

Cuckoo 电影数据可视化系统说明文档

一、 文档介绍

此文档主要对第九组的软件工程项目——Cuckoo 电影数据可视化系统的功能和细节做一些必要性说明。主要从代码逻辑、实现思路、网页具体展示效果几个方面来介绍我们的项目功能，便于审核者理解评分。

二、 项目网页使用说明

1、 主页（/cuckoo/index/）

（1） 导航栏

导航栏位于各个页面(除登录、注册外)的顶部，且导航栏的位置不会随着页面滚动而改变，用户可以随时使用导航栏的链接跳转各个页面，方便快捷。

主页的导航栏由三部分组成，包括 logo，搜索按钮以及导航栏主体，分别对应图 2-1-1 以及图 2-2-2 标示的绿色框，黑色框以及橙色框。

导航栏的关闭状态如图 2-1-1 所示。其中绿色框为网页的 logo，点击即可跳转主页，此条在任何界面的导航栏均有效。黑色框为搜索按钮，点击即可弹出搜索栏。橙色框为导航栏打开按钮，点击即可显示导航栏的具体内容。



图 2-0-1-1 主页导航栏(关闭状态)

导航栏的打开状态如图 2-2-2 所示。其中黑色框为导航栏关闭按钮，点击即可关闭导航栏的具体内容。另外，在橙色框展示的是即为导航栏的具体内容。不同页面的导航栏具体内容并不完全一致。可以看到，在主页的导航栏中主要包括“HOME”、“SERVICES”以及“LOG IN”三个栏目。下图中“LOG IN”一栏显示“TYJ”，是因为此时用户已经登录，显示的为登录用户的用户名，若用户未登录即显示“LOG IN”。



图 2-1-2 主页导航栏(打开状态)

其中“HOME”一栏的详细内容如图 2-1-3 所示,当鼠标移动到“HOME”上时,即可看到下图所示的内容,鼠标移开则收起详细内容。对于其他两栏,展示与收起的方式一致。

对于“HOME”栏,点击“HOME”即可跳转主页,这个主页的模式与当前访问的页面模式一致。也就是说,如果当前处于白天模式,则跳转的主页也为白天模式。点击“HOME DEFAULT VERSION”即可跳转白天模式的主页,同样的点击“HOME DARK VERSION”即可跳转黑夜模式的主页。

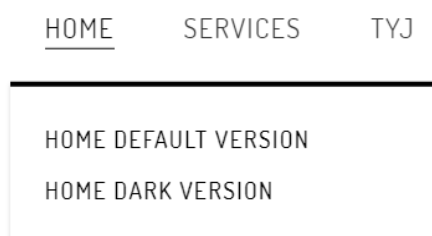


图 2-1-3 “HOME”详细内容

对于“SERVICES”栏,其详细内容包括可以跳转电影榜单及排序页面的“MOVIE TABLE”、电影趋势页面的“MOVIE TREND”、劳模演员页面的“MODEL ACTORS”、票房数据页面的“BOX OFFICE SHARE”。

对于“LOG IN”栏,若用户未登录,则其详细内容为可以跳转登录界面的“LOG IN”。若用户已经登录,则在用户名下的详细内容为“LOG OUT”,点击则会弹出提示窗口,提示“Are you sure to log out?”,此时点击确定则注销登录,点击取消则取消注销操作。

(2) 搜索栏

点击图 2-1-1 黑色框中的搜索按钮即可打开如图 2-1-4 所示的搜索栏。其中,白色框为关闭按钮,点击即可关闭搜索框。橙色框为搜索栏主体,可以在橙色框内键入想要搜索的文字。绿色框为操作提示,按下“Enter”键即可跳转搜索详情页,显示搜索的结果,按下“ESC”键即可关闭搜索框。蓝色框为搜索提示,Cuckoo 可以支持对电影名称、演员以及导演的搜索。

搜索功能只对登录用户开放,若用户未登录则跳转登录界面。

(3) 主体内容

主页的如图 2-1-5 所示，包括 Cuckoo 能够提供的四个功能。与导航栏的“SERVICE”相似，点击即可跳转相应的可视化界面。



图 2-1-4 主页搜索栏

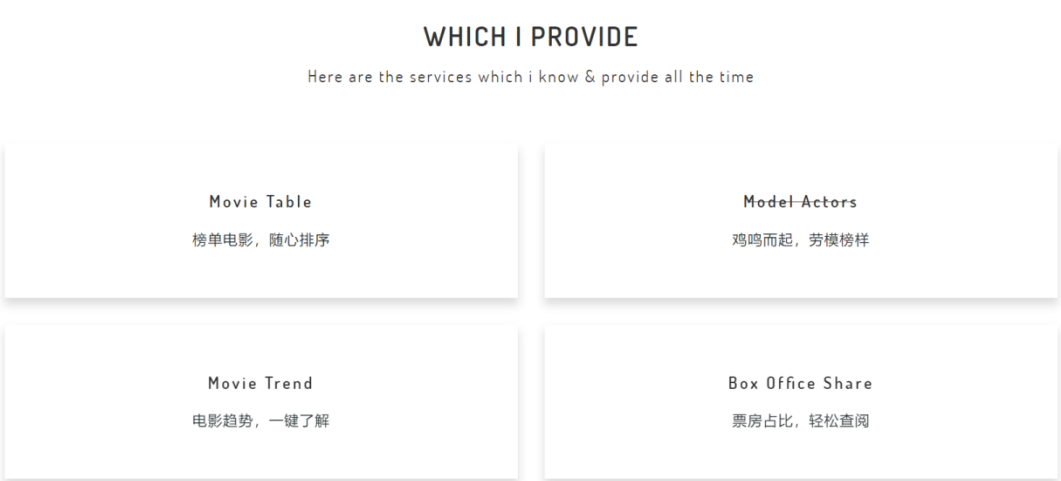


图 2-1-5 主页主体内容

2、搜索页面

(1) 页面主体

搜索“刘昊然”的结果如图 2-2-1 所示。搜索页面主要由导航栏、搜索结果以及侧边栏组成。其中导航栏与主页的导航栏完全一致。

搜索页面单个搜索结果如图 2-2-1 中绿色框所示，包括图片、电影名称、电影评分、参演人员以及收藏按钮。点击图片即可跳转猫眼电影网站对应电影的详情界面。

图 2-2-1 中橙色框为收藏按钮，若该电影已被用户收藏，则显示“已收藏”，点击则取消收藏。若该电影未被用户收藏，则显示“收藏”，点击后该电影被收录到用户的收藏夹中。



