

Code Quality Review report

IFoundMovie

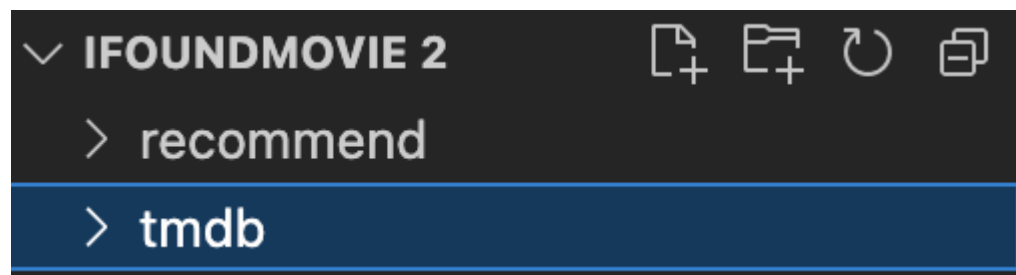
Xufeng Liu, Yixu Zhou

1. Introduction

The running of IFoundMovie is mainly divided into the front end and back end. The front end includes static UI templates and various interactive interfaces. The back end contains the database and API interface. The front end mainly uses HTML, CSS and PHP language, and the back end mainly uses Java and Python. This report carries out a quality assessment from the following aspects.

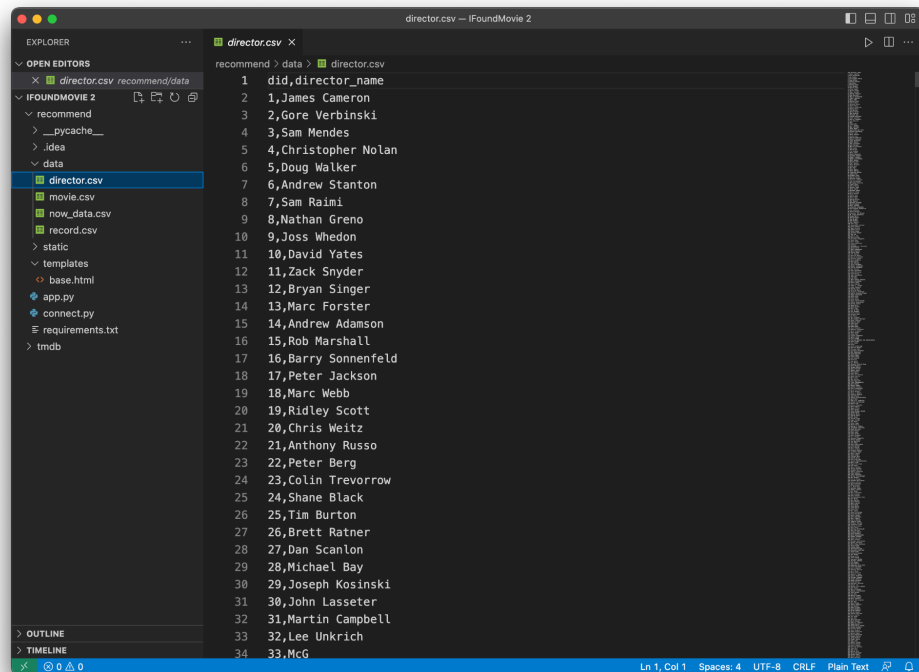
2. Specific Sections

2.1. Integral part



All of our code is divided into two sections: recommend, which mainly stores the AI algorithm section and various data sets, and tmdb, which mainly processes the data obtained from tmdb.

2.2. Data Set Part



In recommend/data, you can see that we have four CSV-type data sets that store director data, movie data, id data, match value data, and converted value.

This approach combined with the code to our background data more convenient and quick management.

