
        else:
            movie_list = Movie.objects.order_by('-num')[:15]
        print('from here')
        return movie_list
    # most_tags = Tags.objects.annotate(tags_sum=Count('name')).order_by('-tags_sum').filter(movie__rate__user_id=user_id).order_by('-tags_sum')
    # 选用户最喜欢的标签中的电影，用户没看过的10部，对这10部电影，计算距离最近
    un_watched = Movie.objects.filter(~Q(rate__user_id=user_id), tags__in=user_prefer).order_by('?')[:30] # 看过的电影
    watched = Rate.objects.filter(user_id=user_id).values_list('movie_id', 'mark')
    distances = []
    names = []
    # 在未看过的电影中寻找
    # 后续改进，选择top15
    for un_watched_movie in un_watched:
        for watched_movie in watched:
            if un_watched_movie not in names:
                names.append(un_watched_movie)
                distances.append((similarity(un_watched_movie.id, watched_movie[0]) * watched_movie[1], un_watched_movie))
    distances.sort(key=lambda x: x[0], reverse=True)
    print('this is distances', distances[:15])
    recommend_list = []
    for mark, movie in distances:
        if len(recommend_list) >= k:
            break
        if movie not in recommend_list:
            recommend_list.append(movie)
    # print('this is recommend list', recommend_list)
    # 如果得不到有效数量的推荐 按照未看过的电影中的热度进行填充
    print('recommend list', recommend_list)
    return recommend_list
```

## 2 项目执行

在此对项目执行的计划与现实进行对比展示。

### 2.1 人员分工情况

人员安排与计划基本一致，合理的分工以及按照分工计划执行帮助我们按时的完成开发任务。

表二 人员分工表

姓名	计划工作内容	实际工作内容
肖品	产品规划设计、产品评审收、需求可行性评估、开发架构、算法研究、部署上线、开发优化	产品规划设计、需求可行性评估、开发架构、算法研究、部署上线、开发优化、自动化测试
王子箐	制定开发计划、任务分解 wbs、项目规划、交付、评审、测试计划、测试用例编写、测试方案、功能测试	制定开发计划、任务分解 wbs、项目规划、测试计划及测试用例编写、报告撰写、开发优化
邓盼盼	需求调研、业务功能梳理、原型设计、需求确认、UI 设计、接口协议设计、编码规范、开发、bug 处理	需求调研、业务功能梳理、需求确认、UI 设计、接口协议设计、开发优化、bug 处理

### 2.2 WBS

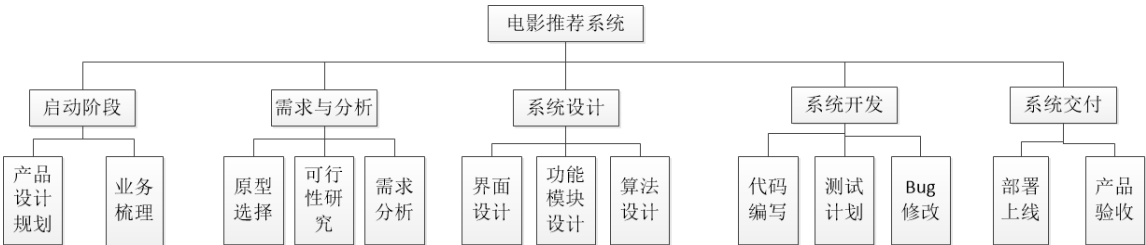


图 2.1 WBS 图

### 2.3 里程碑与交付物

里程碑安排如下图所示：

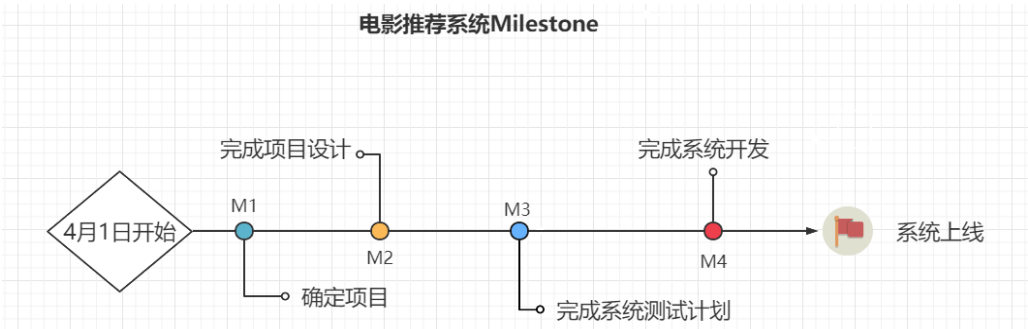


图 2.2 里程碑图

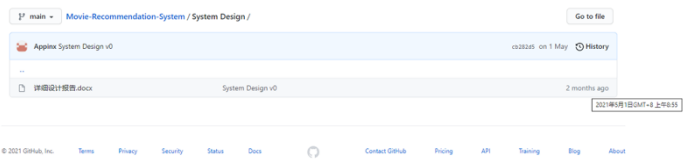
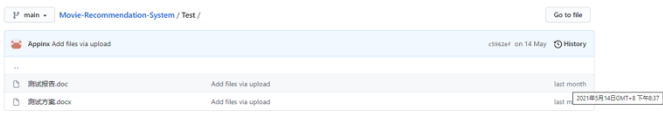
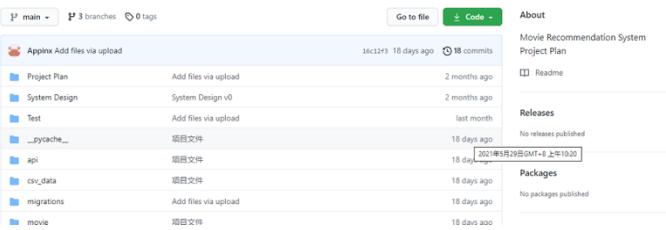
交付物及交付时间如下表：

表三 里程碑交付物表

里程碑	交付物	计划交付日期	实际交付日期
确定项目范围	项目计划书	2021.04.22	2021.04.22
完成项目设计	系统详细设计、测试用例	2021.04.30	2021.05.01
完成系统测试计划	测试方案、测试报告	2021.05.14	2021.05.14
完成系统开发	系统开发源码	2021.05.28	2021.05.29
系统上线	可运行的网站项目	2021.06.08	2021.06.05

实际交付物及交付时间如下表：

表四 实际交付表

交付物	交付日期
<div><p>项目计划上传</p></div>	4.22
<div><p>系统详细设计上传</p></div>	5.1
<div><p>系统测试文档上传</p></div>	5.14
<div><p>系统代码上传</p></div>	5.29