图 2-2-1 搜索页面主体

(2) 侧边栏

搜索页面的侧边栏如图 2-2-2 所示。侧边栏由搜索框及用户收藏夹组成。其中橙色框为搜索框，显示现在展示的搜索结果的输入，在搜索框内键入新的输入文本，点击绿色框内的搜索按钮即可搜索新的内容。

黑色框为用户收藏夹，未登录用户的收藏夹内无任何内容。对登录的用户，所有收藏的电影将在收藏夹内显示。灰色框为收藏夹中的一个内容，包括图片及电影名称，点击图片或电影名称即可跳转猫眼电影网站对应的电影详情页面。



图 2-2-2 搜索页面侧边栏

3、数据分析页面

(1) 页面主体

电影排名页面主体如图 3-1-1 所示，劳模演员页面与其相似，都是可根据用户输入的标签生成按电影票房排序的一个表格，其中橙色框即为得到的表格信息。淡绿框为可多选标签，点击标签会出现下拉框，可以选择多个标签，若不选则默认该选项所有标签都参与排序。墨绿框为单选标签，同样会有下拉框，只可选择一项，若不选默认为第一项。

选择完毕后点击蓝色框中的 Sort 字样即开始进行分析，将在下方表格中生成相应的数据。同时网页会跳出一个弹窗（注意：这个弹窗可能会被浏览器拦截），新窗口中包含这张表格的数据可视化柱状图。



图 2-3-1 数据分析页面主体

电影票房趋势分析与类型电影票房占比两种分析功能的页面略有不同,它们没有表格和弹窗,而是在页面中直接以可视化图表展示分析结果。其中票房趋势以折线图展示,票房占比以柱状图和饼状图展示。

如图 2-3-2 为折线图,点击上方的标签可以即时删除某一条数据,鼠标移至某一点将会出现该点的具体数据信息,饼状图类似。

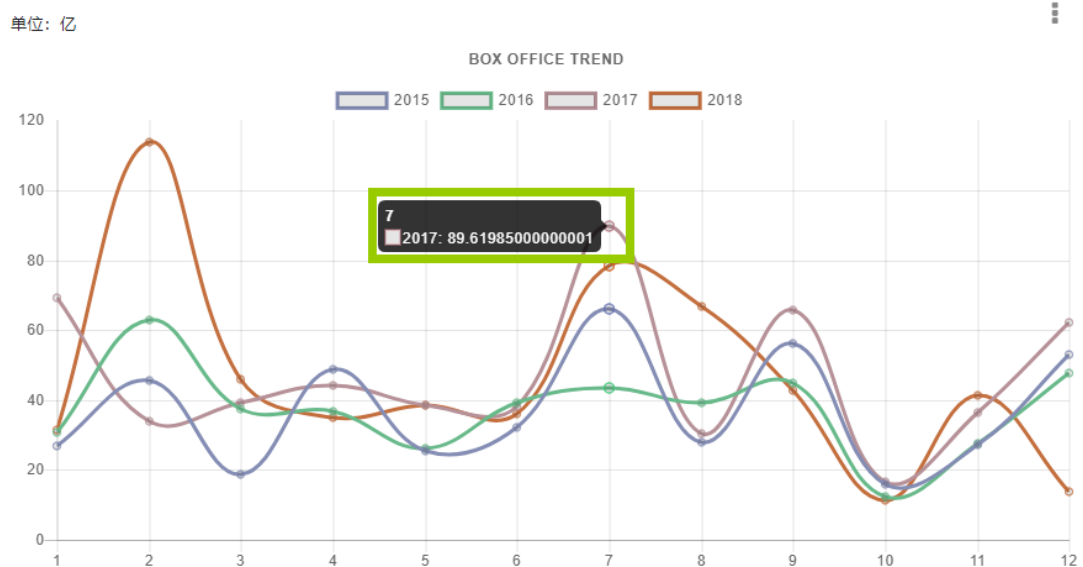


图 2-3-2 票房趋势分析折线图

(2) MovieTop 与 ActorTop 的弹出窗口

如图 2-3-3 是弹出窗口的部分按钮和柱状图,该页面同样包含导航栏与

侧边栏，这里不再赘述。其中柱状图可与用户动态交互，当鼠标移至某一列上方时，该列会出现暗色以示选中，和蓝色框所示的提示语，且该提示框会随用户鼠标移动。

点击橙色框的 Save All 和黄色框的 Download Image 都可以将该柱状图下载为图片存储，浏览器会出现一个提示框，用户可以自定义图片名称。其中黄色框中的按钮需要点击绿色框才会出现，鼠标再次点击空白处便会缩回隐藏。

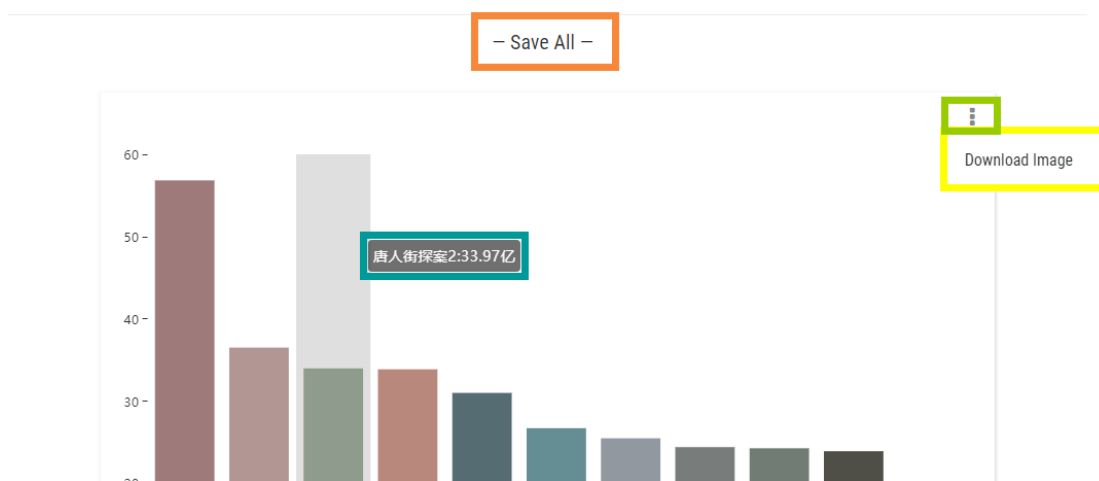


图 2-3-3 弹出口口部分画面

(3) 侧边栏

如图 2-3-4 所示为数据分析页面的侧边栏，其中左侧为关闭时画面，右侧为选项卡都打开时的样子。其中“tyj”为当前登录用户的用户名，若未登录会显示空。点击 Home 键可返回主页；点击 Services 会出现 Cuckoo 为用户提供的数据分析列表，即右图灰色框所示除当前页面提供的功能之外的其他三个功能；Favorite 为用户收藏夹，打开会出现用户已收藏的电影名称；History 为用户下载过的图表历史，打开会出现那些图表的缩略图，点击图片会重新下载该图，点击垃圾桶按钮可将该图片从历史记录中删除。