2.3. Code Style

```
303     data = pd.DataFrame(nowData)
304     pd.DataFrame(nowData).to_csv('./data/movie.csv', index=False, mode='a+', head
305
306     if bl_record==False:
307         count=1
308         nowData = ['']
309         nowData[0] =(did,mid,count)
310         data = pd.DataFrame(nowData)
311         pd.DataFrame(nowData).to_csv('./data/record.csv', index=False, mode='a+', head
312     else:
313         recordList[now_index]=[did,mid,count]
314         data = pd.DataFrame(data=recordList)
315         pd.DataFrame(data).to_csv('./data/record.csv', index=False)
316     return did
317
318     # Get the most similar director information
319     def topn_simusers(did, n,pd_users):
320         users = pd_users.loc[did,:].sort_values(ascending = False)
321         topn_users = users.iloc[:n,]
322         topn_users = topn_users.rename('score')
323         #print("Similar actor as actor:", did)
324         return topn_users
325
326     # Build matrix
327     def createDataFrame():
328         path = './data/now_data.csv'
329         ratings = pd.read_csv(path, names=None)
330         rp = ratings.pivot_table(columns='mid', index='did', values='matching')
331         rp = rp.fillna(0)
332         rp_mat = np.matrix(rp)
333         m, n = rp.shape
334         mat_users = np.zeros((m, m)) # Two dimensional array, put the small one here
335         # Movie director similarity matrix (1- corrected cosine similarity = corrected co
336         for i in range(m):
```

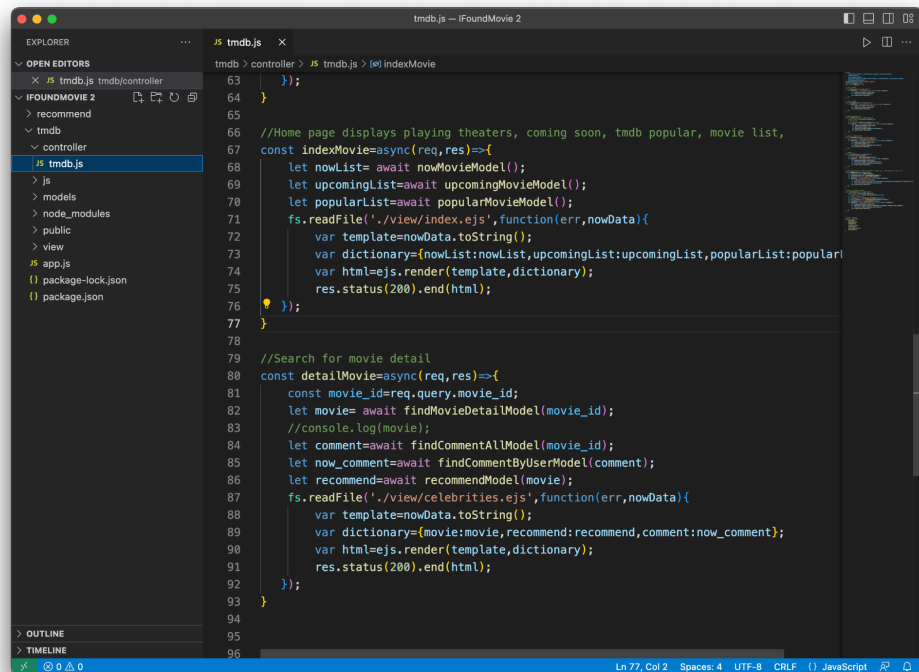
In our main code, we adopt the way of rich comments, which not only provides a supplementary description for a certain code but also facilitates our future modification and debugging.

2.4. Requirement

```
1  Flask==2.2.2
2  mysql-connector==2.2.9
3  numpy==1.23.4
4  pandas==1.4.4
5  PyJWT==2.6.0
6  requests==2.28.1
7  scipy==1.9.1
8
```

We also generate the requirement.txt to explain the version requirements for some of the necessary plug-ins to run.