```
Server Software:      V2SIServer/0.2
Server Hostname:      101.132.155.8
Server Port:          8000
Document Path:        /
Document Length:      32473 bytes
Concurrency Level:     10
Time taken for tests:  477.062 seconds
Complete requests:     10000
Failed requests:        0
Total transferred:     328320000 bytes
HTML transferred:      324720000 bytes
Requests per second:   20.96 [#/sec] (mean)
Time per request:      477.062 [ms] (mean)
Time per request:      47.706 [ms] (mean, across all concurrent requests)
Transfer rate:         673.13 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median  max
Connect:      4   12+76.9      4    4021
Processing:   38  464+624.8    305  27244
Waiting:      23  146+150.7     88   4149
Total:        44  477+627.1    313  27251

Percentage of the requests served within a certain time (ms)
 50%    313
 66%    389
 75%    503
 80%    649
 90%   1082
 95%   1577
 98%   1988
 99%   2492
100%   27251 (longest request)

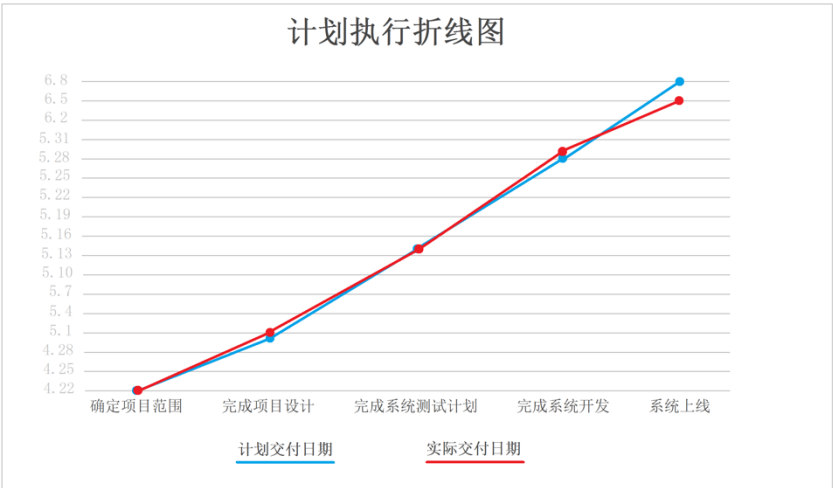
C:\Users\Administrator\Desktop\httpd-2.4.48-ollk-x86-vc15\Apache24\bin>
```

6.5

上线并测试服务器并发

## 2.4 计划执行折线图

计划执行对比折线图如下图所示：



图八 计划执行对比折线图

### 3 测试计划

#### 3.1 人工测试结果

项目名称		电影推荐系统			
序号	主要模块	功能点	预期测试结果	实际测试结果	需完善内容
1	电影模块	上新电影	管理员可以通过系统上架新的电影 填写电影的描述和分类信息并将电影上架	可以完成电影上新，填写电影描述与分类。对各项内容不为空约束。对链接内容有格式约束。	填写内容时，用户体验感不流畅，很多内容需要大量手动操作。部分操作可以改善为自动填写。
2		按主演、导演查看相关电影	用户可以在电影详情页针对具体主演或导演进行进一步查找	因时间及各方面原因没有完成此功能。	在系统未来的周期中可以对此功能进行完善。
3		按类别查看电影	用户可以根据分类进行电影的查看	可以完成根据随意分类进行电影的查看。	基本无需完善，可以对标签查看功能进行升级，实现多标签选择。
4		按名称搜索电影	用户可以直接搜索电影名称查找电影	可以完成搜索电影名称查找电影。	基本无需完善。
5	推荐系统模块	推荐算法的运行	用户提供评分、收藏、标签信息系统进行计算，及时反馈，推送给用户个性化的电影推荐信息	可以根据评分收藏操作对推荐内容进行更新。	基本无需完善，后期可以研究更多个性化推荐算法，对个性化推荐性能加以提升。
6		后台信息更新	用户提供历史数据，系统对用户的兴趣爱好进行建模为用户提供基于个性化推荐的电影信息	可以根据评分收藏操作对推荐内容进行更新。	基本无需完善，后期可以研究更多个性化推荐算法，对个性化推荐性能加以提升。
7		热门及高分排行榜单	在电影信息更新后，系统基于大数据的分	可以根据评分对电影进行热度排行。	基本无需完善，可以对热门排行进行更细致

			析，对所有的电影进行热度排行		的分类展示作为提升。
8		相似电影推荐	用户在电影的详情页面可以看到与该电影相似的其它电影	可以实现相似电影推荐功能。	基本无需完善。
9	用户模块	用户注册	用户可以提交注册信息，完善个人信息页面	可以实现用户注册功能，对用户名密码均有不为空和长度约束，对用户名有主键约束。	可以增加对密码复杂度的约束。
10		登录	用户登录后可以进行各种用户操作以及在个人信息页面查看自己的评论和收藏	可以实现用户登录功能，对不相符的用户名密码可以甄别。	可以增加验证码约束。
11		用户评论	用户可以基于电影内容、其他用户评论、个人想法等对此电影进行评论与评分	可以实现评分评论功能。有内容不为空的约束。	基本无需完善。
12		用户收藏	用户可以在电影详情页面浏览后，可以将感兴趣的电影收藏至个人收藏夹	可以实现收藏和取消收藏功能。	基本无需完善。
13		用户浏览电影	用户根据排行榜或搜索或首页推荐进入电影详情页进行浏览	可以实现浏览功能。部分电影会因为没有版权的原因无法正常显示。	在出现无版权情况时增加显示原因的报错页面以及给用户一个反馈通道。
14		添加兴趣标签	在注册时用户可以选择兴趣标签，在使用系统过程中也可以根据标签查找电影	可以在用户注册时选择标签。 可以实现根据标签查看电影。	基本无需完善。

### 3.2 自动化测试结果

自动化测试脚本包括：1) 清空所用用户 2) 随机生成用户以及评分 3) 清空所有结果矩阵 4) 重新计算结果矩阵

清空用户测试代码展示：