其中除 Home 与 Services 可不登录使用之外，其他功能都需要登录后才可正常使用，否则即使点击也不会出现正常画面。

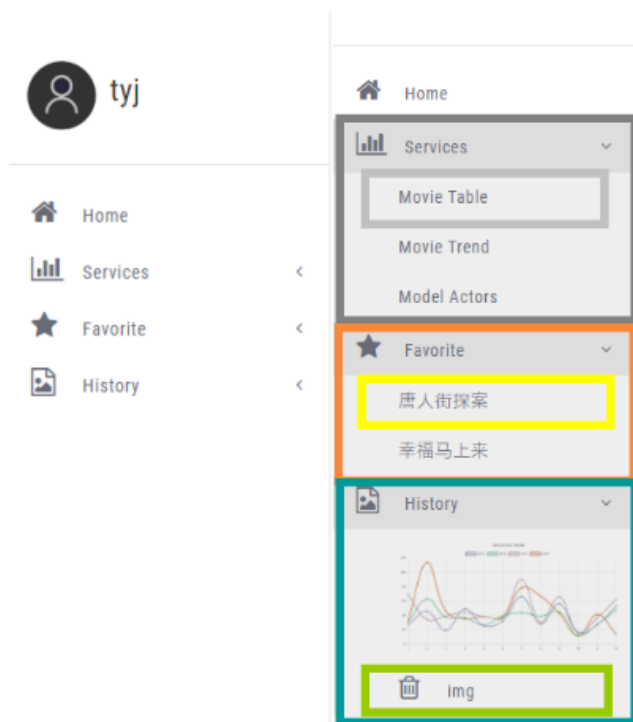


图 2-3-4 侧边栏

(4) 导航栏

数据分析页面的导航栏与主页导航栏基本相同，但新增了一个 EYE 按钮，鼠标移至其上方会出现一个下拉框，其中有两个选项“DEFAULT”、“DARK”，可以切换日夜间模式，点击后页面不会跳转。

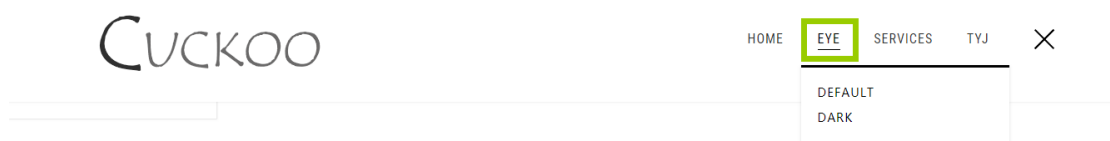


图 2-3-5 导航栏

4、登录注册页面

(1) 登录页面

从任一页面的导航栏中点击 LOGIN 按钮进入登录界面，登录页面主体如图 2-4-1 所示，绿色框中为输入框，可根据提示输入用户名与密码，其中密码为非明文展示。点击橙色框中的 LOGIN 按钮即可向后台发送请求，后台收到信息后会根据是否成功登录给出相应提示语。点击蓝色框中的 HOME 可放弃登录返回主页。若无账号，可点击黄色框中的字样跳转到注册界面，若忘记密码

码，可点击灰色框中字样跳转到相应页面找回密码。

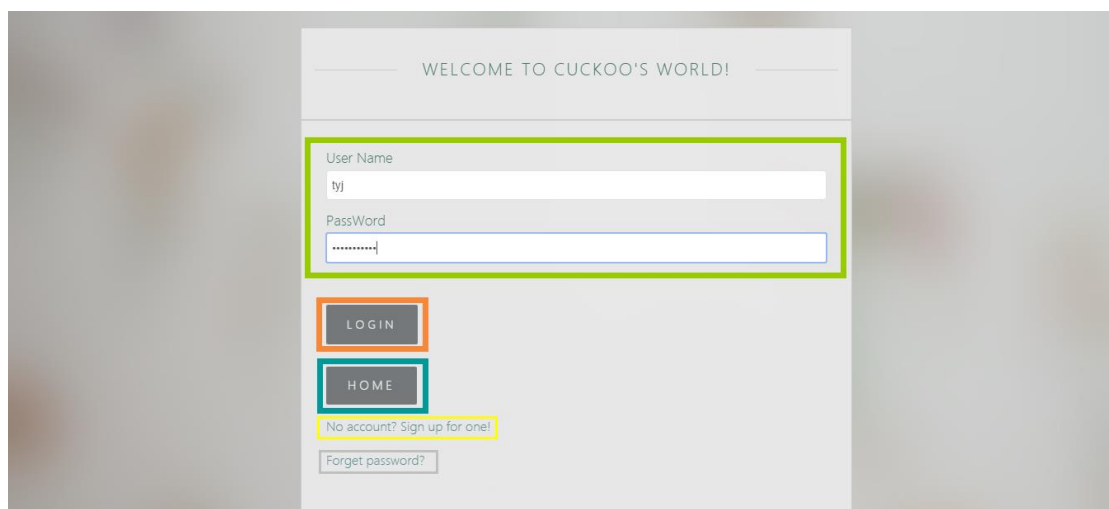


图 2-4-1 登录页面

其中，用户名与密码不可为空，且只能由数字、大小写字母和下划线组成，长度有一定限制，若其格式不符合要求或者没有必填项未空就按下了 LOGIN 试图登录，则无法发送请求，页面会提示输入有误，如图 2-4-2 为提示页面，点击页面中的 CLOSE 可返回表单重新填写，此时输入有误的框会跳动标红。

