COMP9900 Information Technology Project

Project Report

Term 3 2020

Team chongchongchong

Mentor: Iwan Budiman

Submission Date: 2020-11-16

Name	zID	Role
Yipeng Han	z5192958	Scrum Master/Developer
Mingyan Yu	z5196418	Developer
Ziwei Li	z5187737	Developer
Kaisen Luo	z5185842	Developer

Contents

Project Overview

Background

In recent years, watching movies has become one of the most enjoyable activities for people to release their pressure and share their time with friends and families. When the market of movies is developing rapidly, thousands of new movies are released annually which make it hard for people to find the movies they want most in a short time. Currently, there are some websites related to finding movies and providing the information about the movies at the same time but however, their efficiency and functions are not totally user friendly. Some of their functions are useless for most of the users and slow down the searching period. Therefore, we want to implement a website to help the users easily achieve their goal of finding movies and even sharing their ideas with the other users.

What is FilmFinder?

The FilmFinder is a professional website for searching movies. The users can search the movies they want using the name, genre, description and director of the movie. The searching results will show all the relevant movies and their rates. The users can also search for the detailed information of a movie including the name, released time, director, description, newest rates, and the comments from the other users. Besides, the users can add their favourite movies to their wishlist for reference and manage them at any time. Meanwhile, the users can rate and comment a movie based on their own idea and share it with the other users. The system will recommend movies to the users based on their viewing and rating history.

This report will describe the implementation of FilmFinder. The whole implementation process follows the sprints in Jira and all the user stories supposed in the submitted project proposal are achieved. The outcome of each implementation process has been uploaded to GitHub and demoed twice to the course supervisor in arranged time.

During the implementation, our group members use GitHub and WeChat to share the files and write weekly working diary. We use WeChat group and Zoom conference to discuss the

problems and ideas happened in the implementation and complete the retrospectives after each demonstration.

In the following part of this report, we will show the features, architecture, and highlights of FilmFinder as well as the challenges we faced during the implementation. Before the end of this report, we will provide the documents and guidance about how to install and operate the FilmFinder.

System Architecture
System Design

The FilmFinder system consists of three layers, namely the front-end, the back-end and the database. The front-end is equivalent to the face of the system. For users, the front-end is where users and administrators interact directly with the program. The back-end is the main logic module for program operation, mainly processing front-end interactive information, and direct interaction with the database. After getting the front-end request, the back-end can read data from the database and return to the front-end for display. The database is mainly used to store all information related to the program.

In this project, we use python's flask framework for back-end development and use SQLAlchemy to separate the back-end from the database. SQLAlchemy is an object-relational mapping library. In the database, we chose the powerful relational database MySQL. For the front end, we used the Jinja module of python to render the basic HTML, CSS, JavaScript and bootstrap framework.
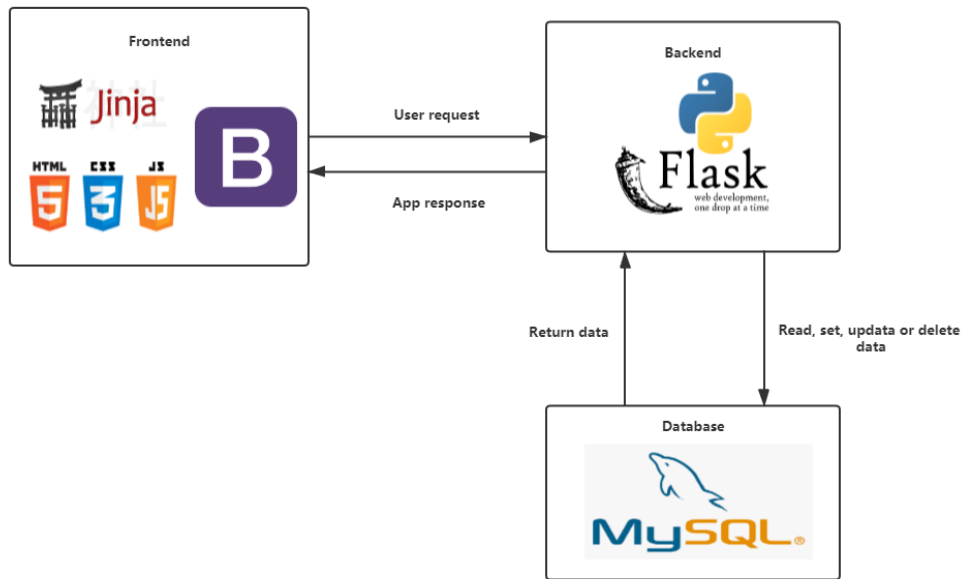
Data View

We chose MySQL as the database of our system. MySQL is a relational database. It has a strong versatility, which is conducive to transplantation on different operating systems.