```
if __name__ == '__main__':

    #清空用户
    clear_user()

    # 随机生成用户打分(参数为生成数量)
    populate_user_rating(10)

    #清空rate
    clear_rate()

    #计算用户rate
    populate_rating()
```

生成用户测试代码展示

```
#生成用户(个数)
def populate_user_rating(user_numbers):
    for i in range(user_numbers):
        user_name = random_user_name()
        print("-----"+user_name+"-----")
        try:
            user, created = User.objects.get_or_create(username=user_name,
                                                         defaults={'password': user_name, "email": random_user_name() + '@163.com'})

            for movie_id in random_movie_id():
                rm = random_mark()
                mo = movie_id
                Rate.objects.get_or_create(user=user, movie_id=mo, defaults={'mark': rm})
                print(mo, rm)
        except Exception as e:
            raise e
```

```
def populate_user_rating(user_numbers):
    for i in range(user_numbers):
        user_name = random_user_name()
        print("-----"+user_name+"-----")
        try:
            user, created = User.objects.get_or_create(username=user_name,
                                                         defaults={'password': user_name, "email": random_user_name() + '@163.com'})

            for movie_id in random_movie_id():
                rm = random_mark()
                mo = movie_id
                Rate.objects.get_or_create(user=user, movie_id=mo, defaults={'mark': rm})
                print(mo, rm)
        except Exception as e:
            raise e
```

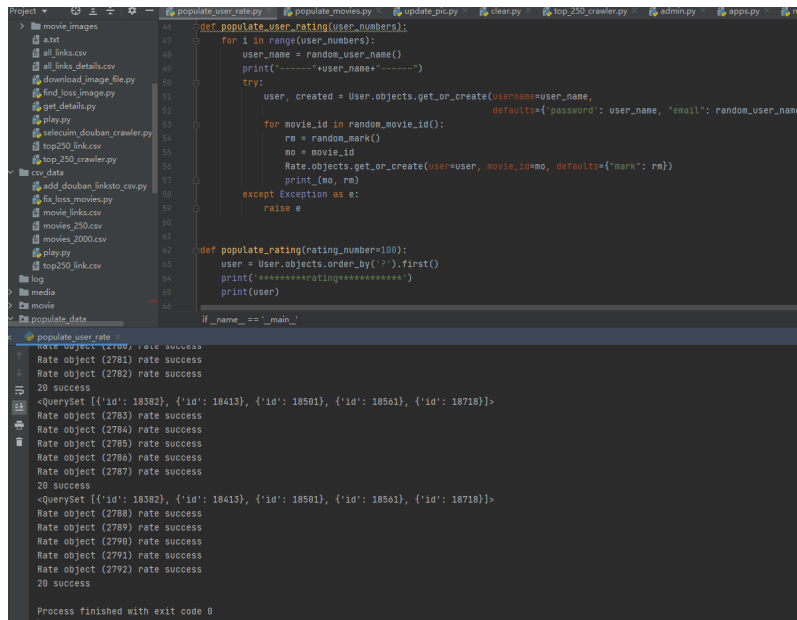
```
def populate_rating(rating_number=5):
    user = User.objects.order_by('?').first()
    print('*****rating*****')
    print(user)

    num = 0
    while num < rating_number:
        for movie_id in random_movie_id():
            num += 1
            try:
                rate = Rate.objects.create(user=user, movie_id=movie_id, mark=random_mark())
                print(rate, 'rate success')
            except Exception:
                pass
        print(rating_number, 'success')
```