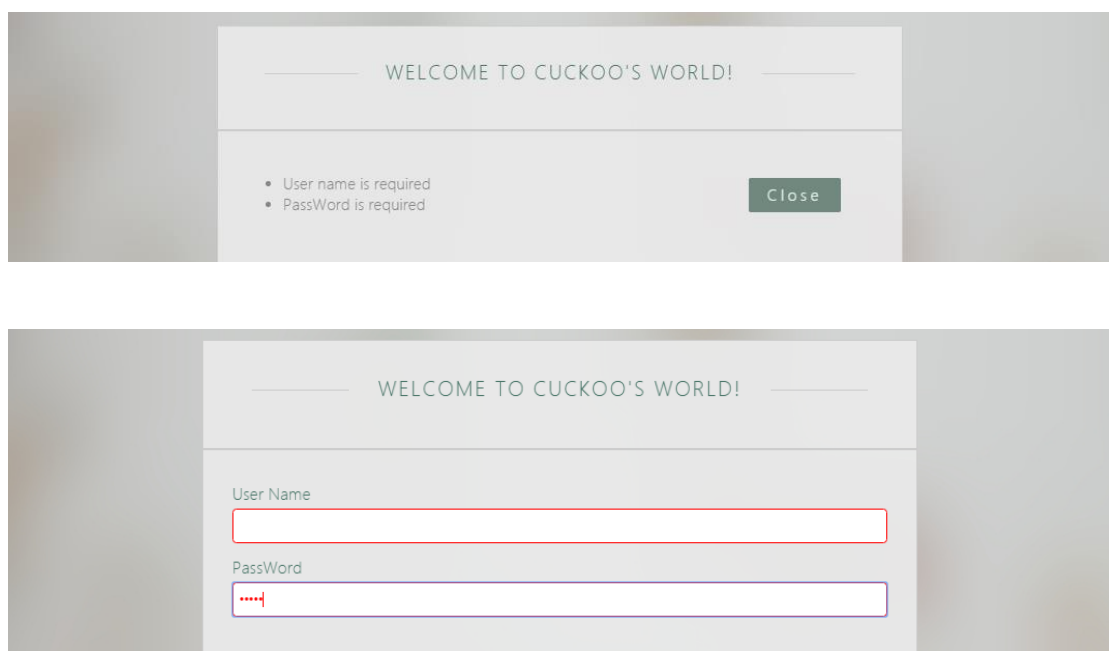
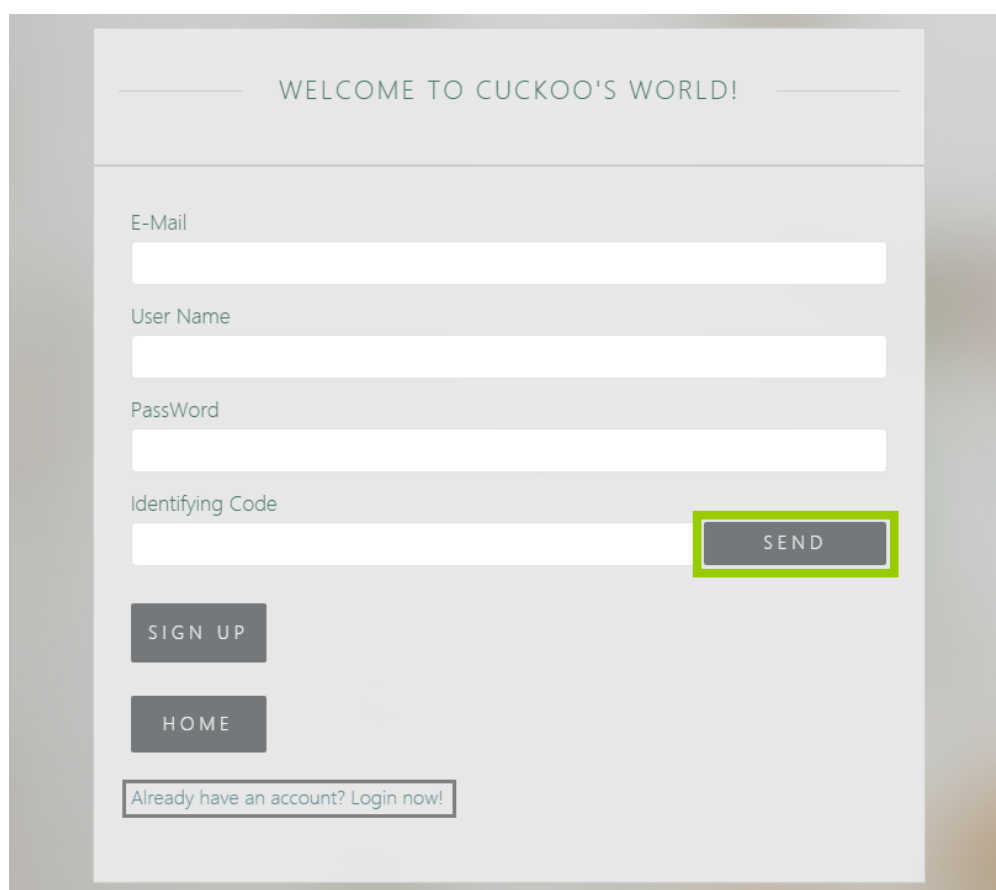


图 2-4-2 登录页面错误提示

(2) 注册页面

注册页面与登录页面基本相同，只是需要输入的信息增加了邮箱和验证码。用户需要填写正确格式的邮箱之后，点击绿色框中的 SEND 按钮，服务器会向用户填写的邮箱发送验证码（若未收到可以查看一下邮箱的垃圾桶，也许会被邮箱归为垃圾邮件），收到验证码后将其填写到输入框，所有信息正确填写后点击 SIGN UP 即可注册账号。

若已经有一个可用账号，则可以点击下方的灰色框中字样进入登录页面。

The image shows a registration form titled "WELCOME TO CUCKOO'S WORLD!". It contains four input fields: "E-Mail", "User Name", "PassWord", and "Identifying Code". A green rectangular box highlights the "SEND" button located to the right of the "Identifying Code" field. Below the input fields are two buttons: "SIGN UP" and "HOME". At the bottom, there is a link that says "Already have an account? Login now!".

WELCOME TO CUCKOO'S WORLD!

E-Mail

User Name

PassWord

Identifying Code

SEND

SIGN UP

HOME

Already have an account? Login now!

图 2-4-3 注册页面主体

(3) 找回密码页面

找回密码页面与注册页面十分类似，填写邮箱发送验证码后填写验证码，用户名和新密码，点击 SUBMIT 提交到后台，若信息都正确，则可成功修改密码，若信息有误，页面会给出相应的提示。

如果又突然想起密码了，可以点击下方灰色框中的字样跳转到登录页面。

WELCOME TO CUCKOO'S WORLD!