Below is the ER diagram of our database:
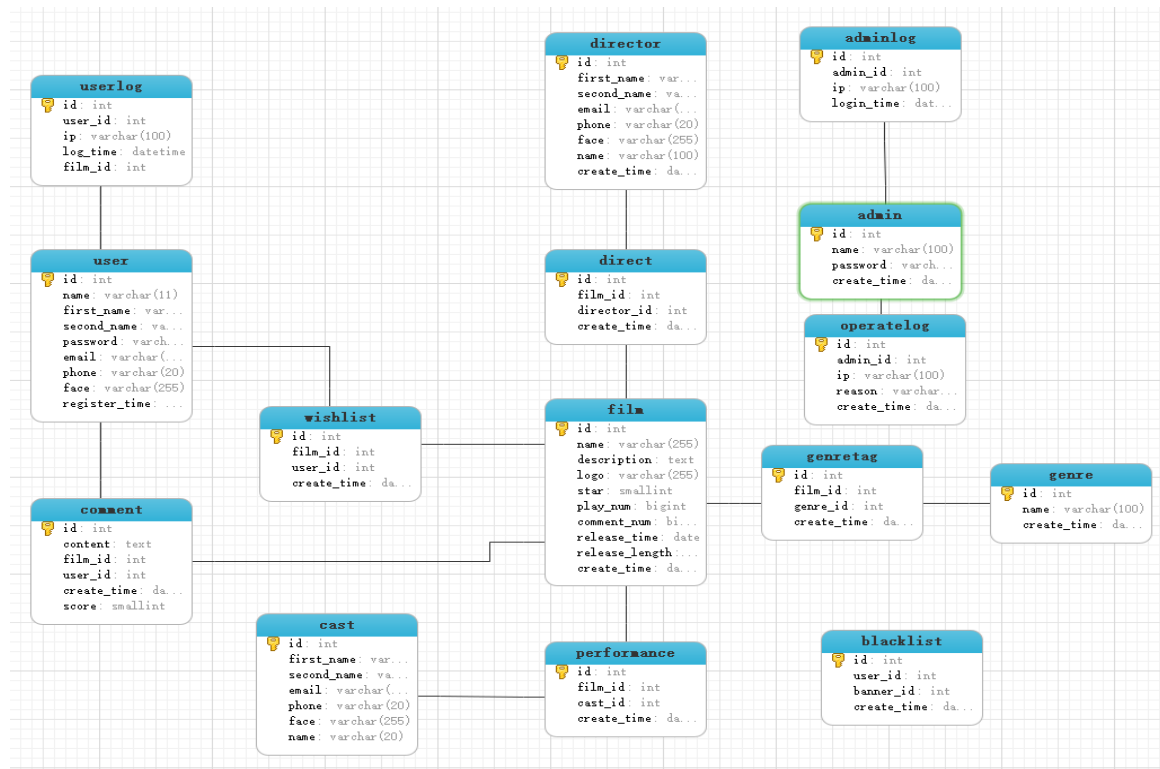
Key Feature Highlights

Information retrieval and restoration

FilmFinder can help the users search the movies via different kinds of information including movie name, description, genre, and director. After searching, the results will show all relevant movies and its detailed information including name, description, genre, director, average rates, and reviews. This feature helps the users achieve all the information they want by just input a single relevant word and click searching button.

Recommendation system

FilmFinder provides a recommendation system to its users. This feature can recommend relevant movies to the users based on the users' searching and viewing history. It will help the users easily achieve their favourite kind of movies and make the whole website much more personalized.

Data maintaining and operating

The database of FilmFinder not just can store the information of plenty of movies, it can also store and keep all the users' information and preference. The user can modify their own information like their own username or login password in the account page. For the movies the user likes, he or she can add them into their own wishlist and remove them when they change their mind. All the operations above can be achieved once the user clicks without any delay.

Information sharing and blacklist

The users in FilmFinder can rate and comment movies according to their own ideas. At the same time, users can see the others' rates and comments about the movie and have the access to put the users into their own blacklist if they hold different ideas. Once addressed into the blacklist, that user's rates and comments about all movies will not be seen by the user who holds the blacklist. The rates of each movie will be recounted automatically except the rates by the users in the blacklist. This feature can help to minimize the influence of

malicious users and allow the users to communicate with the people who have similar ideas or interests.

System and Feature Walkthrough

Customer App

Landing Page, Register and Sign in

The FilmFinder program allows users to search for movies on the website, view related information and film reviews of movies in the state of visitor. In addition, other functions require users to log in to use them.

Users can enter nickname, name, email, mobile phone number and other information to register an account. After registering an account, users can change their information in the profile, and upload an avatar to show their personality.

To protect user passwords from potential leaks, the system stores user passwords as a SHA256 hexadecimal digest (64 characters long) instead of a plain text password, it is displayed as the password varchar (64) column in the database system. It ensures that the user's password will not be disclosed even if there is a database leak.

User's profile

After logging in to the website, users can click the user button at the top right to enter their homepage at any time. Here users can change their login name, email address and mobile phone number, as well as upload new avatars, which support jpg format.

User's comment page

The comment button can be found in the navigation bar on the left side of the user homepage. The user clicks to enter the user comment page, where the user can view all the film reviews they have posted, and the user can turn the page to view the film reviews. Users can delete the film reviews they do not like while the scores they gave when they posted this film review are also deleted.
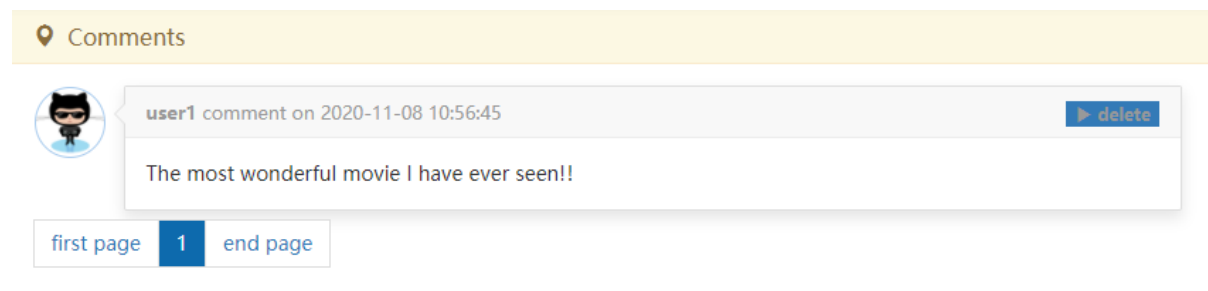

User's wishlist page

Users can click wishlist button which is on the left side of the homepage to enter the user's wishlist interface. Users can view all the movies they put on wishlist here. Users can click on film detail on the right side of the movie to enter the detailed information page of the movie. Users can also click delete to remove the movie from the wishlist.