```
def populate_movies(filename):
    opener = open(filename, 'r')
    reader = csv.reader(opener)
    next(reader)
    for line in reader:
        id, title, image_link, country, years, director_description, leader, star, description, tags, imdb, language,
        origin_years = years
        # 数据清洗
        years = re.search(pattern=r'\d{4}?(-\d{0,2})?(-\d{0,2})?', string=years)
        if years is None:
            years = origin_years.split('/')[0]
        else:
            years = years[0]
        res = re.match('\d+', star)
        int_d_rate_num = int(res[0]) if res else 0
        # res = re.match('[\u4e00-\u9fa5]+', title)
        # if res is not None:
        #     title = res.group()
        pic_name = 'movie_cover/' + image_link.split('/')[-1]
        years = parse_time(years)
        leader = '/'.join(ast.literal_eval(leader))
        douban_id = douban_link.split('/')[-2]
        # 存入数据库
        movie, created = Movie.objects.get_or_create(name=title, defaults={'image_link': pic_name, 'country': country,
            'years': parse_time(years), 'leader': leader})
```

随机生成用户以及评分测试代码及结果展示(生成5位用户,为标签“运动”  
的电影随机打分)



```
def populate_user_rating(user_numbers):
    for i in range(user_numbers):
        user_name = random_user_name()
        print("-----+user_name+-----")
        try:
            user, created = User.objects.get_or_create(username=user_name,
                defaults={'password': user_name, 'email': random_user_name()})
            for movie_id in random_movie_id():
                rm = random_mark()
                mo = movie_id
                Rate.objects.get_or_create(user=user, movie_id=mo, defaults={'mark': rm})
                print(mo, rm)
            except Exception as e:
                raise e

def populate_rating(rating_numbers=100):
    user = User.objects.order_by('?').first()
    print("*****rating*****")
    print(user)

