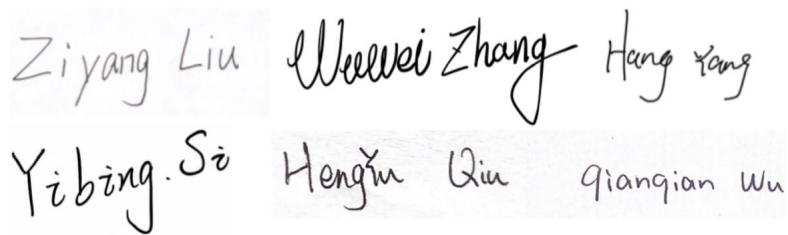


Design Document

COMP208 GROUP PROJECT



A photograph of six handwritten signatures in black ink on a light-colored background. The signatures are arranged in two rows. The first row contains 'Ziyang Liu', 'Wuwei Zhang', and 'Hang Yang'. The second row contains 'Yibing Si', 'Hengyu Qiu', and 'qianqian Wu'.

TEAM39

Ziyang Liu, Wuwei Zhang, Hang Yang,
Yibing Si, Hengyu Qiu, Qianqian Wu

March, 2021

Contents

1	Changes to Specification	3
1.1	Project Review	3
1.2	Changes	3
1.3	Supplement	3
2	Database Design	4
2.1	ER Diagram	4
2.2	Data Dictionary	5
2.3	Logical Table Structure	9
2.4	Physical Table Structure	10
3	Business Rules	13
3.1	Business Policies	13
3.2	Business Rules	13
3.3	Business Rules force constraints on the data in the database	13
3.4	Privacy Policy and Terms of Service	13
3.5	Validation Rules	14
4	Transaction Table/Matrix	14
5	Algorithm Design	16
6	Process Design	17
6.1	Use Case Diagram	17
6.2	Use Case Description	18
6.3	Data Flow Diagram	25
6.4	Navigation structure diagrams	26
6.5	Interface Design	27
6.6	Story Board	29
7	Testing Design	30
7.1	Testing Strategy	30
7.2	Test Form	30
8	Planning and Risk Assessment	32
8.1	Planning	32
8.2	Required Skills	33
8.3	Challenge	33
8.3.1	General Challenge	33
8.3.2	Technological Challenge	33
9	References	34

1 Changes to Specification

1.1 Project Review

The purpose of our project is to develop a movie website to help users obtain their potential favorite movies based on collaborative filtering which analysis user's preference and behavior, moreover, to build an interest-oriented platform to record, share ideas and communicate with other users.

1.2 Changes

During the design phase of the project, we made three major changes and four supplements based on our initial requirements.

First of all, regarding the recommendation algorithm, we decided to use the collaborative filtering algorithm instead of the convolutional neural network. The collaborative filtering algorithm saves time in developing languages, analyzing documents, developing parsing tools and stemming algorithms, mainly focusing on clustering algorithms[1]. On the basis of the Item-based collaborative filtering algorithm, other algorithms may be used to implement process.

Second, we decided to use MySQL database instead of other non-relational models. MySQL is a stable and reliable database, and it can be connected to the system to promote the effective management of the database. Its reliability, high performance and on-demand scalability have a significant effect on our data storage and system implementation[2]. We decide to use the TMDb api as the main data source for our website, including movies, actor, genre and cast information. We also use 'TMDb Dataset' and 'The Movies Dataset' in Kaggle which contains the user's history and as the data source of the collaborative filtering system.