User's blacklist page

The user can click the blacklist button to enter the user's blacklist page. Users can view all users in the blacklist on the page. Users can also remove them from the blacklist by clicking the delete button on the right side.
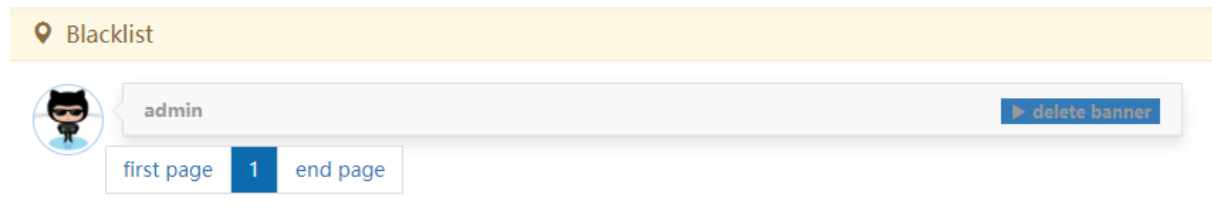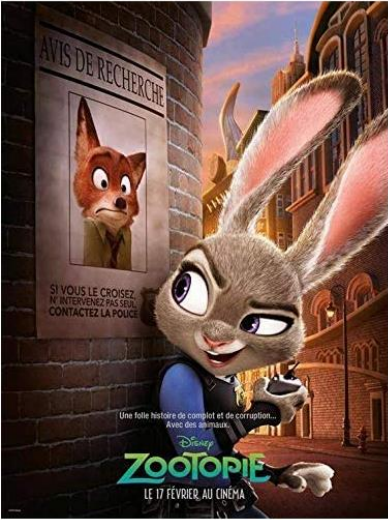

Movie browsing and searching

This feature allows all website users to search for movies and view the search results. Searching function is not only effective for registered users, visitors without a registered account can also use the search function of the website. The website provides a total of four search methods, which can be searched by name, director, genre, and description. The search supports fuzzy search and can ignore the difference in capitalization.
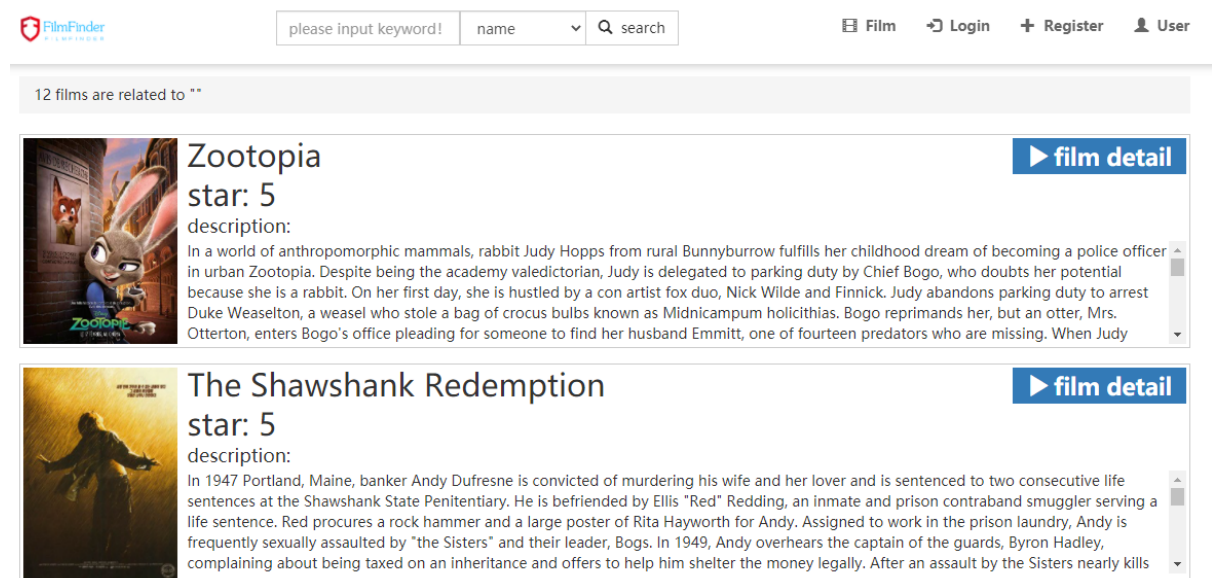
Movie detail page

The user can click on the film detail on the right side of the search results to enter the detailed information page of the movie. The upper side of the page displays the detailed information of the movie, including director, genre, duration, description, etc. On the left is the movie poster.

Film review, wishlist and blacklist

Scroll down on the detail page is the film review section of the movie. Here users can view the relevant ratings of the movie in the login and guest status. In addition to viewing movie reviews, users can also write movie reviews and rate movies. The user cannot be in the guest state when using this function and can only use this function after logging in to the account. Users can also click on wishlist on this page to add movies to their wishlist. This function also requires the user to log in before it can be used.

If users find an offensive comment, they can click the Add banner button next to the user. This will add users to the blacklist. Their comments will no longer be displayed on the page, and the movie ratings will be recalculated after excluding their ratings. This function also requires the user to be logged in.

Film recommendation

In the recommendation section, based on the user's current latest movie reviews, recommend movies that may be of interest to the user. If the user is not logged in to the system, the user is recommended to the movie with a higher score in the current system library. If the user has logged into the system, recommendations are made based on the user's reviews. Moreover, the user can also select the recommendation type, and the user can choose to recommend according to the director or genre.

Admin Dashboard
Control panel

The admin dashboard in FilmFinder has four main columns shown at the left of the webpage. The features in these four columns satisfy the need to monitor the website and manage different searching conditions.
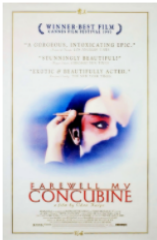
The first column is "First page". It is a control panel. The admin can monitor the percentage of memory used of FilmFinder. When the memory used changes, the pointer and the percentage number will change at the same time to make the admin able to know the situation in time.