E-Mail

User Name

Identifying Code

SEND

NEW PassWord

SUBMIT

HOME

Something suddenly come to you?

图 2-4-4 找回密码页面主体

三、 主要功能说明

1、 系统账户登录功能

登录、注册、找回密码的功能实现都是类似的，在此以登录为例，对其实现原理，程序逻辑进行简要的解释。

用户在前端填写完数据后，点击 LOGIN 按钮即可提交表单，此时将会进入 jquery.fform.js 文件中进行填写数据规范性判断。在 login.html 文件中，这一段代码定义了每个输入框的输入要求。

```
<script type="text/javascript">
$(document).ready(function () {
    $('#form').fform({ animation: 'flip', submitButton: '#submit',
        validationIndicator: '#validation', errorIndicator: '#error',
        successIndicator: '#success',
        'fields':
            [{ 'id': 'username', required: true, requiredMsg: 'User name is required',
                type: 'alpha', validate: true, msg: 'User name can only consist of letters, numbers and _'},
              { 'id': 'password', required: true, requiredMsg: 'PassWord is required',
                type: 'alpha', validate: true, msg: 'Password can only consist of letters, numbers and _'},
            ] });
});
</script>
```

js 文件中的 init 函数除了设定了页面的动态效果之外，还设定了两个按钮的触发函数，分别是发送邮件的 SEND 按钮和提交表单的 SUBMIT 按钮，对于登录，则只会触发名为 LOGIN 的 SUBMIT 按钮。

```
$(settings.submitButton).click(function (e) {  
    successIndicator.css({ display: 'none' });  
    errorIndicator.css({ display: 'none' });  
    validationIndicator.css({ display: 'none' });  
  
    e.preventDefault();  
    e.stopImmediatePropagation();  
    validationResult = '<ul id="validation-list">';  
    isValid = true;  
    isEmailValid = true;  
    isUserNameValid = true;  
    validate();  
    if (!isValid || !isUserNameValid || !isEmailValid) {...}  
    else {  
        //add animate classe  
        let w = parseFloat($(this).outerWidth());  
        $(this).addClass('animate');  
        $(this).data('text', $(this).val());  
        $(this).val(settings.loadingText);  
        try_login();  
        return false;  
    }  
});
```

该文件下方有定义

函数中（在 170 行左右）首先初始化了输入可行标签，然后进行可行性判断（validate 函数，下有定义），若通过，则尝试发送登录 Ajax 请求，方式为 POST，携带的数据有用户名和密码，django 通过对请求来源 url 的分析，从 url.py 文件中找到了对应的视图函数，后台处理完后根据登录状态返回相应的提示语；若未通过则定义了一些提示语和提示动态效果。

```
path('login/<int:bg_color>', cuckoo_views.login, name='login'), # 登录页面  
path('register/<int:bg_color>', cuckoo_views.register, name='register'), # 注册页面  
path('forget/<int:bg_color>', cuckoo_views.forget, name='forget'), # 找回密码页面
```

```
# 处理登录，需要用户名存在且与密码匹配  
@csrf_exempt  
def deal_login(request):  
    username = request.POST.get('username')  
    password = request.POST.get('password')  
    user = auth.authenticate(username=username, password=password)  
    if user is not None:  
        auth.login(request, user)  
        request.session['user'] = username  
        return JsonResponse({"message": "Login successfully. Welcome to Cuckoo's world!", "s": True})  
    else:  
        return JsonResponse({"message": "User name does not exist or password error.", "s": False})
```

2、数据可视化功能

数据可视化主要包含了四个必选功能，其中 top 电影和劳模演员逻辑结构相似，且都有两个额外的可视化图表弹窗；票房份额和票房趋势直接生成可视化图表，

两者也是类似的，下面分别以 top 电影和票房趋势为例，对数据可视化功能做一个说明。下图是这 6 个页面的 url 及对应的视图函数。

```
path('table/<int:bg_color>', cuckoo_views.movie_table, name='table'), # 电影列表页面
path('actors/<int:bg_color>', cuckoo_views.model_actors, name='actors'), # 劳模演员
path('share/<int:bg_color>', cuckoo_views.box_office_share, name='share'), # 总票房分析
path('trend/<int:bg_color>', cuckoo_views.movie_trend, name='trend'), # 票房趋势变化
path('movie-top/<int:bg_color>', cuckoo_views.movie_top, name='movie-top'), # top电影可视化页面
path('actor-top/<int:bg_color>', cuckoo_views.actor_top, name='actor-top'), # 劳模演员可视化页面
```

(1) Top 电影

用户在选择框中做好选择后，点击右侧的 Sort 按钮即可提交生成表格，且会弹出一个窗口。对于 Sort 按钮绑定了一个点击触发事件，点击按钮就发送请求携带数据至后台。值得一提的是，我们的所有数据分析都有可支持多选的选择框，因此为了便于后台函数分离各个标签，我们在每一个选项中增加了一个‘-’。（下图写在对应的 html 文件中）

```
341 </script>
342 <script type="text/javascript">
343     $(document).ready(function () {
344         $('#Sort').click(function () {
345             $.ajax({
346                 url:'deal-movie-top',
347                 type:'GET',
348                 async:true,
349                 data:{
350                     month:$("#month option:selected").text(),
351                     season:$("#season option:selected").text(),
352                     area: $("#movieaddr option:selected").text(),
353                     year: $("#year option:selected").text(),
354                     movietop: $("#movietop option:selected").text(),
355                     movietype: $("#movietype option:selected").text()
356                 },
```

django 经过处理后找到对应视图函数，在视图函数中拿到请求里的数据，调用分析库相应函数（top_movie）进行分析处理，得到分析结果的数据 list。

```

# 处理具体数据分析请求
def deal_movie_top(request):
    year = request.GET['year']
    month = request.GET['month']
    area = request.GET['area']
    movietop = request.GET['movietop']
    movietype = request.GET['movietype']
    season = request.GET['season']

    print(year+season+month+area+movietype+movietop)
    top_movie_result = analyse.top_movie(area, movietype, year, season, month, movietop)
    print(top_movie_result)
    # label box 为了便于生成图表，将需要的数据在后台提前提取
    label = []
    box = []
    for movie in top_movie_result:
        label.append(movie[3])
        box.append(movie[1])
    box.append(0)
    return JsonResponse({"item_set": top_movie_result, 'label': label, 'box': box})