if __name__ == '__main__':
```

populate\_user\_rate success  
Rate object (2781) rate success  
Rate object (2782) rate success  
20 success  
<QuerySet [(1'id': 18382), (1'id': 18413), (1'id': 18501), (1'id': 18561), (1'id': 18718)]>  
Rate object (2783) rate success  
Rate object (2784) rate success  
Rate object (2785) rate success  
Rate object (2786) rate success  
Rate object (2787) rate success  
20 success  
<QuerySet [(1'id': 18382), (1'id': 18413), (1'id': 18501), (1'id': 18561), (1'id': 18718)]>  
Rate object (2788) rate success  
Rate object (2789) rate success  
Rate object (2790) rate success  
Rate object (2791) rate success  
Rate object (2792) rate success  
20 success  
Process finished with exit code 0

电影推荐系统

首页 标签 进入后台 LGCZQ 退出登录

推荐吧! 爸爸 DANGAL



标签: 剧情 家庭 传记 运动  
国家: 印度 导演: 尼提希·瓦里  
主演: 阿米尔·汗/法缇玛·萨那·沙卡/桑亚·玛荷塔/阿帕尔夏克提·海拉尼/沙希德·卡普尔/康伊拉·沃西/苏哈  
介绍: 马哈维亚(阿米尔·汗 Amir Khan 饰)曾经是一名前途无量的摔跤运动员,在放弃了职业生涯后,他最大的遗憾就是没有能够为国家赢得金牌。马哈维亚将这份希望寄托在了尚未出生的儿子身上,希望儿子能够给他生了两个女儿,取名吉塔(法缇玛·萨那·沙卡 Fatima Sana Shaikh 饰)和巴比塔(桑亚·玛荷塔 Sanya Malhotra 饰)。让马哈维亚没有想到的是,两个姑娘展现出了杰出的摔跤天赋,让他...  
收藏人数: 0 点击收藏  
豆瓣评分: 9.0 添加评分 1.0 提交  
前往观看

猜你喜欢 (USERBASED)

谍影特工 The November Man

上映日期:2016-03-04  
导演:罗杰·唐纳森  
豆瓣评分:6.4

霸王别姬

上映日期:1993-01-01  
导演:陈凯歌  
豆瓣评分:9.6

东北插班生

上映日期:2017-07-15  
导演:姜秉辰  
豆瓣评分:6.2

无间道 無間道

上映日期:2002-12-12  
导演:刘伟强  
豆瓣评分:9.2

猜你喜欢 (ITEMBASED)

LEON THE PROFESSIONAL

SHAWHANK

EDGE OF DARKNESS

## 4 课程学习

在软件项目管理这门课程中，我们受益匪浅。学习到了很多关于产品开发过程的要点以及开发各个步骤中的注意事项，在此进行简要的整理。

### 4.1 项目计划

开发系统最首要的任务就是确定项目范围和目标，知道了要做什么之后才能更好的规划该怎么做。