The admin can also refresh the memory used situation himself by clicking the small refreshing icon. The real time memory used situation can be saved and downloaded for reference by clicking the download icon. On the right side of this page, the admin can set the number of limit speed, limit memory and limit clients of FilmFinder.
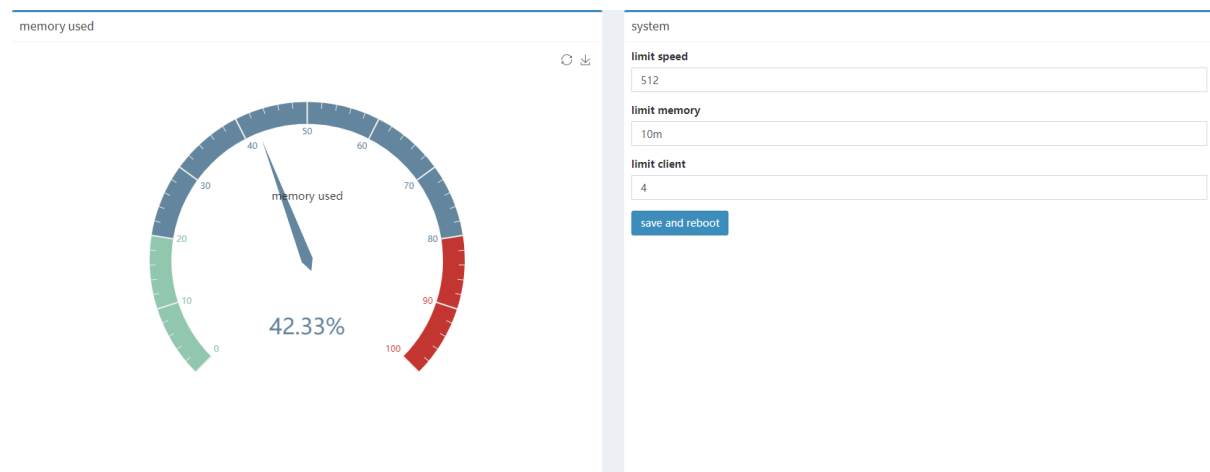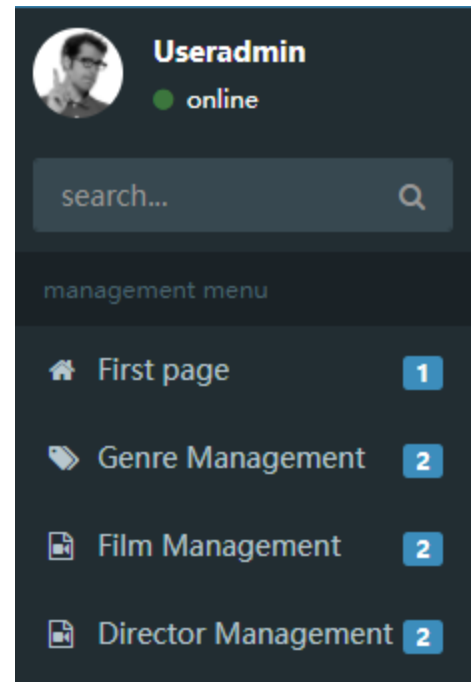
Genre management

The second column is "Genre Management". In this column, the admin has the access to add genre to the database in the first sub-column by typing the name of the genre and clicking the submit button. Once the new genre is submitted, it can be searched by the users in the user page.

| FilmFinder Admin | | | Genre management   >   Add Genre |

Add Genre

genre

please input genre!

submit

In the second sub-column, the admin can watch all the genres available for searching at that moment. He can search existing genres using the searching bar and has the access to delete the genre through clicking the red delete button at the right of the matched genre.

id	name	create time	options
8	drama	2020-11-07 14:06:05	delete
6	music	2020-10-06 10:56:54	delete
5	animation	2020-10-06 10:56:46	delete
2	horror	2020-10-06 10:56:29	delete
1	science	2020-10-06 10:52:14	delete

Film management

The third column is "Film Management". The admin can add or manage existing films in this column. In the first sub-column, the admin can add new films by inputting all relevant information including name, description, release length and release time. Meanwhile, the admin needs to upload a logo about the added films and selects matched genre and director from the drop-down menu. If the new added film does not belong to any of the existing genres or directors, the admin needs to add a new genre or director first before he adds the new film.

In the second sub-column of "Film Management", the admin can watch all the films available and their information in FilmFinder. He can search the film using the searching bar at the top right of this page. If he wants to delete any of the existing movies, he can easily do it by clicking the red delete button at the right of the matched film.

FilmFinder   🏠 Film Management > Film list

Film list								please input key 🔍
id	name	length	genre	star	play_num	comment_num	release_time	
22	Jaws	124 minutes		0 star	0	4	1975-06-20	delete
20	Up	96 minutes		0 star	0	12	2009-05-09	delete
19	Zootopia	108 minutes		5 star	1	109	2016-03-04	delete
18	WALL·E	98 minutes		0 star	0	15	2008-06-27	delete
17	Interstellar	169 minutes		0 star	0	12	2014-11-07	delete
16	Inception	148 minutes		0 star	0	15	2010-07-16	delete
15	Spirited Away	125 minutes		0 star	0	17	2001-07-20	delete
14	Life Is Beautiful	116 minutes		0 star	0	8	1997-12-20	delete
13	Titanic	194 minutes		0 star	0	13	1997-12-19	delete
12	Léon	133 minutes		0 star	0	10	1994-09-14	delete

Home 1 2 next page last page

Director management

The fourth column is "Director Management". In the first sub-column, The admin can add new directors into the database of FilmFinder by typing the name of the new director and clicking the blue submit button.

Add Director

director

please input director!

submit

In the second sub-column of "Director Management", the admin can view all the existing directors and the creating time of each of them. The admin can also search existing directors using the searching bar at the top right of this page. If the admin wants to delete any of the existing directors, he can easily do it by clicking the right delete button at the right of the matched directors.

Director List			please input key 🔍
id	**name**	**create time**	**options**
29	James Wan	2020-11-14 19:43:22	delete
28	George Lucas	2020-11-14 19:08:04	delete
27	Don Hall	2020-11-14 15:29:15	delete
25	Pete Docter	2020-11-07 16:40:35	delete
24	Billy Wilde	2020-11-07 14:04:06	delete
23	Gabriele Muccino	2020-11-07 14:03:53	delete
22	Francis Ford Coppola	2020-11-07 14:03:44	delete
21	Byron Howard	2020-11-07 14:03:36	delete
20	Andrew Stanton	2020-11-07 14:03:19	delete
19	Rajkumar Hirani	2020-11-07 14:03:11	delete

Home 1 2 3 next page last page

Future Feature Roadmap

Use distributed cache database to improve database I/O performance

There are many operations that require frequent database interactions on current websites. Among them, the most interactive and the most frequently used database is the blacklist function. According to demand, after users add people in the comment area to the blacklist, their comments will no longer be displayed, and the score will be recalculated. If the number of users of the website increases, the number of such concurrent operations will also increase. In the end, it may lead to long website response time. Therefore, in the future, we can apply distributed databases to solve this problem. Through distributed storage, the I/O performance of the database can be improved, thereby alleviating this problem.