```

前端网页得到后台回复的响应 (JsonResponse) 后，对成功与失败两种结果分别处理。若成功，则对于 top_movie_result (在前端的变量名为 item_set) 的每一条数据，动态生成表格中的一行，打开新可视化弹窗。为了实现 label 和 box 的数据跨网页传输，我们在现网页中设定了两个隐藏标签用来暂存数据，然后在弹窗网页 (movie-top.html) 中通过 window.opener 取出相应数据，随后调用 echarts，传入得到的数据画出图表。

(2) 票房趋势

此功能与前所说的 top 电影大同小异，唯一的差别在请求发送成功后的处理上。这里并没有生成表格，也没有弹窗，是直接生成图表，需要点明的是，这里的折线图（和票房分析的饼图）我们并没有使用 echarts 库，而是直接调用的 charts，用 canvas 画图。Canvas 存在一个问题，每次画图是并不会清空上一张图片，而是直接在其基础上再添加，如果不做处理就会出现图表重叠的 bug。之前尝试了好几种方法都没有效果，因此我们在生成新图表前直接将原画布删除，新增了一个画布。

```

284 <script type="text/javascript">
285     $('#Sort').click(function () {
286         $.ajax({
287             type: 'GET',
288             data: {
289                 movieaddr: $('#movieaddr option:selected').text(),
290                 movietype: $('#movietype option:selected').text()
291             },
292             url: 'deal-trend',
293             success: function (result) {
294                 $('#line-chart').remove(); // 移除旧画布
295                 $('#div-line').append('<canvas id="line-chart" width="800" height="400"></canvas>'); // 添加新画布
296                 new Chart(document.getElementById("line-chart"), {
297                     type: 'line',
298                     data: {
299                         labels: [1,2,3,4,5,6,7,8,9,10,11,12],
300                         datasets: [{
301                             data: result.five,
302

```

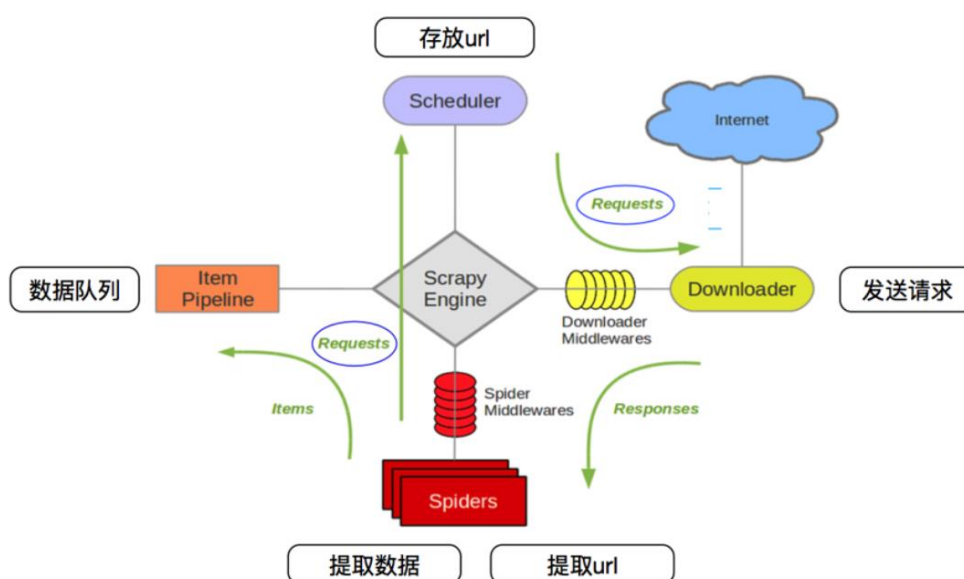
本来我们的柱状图也是直接使用 canvas 画得的，但是当标签过多时，canvas 会自动隐藏一半的标签，网页动态查看还好，有互动，但是当用户需要

下载图表时，隐藏的 x 轴标签就再也无法查看了，这是肯定不行的。但是我们找了很久也没有找到解决办法，最后直接抛弃了 charts，换用提供了标签间接口的 echarts 了，并且将标签设定为竖向展示，解决了标签过长导致的重叠问题。

3、爬虫

框架：scrapy+selenium

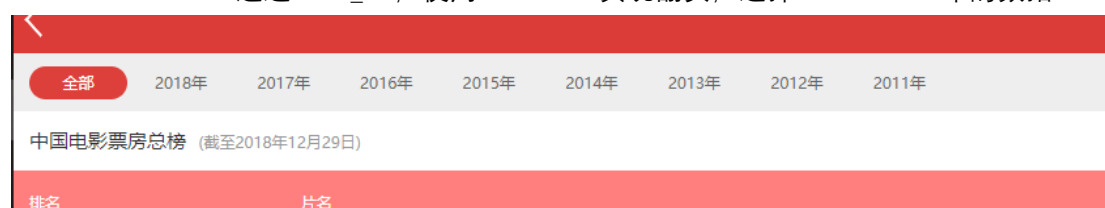
Scrapy 的框架如下：



域名：<http://piaofang.maoyan.com/>

实现方法：

- Start_url: <http://piaofang.maoyan.com/rankings/year>
- 通过 start_url，使用 selenium 实现翻页，选择 2015-2018 年的数据