在开始设计规划之前，要先明确项目开发的环境，即：项目的特点、成员的角色等。

人员分工的重要性，每个人员都要根据各自的擅长来安排任务，并不是每个人都参与到项目的每个步骤才能开发出好的产品，并且成员之间需要定期交流讨论。

完成项目计划书可以帮助我们明确开发任务，包括：WBS 规划、工作的内容、每个工作的工作量、交付计划及里程碑、变更管理等。

风险是一定会存在的，我们要正确看待风险，进行风险预测与评估，在风险发生时才能更可能的减少损失。

### 4.2 项目开发

利用管理工具和图表来协助开发，比如甘特图可以表明每个工作有多少任务量以及各个任务的先后关系，可以帮助我们队工作任务进行安排。

执行过程中，要把理论付诸实践，对执行情况和交付物进行监管，一个好的计划加上好的执行才能完成好的产品。

在开发整个项目的过程中都需要进行测试，要尽早的发现问题解决问题，一旦问题在最后才被发现将会耗费很多的精力来解决。

大多数项目是由交付物驱动的，所以我们需要规划交付物与交付时间。交付物一般包括计划书、设计书、测试计划、实际的产品等。

在项目提交之后也不意味着结束，一个好的产品需要长久的进行维护与完善。用户需求可能会发生变更、运行环境也会升级，所以需要有人员进行维护，系统需要不断的迭代来完善自身。

## 5 总结

电影推荐系统的开发用人三人用时 54 天，这次开发的过程，我们小组成员互相了解、一起讨论、安排工作分工，完成了确定项目范围、完成项目设计、完成系统测试计划、完成系统开发、系统上线等任务。

产品开发将预期的功能点基本实现。功能点包括：推荐系统的首页对热门电影排序进行展示，对各类型电影进行部分的展示；新用户在初次使用系统时，会先提交个人信息进行注册，勾选兴趣标签；用户登录后可以按类别查看电影，按名称搜索电影，在电影详情页面进行浏览、评论、收藏操作，完善个人信息；系统在收集了用户的浏览历史、评分、收藏、标签后会进行计算，及时反馈，推送给用户个性化的电影推荐信息；管理员可以将新电影上新到系统中并且完善电影的具体信息，以便推荐使用等。

每个成员都很好的完成了自己的任务，我们会轮流担任项目组长以及每周定期对上周的任务完成情况和下一周的任务安排进行讨论。事实上，定期的面对面交流讨论非常的有利于我们按时保质的完成项目计划。

在开发过程中，我们遇到了一些预期之中以及预料之外的情况。在自己了解不够专业的领域内开发产品很难抓住需求与功能，所以我们浏览了很多同类型的网站，根据他们的功能以及他们用户评论的需求来设计自己的系统，希望完成一个能够满足大部分用户需求的好产品。

电影推荐系统虽然已经交付，基本功能点也已经实现，但它仍然会有很多需要继续完善和维护的地方，包括界面的美化、电影的及时上新、用户操作的拓展、系统功能的拓展等等。我们会继续对系统开发加以思考，接触了解更多的案例，更好的提升自己的能力。

## 6 未来规划

在此对电影推荐系统未来周期中可做的完善进行规划。

### 6.1 电影模块完善

(1) 电影上新填写电影内容时，用户体验感不流畅，很多内容需要大量手动操作，部分操作可以改善为自动填写。

(2) 按标签浏览电影时可以对功能进行升级，实现标签多重选择。

### 6.2 推荐模块完善

(1) 研究更多个性化推荐算法，对个性化推荐性能加以提升。

(2) 可以对热门排行进行更细致的分类展示。

### 6.3 用户模块完善

(1) 用户注册时可以增加对密码复杂度的约束。

(2) 用户登录时可以增加验证码约束。

(3) 用户浏览电影时，出现电影无版权情况页面无法正常显示，给用户反馈通道，通知管理员加以操作。

(4) 增加用户互动、评论热度等拓展功能。

### 6.4 其他

界面美化、兼容性测试、移动端开发等