2.5. TMDB Controller

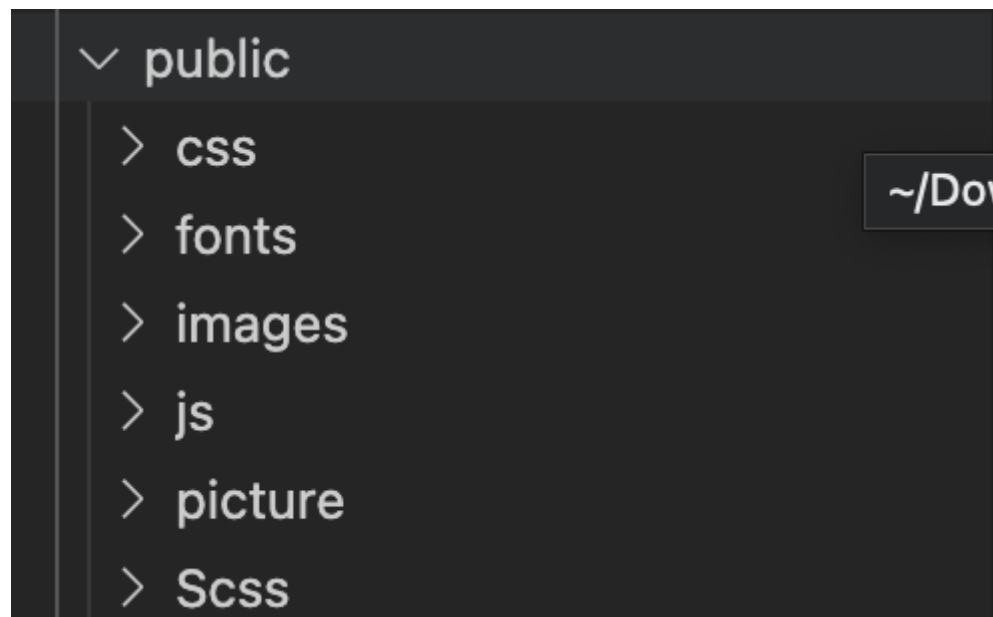


The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like 'tmdb', 'controller', 'models', 'node_modules', 'public', 'view', and files like 'app.js', 'package-lock.json', and 'package.json'. The code editor displays the content of 'tmdb.js', which includes two main functions: 'indexMovie' and 'detailMovie'. The 'indexMovie' function is an asynchronous function that takes 'req' and 'res' as arguments. It calls 'nowMovieModel()', 'upcomingMovieModel()', and 'popularMovieModel()' to fetch data. It then reads a template file 'index.ejs' and renders it with the fetched data. The 'detailMovie' function is also an asynchronous function that takes 'req' and 'res' as arguments. It extracts 'movie_id' from the query and calls 'findMovieDetailModel()', 'findCommentAllModel()', 'findCommentByUserModel()', and 'recommendModel()' to fetch data. It then reads a template file 'celebrities.ejs' and renders it with the fetched data. The status of the response is set to 200 in both functions.

```
63 }
64 }
65
66 //Home page displays playing theaters, coming soon, tmdb popular, movie list,
67 const indexMovie=async(req,res)=>{
68   let nowList= await nowMovieModel();
69   let upcomingList=await upcomingMovieModel();
70   let popularList=await popularMovieModel();
71   fs.readFile('./view/index.ejs',function(err,nowData){
72     var template=nowData.toString();
73     var dictionary={nowList:nowList,upcomingList:upcomingList,popularList:popular
74     var html=ejs.render(template,dictionary);
75     res.status(200).end(html);
76   });
77 }
78
79 //Search for movie detail
80 const detailMovie=async(req,res)=>{
81   const movie_id=req.query.movie_id;
82   let movie= await findMovieDetailModel(movie_id);
83   //console.log(movie);
84   let comment=await findCommentAllModel(movie_id);
85   let now_comment=await findCommentByUserModel(comment);
86   let recommend=await recommendModel(movie);
87   fs.readFile('./view/celebrities.ejs',function(err,nowData){
88     var template=nowData.toString();
89     var dictionary={movie:movie,recommend:recommend,comment:now_comment};
90     var html=ejs.render(template,dictionary);
91     res.status(200).end(html);
92   });
93 }
94
95
96
```

In TMDB.js, we centrally manage the data obtained from tmdb to plan the relevant movie information displayed on each plate of the home page, which is more convenient for the front-end operation.

2.6. UI Design



In the public folder, we manage all the UI design and user interaction elements that are the first things that users come across when they use our site, so we pay a lot of attention to colour matching and styling.

3. Readability

Our code is almost always commented in important places to enhance readability, which makes it easier for us to manage and access, and also brings the viewer more detailed instructions. At the same time, the names of variables and equations are very specific and appropriate, which can be associated with their corresponding meanings and responsibilities at first sight. camelCase is also a standard we follow closely, which makes the code look clean and professional and readable.