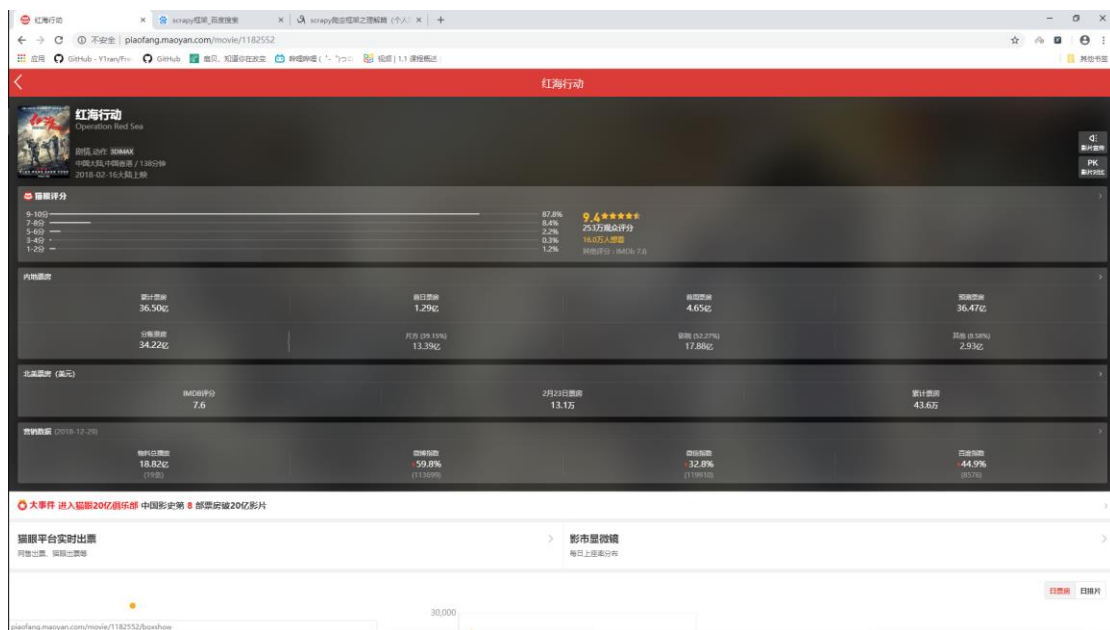
这部分实现的代码在 downloader middlewares 中实现

```

class seleniumMiddleware(object):
    #piaofang.maoyan.com/rankings/year需要使用selenium模拟点击年份使其response对应电影列表
    click_page_url = 'http://piaofang.maoyan.com/rankings/year'
    def process_request(self, request, spider):
        #验证爬虫
        if spider.name == 'maoyan_spider':
            spider.browser.get(request.url)
            # 验证点击页
            if request.url == self.click_page_url:
                print('aaaaaaaaaaaaaaaaaaaaa')
                #spider.browser.find_element_by_xpath("//*[@id='tab-year']/ul/li[2]").click()
                time.sleep(3)
            else:
                try:
                    spider.browser.execute_script('window.scrollTo(0, document.body.scrollHeight)')
                except TimeoutException as e:
                    print('超时')
                    spider.browser.execute_script('window.stop()')
                time.sleep(2)
            return HtmlResponse(url=spider.browser.current_url, body=spider.browser.page_source, encoding='utf-8', request=request)

```

- 通过获取主页面每个电影条目后的 url，将其送到 request 队列之中。



在电影详细界面，获取所需信息。这里使用了正则+xpath 的方法。具体条目如下：

```

class WebscrapyItem(scrapy.Item):
    # define the fields for your item here like:
    # name = scrapy.Field()
    #电影中文名
    movie_name_chi = scrapy.Field()
    #电影英文名
    movie_name_eng = scrapy.Field()
    #电影索引号
    movie_url_index = scrapy.Field()
    #评分
    movie_score = scrapy.Field()
    #导演
    movie_directors = scrapy.Field()
    #主演
    movie_stars = scrapy.Field()
    #类型
    movie_type = scrapy.Field()
    #出品国家
    make_in = scrapy.Field()
    #票房
    movie_boxOffice = scrapy.Field()
    #电影时长
    movie_timeLength = scrapy.Field()
    #上映时间
    movie_releaseDate = scrapy.Field()
    #票房单位
    movie_boxOffice_unit = scrapy.Field()

```


- 同时在电影详细界面，获取每个演员的详细页 url，用于获取其性别。这部分不需要每次爬虫启动都爬，所以我注释掉了，如果有需要把注释给关掉即可。

Maoyan_spider.py

```
#movie_item['movie_releaseDate'] = response.xpath('//html/body/div[2]/section[1]/div[1]/div[3]/div[1]/div[2]/div[2]/div[1]
yield movie_item
'''

stars_list = response.xpath('//html/body/div[2]/section[2]/div/div/div[1]/div[1]/div[2]/div[2]/a/@href').extract()
for star in stars_list:
    print(star)
    #yield scrapy.Request('http://piaofang.maoyan.com' + star, callback=self.actor_detail_parse)
'''
pass
```

- 同时，为了避免反爬虫机制的反爬，使用了 user_agent 及多 ip 策略，对每一条 url 实现随机选择一个 user_agent 及 ip。这部分代码在 download middlewares 中。此处 ip 为静态添加，若有 api 接口可以简单修改一下代码。

```
class my_proxy(object):
    def random_ip_choose(self):
        ip_list = [
            '119.62.192.237:8060',
            '124.89.162.139:8060',
            '120.76.77.152:9999',
            '124.225.144.224:8060',
            '222.135.77.105:8060',
            '117.158.174.164:8060',
        ]
        ip = random.choice(ip_list)
        return ip

    def process_request(self, request, spider):
        ip = self.random_ip_choose()
        print("Current IP:port is " + ip)
        request.meta['Proxy'] = ip

class my_useragent(object):
    def process_request(self, request, spider):
        USER_AGENT_LIST = [
            'MSIE (MSIE 6.0; X11; Linux; i686) Opera 7.23',
            'Opera/9.20 (Macintosh; Intel Mac OS X; U; en)',
            'Opera/9.0 (Macintosh; PPC Mac OS X; U; en)',
            'iTunes/9.0.3 (Macintosh; U; Intel Mac OS X 10_6_2; en-ca)',
            'Mozilla/4.76 [en_jp] (X11; U; SunOS 5.8 sun4u)',
            'iTunes/4.2 (Macintosh; U; PPC Mac OS X 10.2)',
            'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv:5.0) Gecko/20100101 Firefox/5.0',
            'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv:9.0) Gecko/20100101 Firefox/9.0',
            'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:16.0) Gecko/20120813 Firefox/16.0',
            'Mozilla/4.77 [en] (X11; I; IRIX;64 6.5 IP30)',
            'Mozilla/4.8 [en] (X11; U; SunOS; 5.7 sun4u)'
        ]
        agent = random.choice(USER_AGENT_LIST)
        request.headers['User-Agent'] = agent
```

- 同时为了每次输出 csv 文件中或数据库文件中的 item 顺序一致，重构了 csvitemexporter 方法。

```
class itemCsvExporter(CsvItemExporter):
    def __init__(self, *args, **kwargs):
        #delimiter = settings.get('CSV_DELIMITER','.')

        fields_to_export = settings.get('FIELDS_TO_EXPORT',[])
        if fields_to_export:
            kwargs['fields_to_export'] = fields_to_export

        super(itemCsvExporter, self).__init__(*args, **kwargs)
```

并使用 pipeline 实现。

- Scrapy 底层使用 twisted 框架，twisted 框架是有名的多线程异步框架。默认自带线程数为 16。

```
# Configure maximum concurrent requests performed by Scrapy (default: 16)
#CONCURRENT_REQUESTS = 32
```