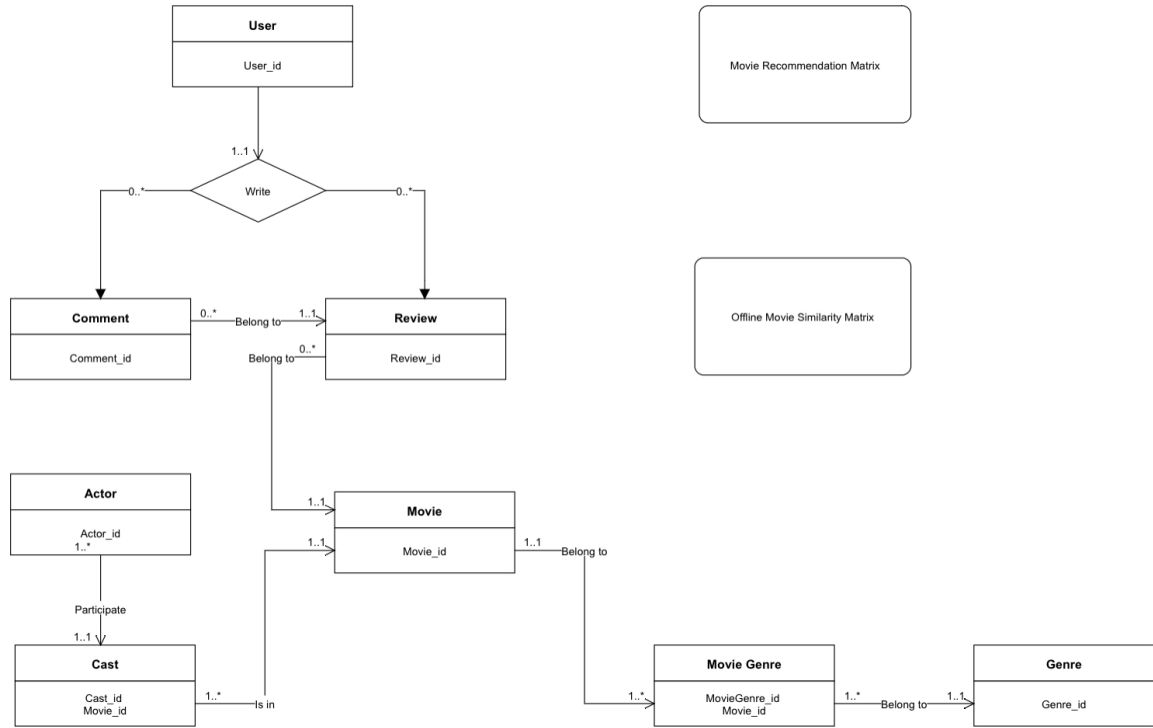
Then, for the choice of front-end and back-end programming, we decided to mainly use HTML, CSS and PHP. And algorithm programming mainly uses Python.

1.3 Supplement

- Verification of registration - valid email verification is required for registration.
- Added Features - Added the ability to view top reviews and actor info pages.
- Added user-case - Add user-case such as View actor and Messages notify.
- Source Control - Github will be used to do remote cooperation and source control.

2 Database Design

2.1 ER Diagram



The movie will be directly displayed on website based on some filling factor. Each movie belongs to one or many specific genres. Each movie has a particular cast. Each cast consists of one or many actors. Each review is published by one particular user. And each comment is under one particular review, each comment is published by one particular user. Once visitor completing register, he or she will become a user. Movie genre is the implied relation between movie and genre.

Offline movie similarity matrix is derived through recommend system, the Collaborative filters will generate save a cosine similarity matrix to check the similarity between movies. Movie recommendation matrix is derived through recommend system, the Collaborative filters will constantly save and modify this matrix based on the user's ranking and the features of movies then put the (movie_id, recommend level) tuple with higher score to the front position of the list.

2.2 Data Dictionary

In this section, there are 4 types of dictionary. The first is an entities table. This table formulate the name of each entity and explain the occurrence of each. Second is the table of each attribute of entities. We stipulate the data type, length, and key of attributes, and for each attribute we determine whether it can be null or multi-valued or not. The third table is the relation table, it describes the relation type, name between entities. The last is a derivative table. It contains the important derivative data and its structure in the collaborative filtering part.