Solve the problem of slow user response

This problem will arise when many users are querying simultaneously. There will be a certain delay in the system, resulting in a decrease in user experience. The reasons may be as follows: Firstly, Flask is a lightweight back-end framework, so its web server has limited performance and cannot handle high-throughput network requests. Secondly, there may be some defects in the system design, and the code logic needs to be optimized. Any redundant code may slow down the running speed of the program. In response to the above problems, we can make the following improvements: first use a high-performance server to deploy the program, such as uWSGI. uWSGI is a high-performance web application server that follows the unified gateway interface service. It can carry high concurrent operations

and improve the response speed of the website. Second, optimize the code structure and delete redundant code.

Adopt front-end frameworks such as react and angular

Most of the current web pages in the system are static, which means that if the information is changed, the web pages need to be refreshed. For example, after adding a user to the blacklist, the user will see the same score as before on the search results page, and then the user needs to refresh manually to update the score on the web page. It is a very bad experience for users to frequently refresh manually. Therefore, our countermeasure is to use front-end frameworks such as react and angular to bind the data and the model, so that the data can be refreshed and displayed.

Use machine learning to improve the recommendation system

The current recommendation system has the following problems: First of all, the problem of freshness, the highest-rated movies in the system may all belong to a certain director or a certain genre, which subconsciously misleads the user's actual interest in the director or movie genre. Secondly, the current recommendation only considers the user's scoring situation. It may happen that the user likes a certain genre or director, and it happens that this work does not meet the user's expectations. Therefore, in the future, a certain degree of calculation can be considered for the user's evaluation content, combined with text content and scores for recommendation.

Implementation Challenges

Database relational design

When our team first designed the database, we only considered the primary key and foreign key relationship. This leads to many problems when the admin system is operating. For example, the administrator cannot delete a movie with reviews, because there is a corresponding relationship between the movie table and the comment table. This is obviously not in line with the actual situation, so we modified the foreign key constraints

between the movie table, director table, comment table, and genre table according to actual needs, so that cascading deletions are supported between them.

Run the program in CSE

The CSE computer is a 6.0 series Debian operating system that supports the front-end and back-end frameworks we use. But we encountered great obstacles in installing the database. First, MySQL is not installed in the CSE machine. Secondly, when we try to download and install MySQL from the official website, the CSE machine will prevent us to do that operating. This means that our database may not be able to run directly on the CSE computer. Therefore, we have configured two cloud databases, AWS EC2 and google cloud. Just modify the __init__ setting in the program, we can connect the program to the cloud database, which can ensure that the program runs on the CSE machine.

Wrong number of comments on detail page

When designing the database, we used id as the primary key of the comment. There is insufficient communication between teammates of our database design and object-relational mapping. This causes the number of comments on the movie detail page to be different from the actual number of comments. After careful review of the code, we found that the teammate in charge of ORM set the number of comments equal to MAX(comment_id), so a very funny error occurred. After we updated the database with the SQL command again, the number of comments can now be displayed correctly.

Implementation of recommended functions

According to demand, we need to recommend similar movies based on the user's latest review, combined with genre and director. However, the explanation here is not rigorous, and several extreme situations are not considered. Therefore, we have designed a recommendation based on the users' latest 10 film reviews. According to their scores from the past 10 film reviews, 1-2 points mean they do not like it, 3 means they have a normal attitude, and 4-5 means they like it very much. Filter out their favourite movies and genres and make recommendations based on the relevant information of the movies currently viewed. In this way, some extreme situations can be avoided, and the recommendation system can recommend as many movies as possible to users.

Third-party Functionalities/Frameworks

Library List

We will introduce here some third-party libraries that we use in the project. We use Flask as a web framework to implement our website. We also use Flask-SQLAlchemy as the connector between the backend and the database. It converts data tuples in the database into Python objects, which allows us to easily manipulate the data. In addition, we use Flask-WTF for all form validation to check whether user input is valid on different pages. The specific list is as follows:

Libraries	Edition
click	7.1.2
Flask	1.1.2
Flask-SQLAlchemy	2.4.4
Flask-WTF	0.14.3
itsdangerous	1.1.0
Jinja2	2.11.2
MarkupSafe	1.1.1
PyMySQL	0.10.1
SQLAlchemy	1.3.19
Werkzeug	1.0.1
WTForms	2.3.3

Library Introduction

PyMySQL 0.10.1

PyMySQL is a python MySQL client library. It is used to connect SQLAlchemy to the database system. Most of the APIs in PyMySQL are able to use with MySQLclient and MySQLdb.

SQLAlchemy 1.3.19

SQLAlchemy is an open-source SQL toolkit for python programming. It helps to connect the database and PyMySQL and provides a series of enterprise-level persistence patterns.

Flask 1.1.2

Flask is a web framework which serves the development and extension during implementation. The flask is based on Werkzeug and Jinja.

Flask-WTF 0.14.3

Flask-WTF is a feature to provide integration with WTForms. It is safely secured and allowed file upload works with Flask upload.

Flask-SQLAlchemy 2.4.4

Flask-SQLAlchemy is an extension for Flask and it provides support for SQLAlchemy in the implementation by some defaults and helpers. It will be easier and efficient to implement and achieve our goal when using Flask-SQLAlchemy.

Werkzeug 1.0.1

Werkzeug is a comprehensive WSGI web application library for python programming.

Jinja2 2.11.2

Jinja2 is a template engine for python programming. It is a text-based template so that it can be used to produce different markup or python source code.

Click 7.1.2

Click is a python package that allows to make command line interfaces with limited amounts of code. Besides, Click can continue to act as intended when multiple Clicks are used together.

MarkupSafe 1.1.1

MarkupSafe can be used in HTML and XML to transfer those characters with special meanings into real meaningful characters. It helps to display the inputs by those untrusted users.

WTForms 2.3.3

WTForms is a flexible forms validation and rendering library for python web development. It can help to create web forms and use the information in the forms easily with a python model.