- 运行 webscrappy 下的 main.py 文件即可运行爬虫。生成 movie_data.csv（或及 actor_data.csv）文件，替换掉 cuckoo 下的 data 数据即可。

4、数据报表打印

对于报表打印，我们对老师给出的要求做了认真的讨论分析，最后得出的结论是这样多有不便。对于用户而言，他们要得到图表的文件模式，必然是要作他用的，如果都保存在文件中，灵活性大大下降了。用户无法直接将结果插入他自己的 word 文档或者 ppt 中，还需要另外打开文档，截图或者复制等等，总之不太方便。

于是我们对需求做了一些大胆的改动，将文档直接改成图片，而勾选下载至同一文档这一功能，我们通过在数据库中保存用户历史下载记录，并在侧边栏的历史记录中显示图表缩略图来替代了。

对于每一个 save 按钮设定点击触发事件，点击后先取得对应图表，将其转存为图片，请用户命名（默认为 img），得到图片的 url 后向后台发送请求。

```

259 <script type="text/javascript">
260     $("#save").click(function () {
261         console.log("111");
262         let myCanvas = $("#e-bar").find("canvas")[0];
263         var image = myCanvas.toDataURL();
264         <!-- console.log(image); -->
265         var img_name = prompt("请输入文件名", "img");
266         if (img_name == null)
267         {
268             console.log("cancel save");
269         }
270         else {
271             $.ajax({
272                 url: 'deal-img',
273                 type: 'POST',
274                 async: true,
275                 data: {
276                     name: img_name,
277                     imgurl: image,
278                 },
279                 success: function (result) {
280                     console.log(result.name);
281                     let img_set = document.getElementById('pages');
282                     let li = document.createElement("li");
283                     let a1 = document.createElement("a");

```

将图表转图片

用户输入文件名

后台收到数据后，判断是否有用户在线，若无则跳转登录页面，若有就拿到当前用户名，将他要保存的图片 url 存入数据库的 history 中，返回该图片的 url 到前端。前端得到回复后，便在侧边栏生成新的缩略图。

```

# 处理保存图片
@csrf_exempt
def deal_img(request):
    if request.user.is_authenticated:
        user_name = request.user.username
        user = User.objects.get(username=user_name)
        imgurl = request.POST['imgurl']
        imgname = request.POST['name']
        the_new_img = user.history_set.create(image_name=imgname, data_url=imgurl)
        the_new_img.save()
        return JsonResponse({'message': '保存成功', 'name': imgname, 'imgurl': imgurl})
    else:
        return login(request, 0)

```

若在侧边栏点击了删除按钮，就向后台再次发送请求，若成功则移除该图片。

```

317     <script>
318         function delimg(logname) {
319             console.log(logname);
320             $.ajax({
321                 url: 'delimg',
322                 type: 'post',
323                 async: true,
324                 data: {
325                     logname: logname,
326                 },
327                 success: function (result) {
328                     let history_item = document.getElementById(logname);
329                     history_item.parentNode.removeChild(history_item);
330                     console.log(result.message);
331                 },
332                 error: function (result) {
333                     alert('ajax:ERROR');
334                 },
335             });
336         });

```

后台收到请求后，对应处理函数如下，即从用户数据库中将该图片 url 移除。

```

# 处理删除图片
@csrf_exempt
def deal_delimg(request):
    if request.user.is_authenticated:
        user_name = request.user.username
        user = User.objects.get(username=user_name)
        imgname = request.POST['logname']
        the_img = user.history_set.filter(image_name=imgname)
        the_img[0].delete()
        return JsonResponse({'message': '删除成功', 'name': imgname})
    else:
        return login(request, 0)

```

四、感言

本次项目真的可以说是困难重重，我们小组 4 个人之前都完全没有接触过前端，甚至于在开始项目之前我们对前后端的概念都十分模糊一头雾水。项目要求实现的这些功能说到底其实并不难，后台的逻辑我们都可以独立完成。但是这一次是要求有用户界面的，我们写的东西不再是仅供程序员之间交流使用，而是要面对用户的，心里一下子就慌了。

我们四个都可以说是普通人吧，面对从未接触过的领域，没有引路人，四处无光，我们很正常的产生了畏难情绪，又正好有别的事情要忙，于是具体表现就是逃避拖延。自开了一两次小会安排下任务后，没有人再提起项目的事，大家都很有默契。直到项目验收前两周，终于没法再拖了，必须要开始了。其实是真的被每周的实验课验收追着跑，两周没睡几个囫圇觉，全组每天抱着电脑跑到 G 栋持续爆肝到凌晨两三点，通宵也是有过的。

最开始时是真的举步维艰，页面搭建不懂，前后台联系不懂，请求不会发，什么都不会，前端还会产生各种乱七八糟的问题，只好到处找前辈请教，实验室的师兄，学前端的亲戚，

刷 django 文档，学前端菜鸟教程，百度谷歌，一路跌跌撞撞的摸索着走。所以理所当然的，前几个页面的结构做成一团浆糊，毕竟是靠想象力在做。还是找助教帮忙之后才恍然得知，噢！原来是这样写的啊？，在这里实名感谢两位助教！

后来终于慢慢渐入佳境，什么都懂一些了，遇到问题也知道该怎么在百度搜索发问了，这些东西不再是未知的怪物，我们开始了解它了。每跨过一个小小的障碍，都要引起热烈的欢呼，沮丧时大家会互相安慰扶持，快乐时会一起大笑庆祝。熬夜赶项目真的辛苦，但是没有人提前走过，没有人说过放弃，没有人抱怨彼此，这个组真的很好。这是我第一次感受到团结的力量，原来一个组真的不是几个人的离散合成，而真的就是一个全新的集合，是一种完全不同于个人的力量。

本来想在这里一一点名感谢，但是写了好几段还是觉得很不好意思，我很感激这一次能和大家一起完成这个项目。我也很感激能给出这样项目的老师，虽然辛苦，但是真的有学到东西，不仅是这些知识，对我而言，这个学习的过程，和大家合作的经历，第一次完成项目的经验，才是无上的瑰宝。

Cuckoo 还是一个雏儿，我们还有很多想法，我们还可以添加很多功能，但是很可惜时间不够了。Cuckoo 的成长只能暂停在这里，也许未来有机会，我会再次完善它雕琢它，但对于这一次课而言，这就是全部了。我很喜欢 Cuckoo，无论它最后会得到多少分，当然，或许它还有一些缺陷，但希望你们也能喜欢它。

——唐颖嘉

唐颖嘉说得对。

我们组能实现真的太不容易了。

——梁俊平