Table 1: Entities Table

Entity Name	Description	Aliases	Occurrence
Movie	The movie itself, the major content in our website	Film	Will be directly displayed on website based on some filling factor
MovieGenre	The movie with one of its genres		The implied relation between movie and genre
Genre	The type of the movie	type, style, label	Each movie belongs to 1 or many specific genres
Cast	The movie participants	members, participants	Each movie has a particular cast
Actor	The director or performer	Director, Actor, Performer, Actress	Each cast consists of one or many actors
Review	The review and ranking from user	score, grade, evaluation	Each review is published by one particular user
Comment	The comment under the review		Each comment is under a particular review, each comment is published by one particular user
User	The people who has registered in our website		Once visitor completing register, he/she becomes a user

Table 2: Attributes Table

Entity	Attributes	Description	Data type and length	Key	Multivalued	NULL
Movie	movie_id	The unique ID to identify the movie	int(9)	PK	No	No
	tmdb_id	The original ID in TMDb	int(9)		No	Yes
	imdb_id	The original ID in IMDb	varchar(9)		No	Yes
	title	The tile of the movie	varchar(55)		No	No
	original_title	The original titile of the movie	varchar(55)		No	Yes
	collection	The series or collection the movie belongs to	varchar(20)		No	Yes
	languages	The language of the movie	varchar(20)		No	No
	release_date	The release date of the movie	datetime		No	No
	overview	The overview of the movie	varchar(511)		No	Yes
	vote_count	The total number of user's rate	int(9)		No	No
	vote_average	The average rate for this movie	float(5)		No	No
	country	The country where the movie release	varchar(20)		Yes	No
	runtime	The runtime of the entire movie	int(5)		No	Yes
	poster_path	The URL of the movie poster	varchar(100)		No	Yes
MovieGenre	genre_id	The unique ID of the movie genre	int(9)	FK,PK	No	No
	movie_id	The unique ID to identify the movie	int(9)	FK,PK	No	No
Genre	genre_id	The unique ID of the movie genre	int(9)	PK	No	No
	genre_name	The name of the movie genre	varchar(15)	AK	No	No
Cast	actor_id	The unique ID to identify the actor	int(9)	FK,PK	No	No
	movie_id	The unique ID to identify the movie	int(9)	FK,PK	No	No
	identity	Performer. Director or both.	varchar(15)		Yes	Yes

Actor	actor_id	The unique ID to identify the actor	int(9)	PK	No	No
	tmdb_id	The original ID in TMDb	int(9)		No	Yes
	imdb_id	The actor ID in IMDb	varchar(9)		No	Yes
	name	The name of the actor	varchar(55)		No	No
	biography	The description of the actor	varchar(511)		No	Yes
	profile_path	The URL of the actor's profile	varchar(100)		No	Yes
	gender	The gender of the actor	enum('male', 'Female', 'Other')		No	No
	date_birth	The birthday of the actor	datetime		No	No
Review	review_id	The unique ID to identify the review	int(9)	PK	No	No
	movie_id	The unique ID to identify the movie	int(9)	FK	No	No
	user_id	The unique ID from the author of the review	int(9)	FK	No	No
	content	The content of the review	varchar(511)		No	Yes
	rating	The rating scored by user	int(2)		No	No
	time_stamp	The last editing time of the review	datetime		No	No
Comment	comment_id	The unique ID to identify the comment	int(9)	PK	No	No
	review_id	The unique ID to identify the review	int(9)	FK	No	No
	user_id	The unique ID to identify the user	int(9)	FK	No	No
	content	The content of the comment	varchar(255)		No	No
	time_stamp	The published time of the comment	datetime		No	No
User	user_id	The unique ID to identify the user	int(9)	PK	No	No
	nickname	The nickname of the user	varchar(15)		No	No
	email	The email address of the user	varchar(50)	AK	No	No
	description	The personal description of the user	varchar(255)		No	Yes
	gender	The gender of the user	enum('male', 'Female', 'Other')		No	No
	profile_path	The URL of the user's profile	varchar(100)		No	Yes
	password	The password of the user account	varchar(15)		No	No
	time_stamp	The registered time of the user	datetime		No	No