Itsdangerous 1.1.0

Itsdangerous is used to help send the data safely in an untrusted environment. The basic idea is to cypher the data you want to send and only the true receiver can decipher and read your data. The other people who are not the matched receiver or do not have the secret key cannot read or modify the data.

System Setup Documentation

Use cloud database

We have set up two cloud databases in total, AWS EC2 and google cloud. Their locations are Ohio, US and Sydney, AU.
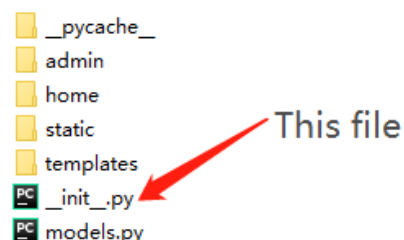
Google Cloud

The ip address of google cloud is 34.87.235.63. Download the code from github, it will connect to the google cloud database by default.

AWS EC2

The ip address of AWS EC2 is 18.220.148.52. Since AWS EC2 is in the United States, the response time between website operations is relatively long. Please use it when google cloud is not available.

How to switch cloud database?

The connection to the database is set in the code __init__.py file.


```
# google cloud database
```

```
app.config['SQLALCHEMY_DATABASE_URI'] = "mysql+pymysql://roo
t:FilmFinder123,.@34.87.235.63:3306/film"
# AWS EC2
# app.config['SQLALCHEMY_DATABASE_URI'] = "mysql+pymysql://r
oot:FilmFinder123,.@18.220.148.52:3306/film"
```

The highlighted code is to connect to google cloud by default. The code which below the comment (# AWS EC2) is the setting which connect to the AWS EC2. If google cloud database cannot be used, please comment out the code and remove the # sign in the part of AWS EC2 code. The code shows as below:
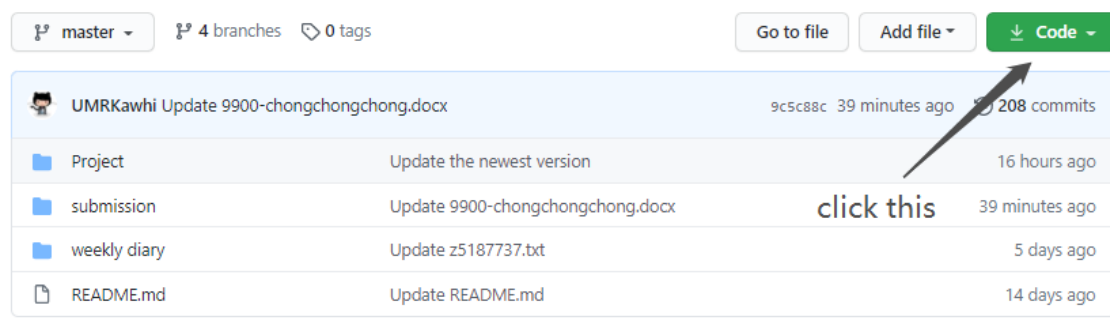
```
# google cloud database
# app.config['SQLALCHEMY_DATABASE_URI'] = "mysql+pymysql://r
oot:FilmFinder123,.@34.87.235.63:3306/film"
# AWS EC2
app.config['SQLALCHEMY_DATABASE_URI'] = "mysql+pymysql://roo
t:FilmFinder123,.@18.220.148.52:3306/film"
```
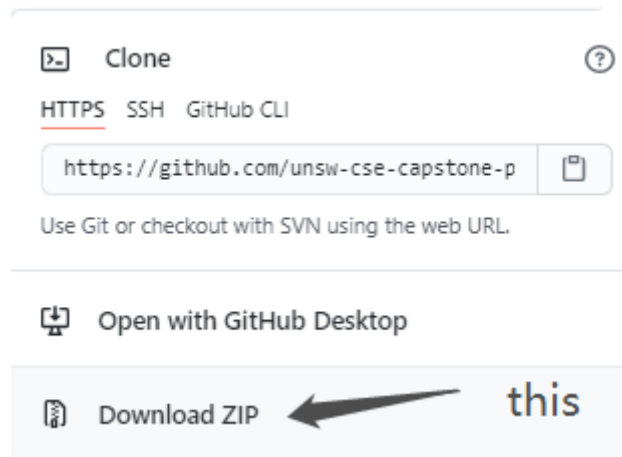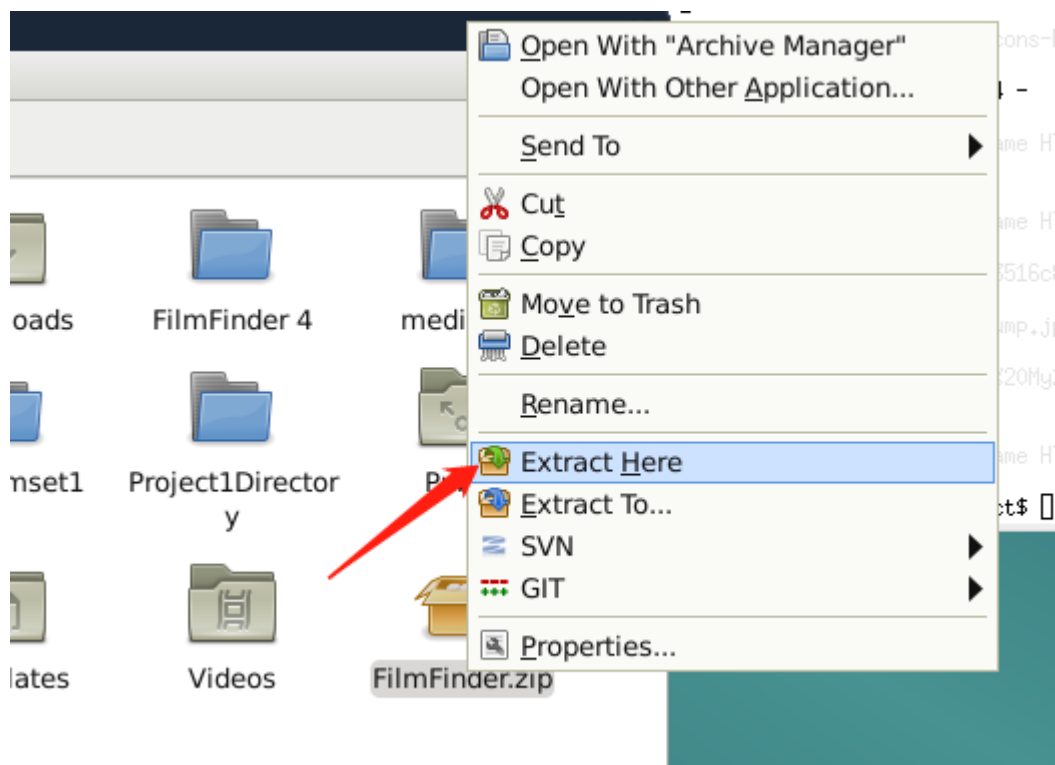
Running program On CSE machine