Table 3: Relation Table

Entity name	Description	Aliases	Occurrence	Entity
Movie	1...1	has	0...*	Review
Movie	1...1	has	1...*	Cast
Movie	1...1	belong to	1...*	MovieGenre
MovieGenre	1...*	has	1...1	Movie
MovieGenre	1...*	belong to	1...1	Genre
Genre	1...1	has	1...*	MovieGenre
Cast	1...*	belong to	1...1	Movie
Cast	1...1	has	1...*	Actor
Actor	1...*	participate	1...1	Cast
Review	1...*	is in	1...1	Movie
Review	1...1	has	0...*	Comment
Review	0...*	is witten by	1...1	User
Comment	0...*	belong to	1...1	Review
Comment	0...*	is witten by	1...1	User
User	1...1	write	0...*	Review
User	1...1	write	0...*	Comment

Table 4: Derivative Matrix

matrix_name	head(row(x))	tail(row(x))
*offline movie similarity matrix	movie_id (int)	list[(movie_id, similarity)]
*movie recommendation matrix	user_id (int)	list[(movie_id, recommend level)]

*Offline movie similarity matrix stores the similarity between movies
Each row is a list within specific limited length, the similarity is sorted in Descending order. The head of the row is the movie_id, the tail is the list of tuples for (movie_id. similarities)

*Movie recommendation matrix stores the recommended movie for users.
Each row is a list within specific limited length, the recommend level is sorted in Descending order. the head of the row is the movie_id, the tail is the list of tuples for (movie_id. recommend level)

2.3 Logical Table Structure

Movie (movie_id, tmdb_id, imdb_id, title, original_title, collection, languages, release_date, overview, vote_count, vote_average, country, runtime, poster_path) Primary Key movie_id
MovieGenre (genre_id, movie_id) Primary Key movie_id Primary Key genre_id Foreign Key movie_id References Movie(movie_id) Foreign Key genre_id References Genre(genre_id)
Genre (genre_id, genre_name) Primary Key genre_id Alternate Key genre_name
Cast (actor_id, movie_id, identity) Primary Key actor_id Primary Key movie_id Foreign Key actor_id References Actor(actor_id) Foreign Key movie_id References Movie(movie_id)
Actor (actor_id, tmdb_id, imdb_id, name, biography, profile_path, gender, date_birth) Primary Key actor_id
Review (review_id, movie_id, user_id, content, rating, time_stamp) Primary Key review_id Foreign Key user_id References User(user_id) Foreign Key movie_id References Movie(movie_id)
Comment (comment_id, review_id, user_id, content, time_stamp) Primary Key comment_id Foreign Key user_id References User(user_id) Foreign Key review_id References Review(review_id)
User (user_id, nickname, email, description, gender, profile_path, password, time_stamp) Primary Key user_id Alternate Key email

The logical table is the expansion of the ER Diagram. As the diagram shows, the relation between Movie and Review, Cast, MovieGenre is all 1:m, therefore we use movie_id in Review, Cast, MovieGenre as the FK because the movie_id is the PK of the Movie. Moreover, as the Cast, MovieGenre is the derivative relation between movie and actor, genre. We also use actor_id as the FK in Cast, genre_id as the FK in MovieGenre. Therefore, the PK in Cast is (actor_id, movie_id), the PK in MovieGenre is (genre_id, movie_id). The one user can write many Review and Comment, therefore the user_id is the FK in Review and Comment. One Comment belongs to one Review but one Review can have many Comment, therefore the review_id is the FK in comment. One movie can have many review, thus the movie_id in Review can be used to identify that movie the Review belongs to.