1. Download the code from github, click https://github.com/unsw-cse-capstone-project/capstone-project-comp9900-h18a-chongchongchong.git

Click the link to jump to the github page of chongchongchong. The page is shown in the figure above. Click the green code button and select download zip.
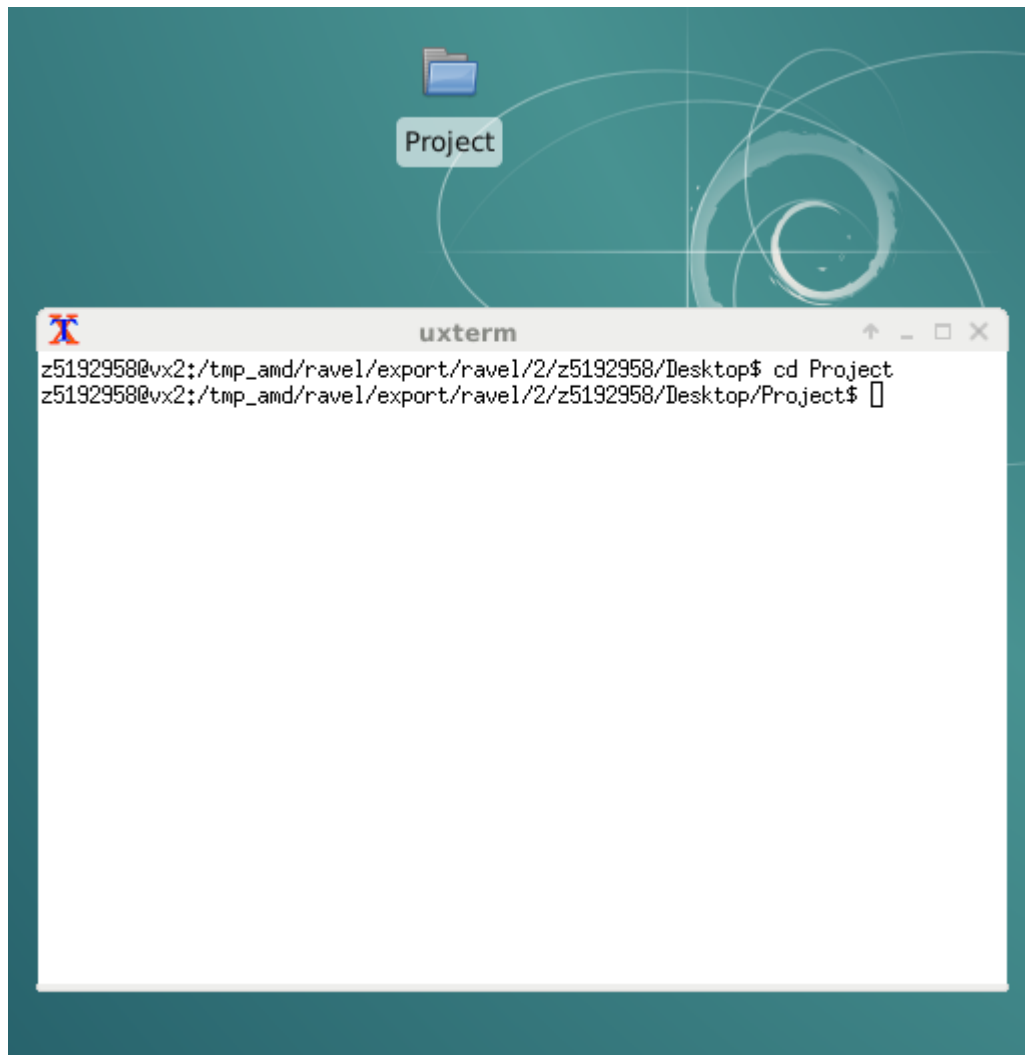
2. **Find the folder and unzip**

Find the directory where the compressed package is located, right-click and click Extract here.

3. **Use the command line tool to enter the folder and install the necessary libraries**

Use the cd command to enter the project folder, the path is shown below:

Use the command **pip3 install -r requirements.txt** to install the necessary libraries

After successful installation, you can see the following information on the command line:

```
Successfully installed Flask-1.1.2 Flask-SQLAlchemy-2.4.4 Flask-WTF-0.14.3 Jinja2-2.11.2 MarkupSafe-1.1.1 PyMySQL-0.10
.1 SQLAlchemy-1.3.19 WTForms-2.3.3 Werkzeug-1.0.1 click-7.1.2 itsdangerous-1.1.0
```
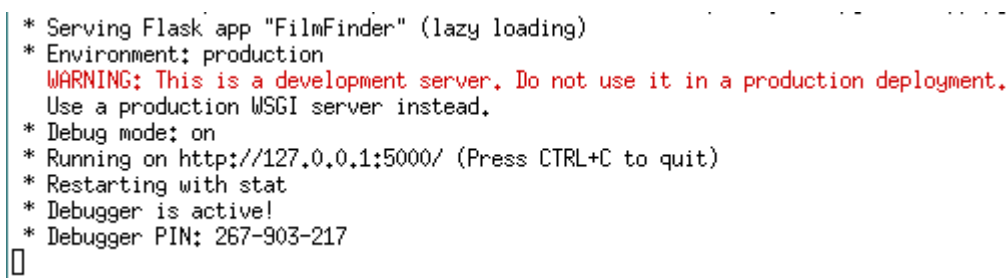
4. **Starting program**