2.4 Physical Table Structure

	Domain	ID	integer with fixed length 9
	Domain	Imdb	fixed char with length 9
	Domain	Title	varchar with max length 55
	Domain	Date	fixed length of char format yyyy-mm-dd
	Domain	Details	varchar with max length 20
	Domain	Overview	varchar with max length 511
	Domain	Vote_count	variable length integer with max length 9
	Domain	Vote_average	float between 1-10 and with precision 5
	Domain	Run_time	variable length integer with max length
	Domain	File_path	varchar with max length 100
Movie(movie_id	ID	NOT NULL,
	tmbd_id	ID	,
	imbd_id	Imbd	,
	title	Title	NOT NULL,
	original_title	Title	,
	collection	Details	,
	languages	Details	NOT NULL,
	release_date	Date	NOT NULL,
	overview	Overview	,
	vote_count	Vote_count	NOT NULL,
	vote_average	Vote_average	NOT NULL,
	country	Details	NOT NULL,
	runtime	Run_time	,
	poster_path	File_path)
	Primary Key	movie_id	
	Domain	ID	integer with fixed length 9
MovieGenre(movie_id	ID	NOT NULL,
	genre_id	ID	NOT NULL)
	Primary Key	movie_id	
	Primary Key	genre_id	
	Foreign Key	movie_id	References Movie(movie_id)
	Foreign Key	genre_id	References Genre(genre_id)
	Domain	ID	integer with fixed length 9
	Domain	Name	varchar with max length 15
Genre(genre_id	ID	NOT NULL,
	genre_name	Name	NOT NULL)
	Primary Key	genre_id	
	Alternate Key	genre_name	

	Domain	ID	integer with fixed length 9
	Domain	Name	varchar with max length 15
Cast(movie_id	ID	NOT NULL,
	actor_id	ID	NOT NULL,
	identity	Name)
	Primary Key	movie_id	
	Primary Key	actor_id	
	Foreign Key	movie_id	References Movie(movie_id)
	Foreign Key	actor_id	References Actor(actor_id)
	Domain	ID	integer with fixed length 9
	Domain	Imdb	fixed char with length 9
	Domain	Name	varchar with max length 55
	Domain	Date	fixed length of char format yyyy-mm-dd
	Domain	Biography	varchar with max length 511
	Domain	File_path	varchar with max length 100
	Domain	Gender	enumeration char of ('male', 'Female', 'Other')
Actor(actor_id	ID	NOT NULL,
	tmdb_id	ID	,
	Imdb_id	Imdb	,
	name	Name	NOT NULL,
	biography	Biography	,
	profile_path	File_path	,
	gender	Gender	NOT NULL,
	date.birth	Date	NOT NULL)
	Primary Key	actor_id	
	Domain	ID	integer with fixed length 9
	Domain	Imdb	fixed char with length 9
	Domain	Name	varchar with max length 55
	Domain	Date	fixed length of char format yyyy-mm-dd
	Domain	Biography	varchar with max length 511
	Domain	File_path	varchar with max length 100
	Domain	Gender	enumeration char of ('male', 'Female', 'Other')
Actor(actor_id	ID	NOT NULL,
	tmdb_id	ID	,
	imdb_id	Imdb	,
	name	Name	NOT NULL,
	biography	Biography	,
	profile_path	File_path	,
	gender	Gender	NOT NULL,
	date.birth	Date	NOT NULL)
	Primary Key	actor_id	

	Domain	ID	integer with fixed length 9
	Domain	Time	char format yyyy-mm-dd hh:mm:ss
	Domain	Content	varchar with max length 511
	Domain	Rating	int between 1-10
Review(review_id	ID	NOT NULL,
	movie_id	ID	NOT NULL,
	user_id	ID	NOT NULL,
	content	Content	,
	rating	Rating	NOT NULL,
	time_stamp	Time	NOT NULL)
	Primary Key	review_id	
	Foreign Key	movie_id	References Movie(movie_id)
	Foreign Key	user_id	References User(user_id)
	Domain	ID	integer with fixed length 9
	Domain	Time	char format yyyy-mm-dd hh:mm:ss
	Domain	Content	varchar with max length 255
Comment(comment_id	ID	NOT NULL,
	review_id	ID	NOT NULL,
	user_id	ID	NOT NULL,
	content	Content	,
	time_stamp	Time	NOT NULL)
	Primary Key	comment_id	
	Foreign Key	review_id	References Review(review_id)
	Foreign Key	user_id	References User(user_id)
	Domain	ID	integer with fixed length 9
	Domain	Name	varchar with max length 15
	Domain	Email	varchar with max length 50
	Domain	Description	varchar with max length 255
	Domain	Gender	enumeration ('male', 'Female', 'Other')
	Domain	File_path	varchar with max length 100
	Domain	Password	varchar with min length 9 max length 15
	Domain	Time	char format yyyy-mm-dd hh:mm:ss
User(user_id	ID	Password
	nickname	Name	NOT NULL,
	email	Email	NOT NULL,
	description	Description	,
	gender	Gender	NOT NULL,
	profile_path	File_path	,
	password	Password	NOT NULL,
	time_stamp	Time	NOT NULL)
	Primary Key	user_id	
	Alternate Key	email	