Using the command **python3 app.py** to activate the program:

After the program is successfully started, you will see the following information on

the command line:

```
* Serving Flask app "FilmFinder" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 267-903-217
```

5. **Viewing the website**

   Use any browser, enter 127.0.0.1/5000 to start accessing the web page. Here is the

   default Firefox browser of CSE as an example.
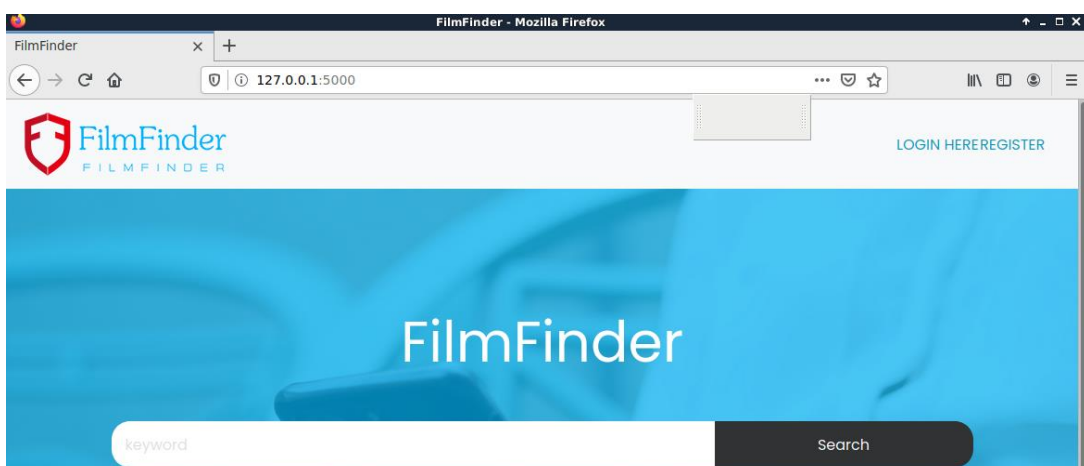
Problems and solutions

1. **The library installation failed because of the backward pip version**

Use the command **python3 -m pip install pip –upgrade** to update the pip
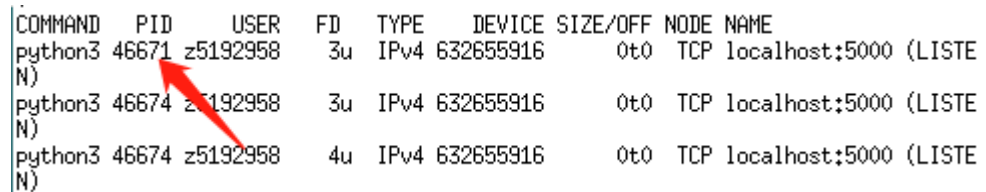version, and then use the installation command.

2. **OSError: [Errno 98] Address already in use**

   Because flask comes with its own server, and the default port is 5000. If the
   previous program is not suspended, port 5000 will be continuously occupied.
   Therefore, you may get this error message when starting the program.
   There are two solutions to fix this problem:

   - **Kill the process using the command line**

     First use lsof -i:5000 on the command line to view the usage of port 5000.
     If port 5000 is occupied, you can see its process information. As shown
     below. Select one of the PIDs, 46671 or 46674, and enter kill + PID on the
     command line. For example, kill 46671 or kill 46674. After performing the
     above operations, you can kill the previous process, and then use
     python3 app.py to start the program.

     ```
     COMMAND  PID     USER    FD  TYPE   DEVICE SIZE/OFF NODE NAME
     python3 46671 z5192958   3u  IPv4 632655916     0t0  TCP localhost:5000 (LISTE
     N)
     python3 46674 z5192958   3u  IPv4 632655916     0t0  TCP localhost:5000 (LISTE
     N)
     python3 46674 z5192958   4u  IPv4 632655916     0t0  TCP localhost:5000 (LISTE
     N)
     ```

     **Warning Warning Warning!!! In extremely special circumstances, it will
     be found that there is no process information, but the port is still
     occupied. In this case, please see the second measure.**

   - **Change program port parameters**

     In the case that the first method fails, you can enter the program folder,
     find app.py and click to enter. In the fifth line of the program, in the
     parentheses of app.run(), add the port parameter. The computer's 1024-
     49151 ports are dynamic ports, so you can choose one at random and fill
     in it. The example here uses port 5454. Save after modification.

```
1 from FilmFinder import app, db
2
3 if __name__ == '__main__':
4     app.debug=True
5     app.run(port=5454)
6
```

Use python3 app.py to start the program. If successful, the program will start on port 5454. Therefore, the address in the browser will also become 127.0.0.1/5454. If the address is still occupied, repeat the above operation until a free port is found.

Reference

Flask PyPi 2020, Python Package Index, accessed 30 Oct 2020,

https://pypi.org/project/Flask/

Werkzeug PyPi 2020, Python Package Index, accessed 29 Oct 2020,

https://pypi.org/project/Werkzeug/

How To Install MySQL on Ubuntu 18.04 Mark Drake 2020, Digital Ocean, accessed 16 Oct 2020,

https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-18-04

Use MySQL Relational Databases on Debian 6 (Squeeze), Linode, accessed 24 Oct 2020,

https://www.linode.com/docs/guides/use-mysql-relational-databases-on-debian-6-squeeze/

Flask: OSError: [Errno 98] Address already in use, stackooverflow, accessed 12 Nov 2020,

https://stackoverflow.com/questions/50784137/flask-oserror-errno-98-address-already-in-use-but-why

SQLAlchemy 1.4 Documentation, SQLAlchemy, accessed 12 Oct 2020,

https://docs.sqlalchemy.org/en/14/

Bootstrap 4 Tutorial, w3schools.com, accessed 12 Oct 2020,

https://www.w3schools.com/bootstrap4/

Google cloud sql authorize all Ips, stackoverflow, accessed 14 Nov,

https://stackoverflow.com/questions/28339849/google-cloud-sql-authorize-all-ips