3 Business Rules

3.1 Business Policies

- Once visitor completing register, he/she becomes a user.
- Only registered users may use comment and review.
- Users have the right to score movies.
- The web may recommend few movies to users.
- Every movie will be classified

3.2 Business Rules

- Each user should set up to 9 account ID and 15 digits password.
- Each movie belongs to one or many specific genres.
- Each user can only be recommended at most 50 movies at one time.
- Each Email Address can only be used to register one account.

3.3 Business Rules force constraints on the data in the database

- The email under different user_id must be unique.
- Each movie in the database will be related to one or different genres.
- The user's account ID is limited to 9 digit and the password is limited to 9-15 characters.
- Each comment is under one particular review, each comment is published by one particular use.

3.4 Privacy Policy and Terms of Service

This website formulates privacy policies and related clauses to protect the interests of users.

The main purpose is to demonstrate how the system collects user data and what services the system will provide:

1. Register as a user

To complete the creation of an account, you need to provide an email address for verification of registration. After the user registration is completed, your email address will be used as the log in account name by default. If you do not agree, you will not be able to complete the registration.

The above information you provide will continue to authorize us to use it during your use of this service. When you delete your account, we will stop using and delete the above information.

2. Message notification

Message notifications are set up on the website to facilitate you to receive daily movie recommendations, reminders on new movies, and user comment responses in time.

More importantly, the website should not provide third-party organizations with any function to sell user information.

3.5 Validation Rules

1. If a user is not registered and wants to access the system (such as the homepage), then the server will detect whether there is a record of the user in the session. If not, then this user will be restricted by permission and cannot view comments, because only if the user inputs the correct email and password, would the server record their ids in the session.
2. Regular expression can be used to check users' password. Finally, the most of cases that the server recognizes users is to specify their ids in the URL, which is a kind of GET method of HTTP request.
3. When a user wants to post a comment, the server will first check whether the current sender is the user recorded in the conversation, and check whether the recipient does exist in the database.

4 Transaction Table/Matrix

I: Insert R: Read U: Update D: Delete

Transactions:

- a) Create movie profile
- b) Edit genre
- c) Create User
- d) Create Actor

Transaction /Table	a)				b)				c)				d)			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
Movie	X					X										
Genre	X					X	X									
Movie genre	X					X	X							X		
Cast	X												X	X		
Actor	X												X	X		
User									X	X						
Review																
Comment																

Transactions:

- e) Edit actor
- f) Create review
- g) Create comment
- h) Delete movie

Transaction /Table	e)				f)				g)				h)			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
Movie																X
Genre																
Movie genre																X
Cast																X
Actor		X	X													
User																
Review					X	X										X
Comment									X	X						X

Transactions:

- i) Delete user
- j) Edit cast
- k) Delete comment

Transaction /Table	i)				j)				k)			
	I	R	U	D	I	R	U	D	I	R	U	D
Movie												
Genre												
Movie genre												
Cast						X	X					
Actor												
User				X								
Review		X		X								
Comment		X		X								X

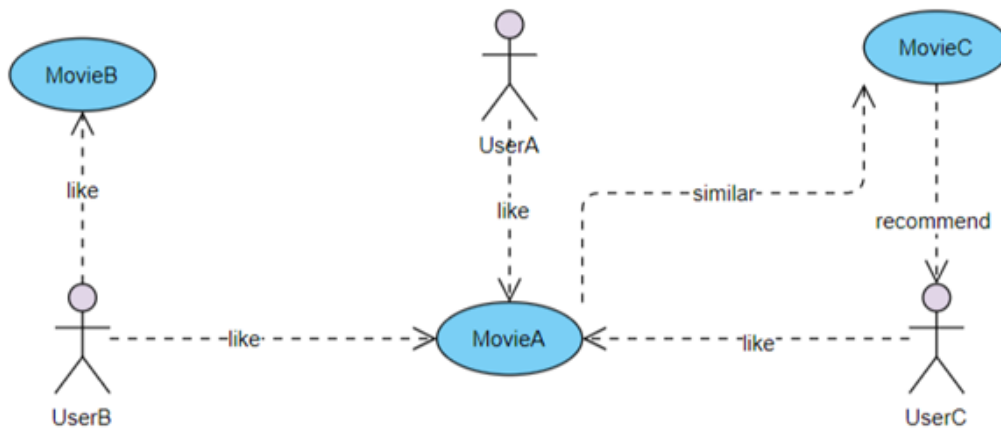
5 Algorithm Design

The key algorithm is Item-based and User-based collaborative filtering algorithm. Item-based collaborative filtering algorithm recommends movies that are similar to the movies users liked before. That means to find the relevance of movies based on the user's historical information, and then generate recommendation information.

The steps of this algorithm are to first calculate the similarity between the movies, and then calculate the predicted value to predict and recommend the movies that the user has not scored.

The advantage of this algorithm is that it has strong interpret-ability and meets real-time requirement, so we use it to recommend movies that users might like.

Below is a simple picture description of this algorithm:

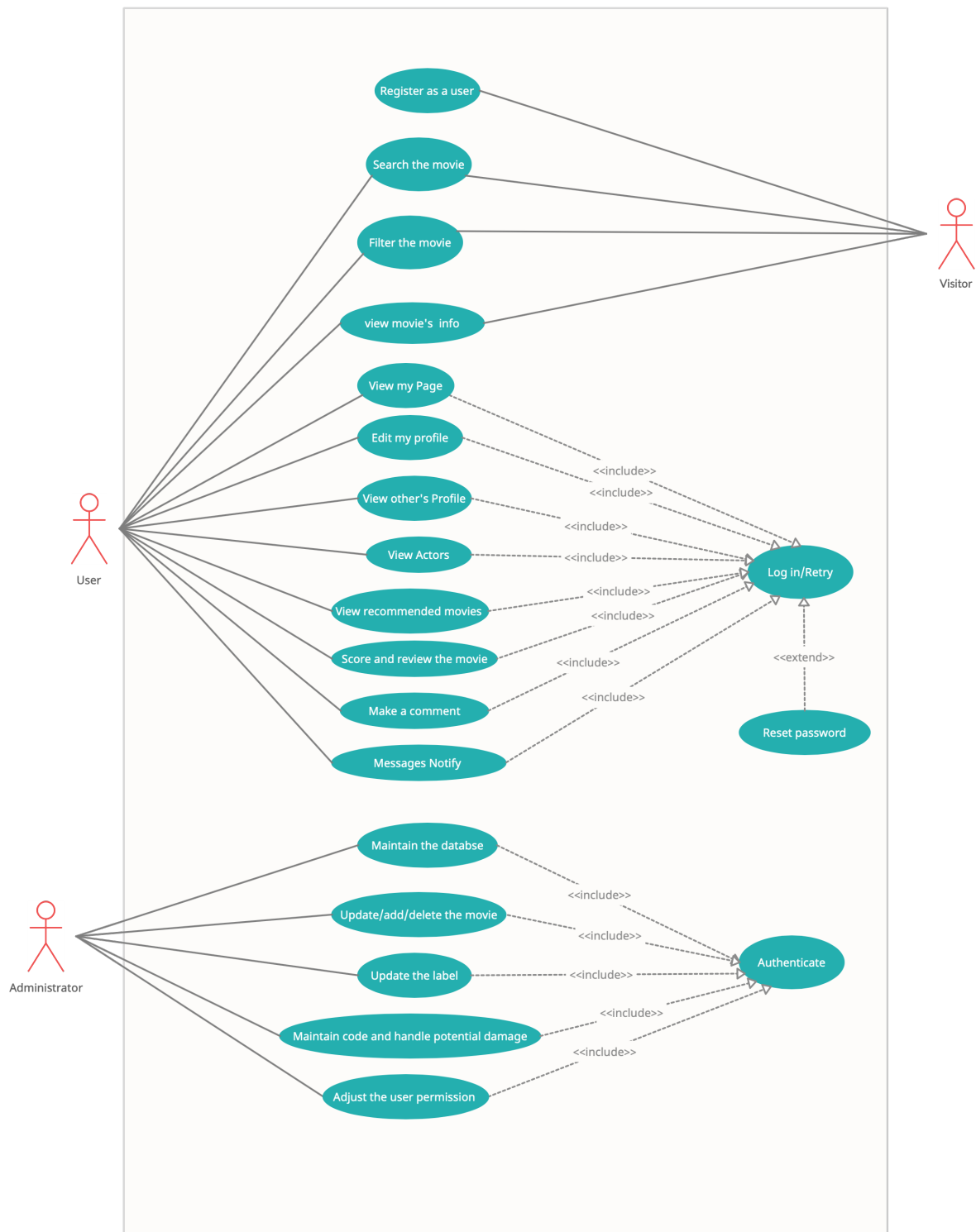


The recommendation system also uses the user-based collaborative filtering algorithm, which can recommends movies that other users with similar interests like, so it is used to recommend movies with high popularity to the user.

The user-based collaborative filtering algorithm mainly includes two steps. The first is to find the collection of users with similar interests to the target user. Then find movies in this collection that users like and that the target user has not heard of and recommend them to the target user.

6 Process Design

6.1 Use Case Diagram



6.2 Use Case Description

ID	UC1
Name	Register as a user
Description	Visitor can register as a user (create an account)
Pre-condition	None
Event flow	1.Visitor clicks Register button. 2.Visitor enter his/her username and password.
Post-condition	A new account created successfully.
Triggers	User clicks "Register" button

ID	UC2
Name	Search the movie
Description	User or Visitor can search a specific movie based on name.
Pre-condition	User click the search button.
Event flow	1.User/Visitor clicks "Search" panel. 2.User/Visitor types name of the movie. 3.Movies will be showed after User/Visitor clicks "Search" button
Post-condition	User/Visitor see a list of movies related to their search
Triggers	User/Visitor clicks "Search" panel

ID	UC3
Name	Filter the movie
Description	User or Visitor can apply a filter to find a range of movies based on Genres, Released periods and Regions etc.
Pre-condition	None
Event flow	1.User/Visitor click 2.User/Visitor click a range of buttons to personalize their search filter . 3.Movies will be showed after User/Visitor clicks "Filter" button
Post-condition	User/Visitor see a list of movies related to their search
Triggers	User/Visitor clicks "Filter" panel

ID	UC4
Name	View movie's info
Description	User or Visitor can view the specific information of a movie
Pre-condition	None
Event flow	1.User/Visitor clicks poster or title of a movie. 2.User/Visitor see lots of information about the movie, include:Director, Writers, Full cast& crew, Storyline, rate, film reviews etc.
Post-condition	User/Visitor see information of the movie they clicked
Triggers	User/Visitor clicks the poster or the title of a movie

ID	UC5
Name	View my Page
Description	User can view his/her own profile, include basic profile (avatar, username, user description, gender etc.) and the data panel of watched list with rates and review)
Pre-condition	User has registered an account and already login.
Event flow	1.User clicks "My Page" button 2.User views his/her own profile
Post-condition	None
Triggers	User clicks "My Page" button

ID	UC6
Name	Edit my profile
Description	User can change his/her profile (avatar, username, user description, gender etc.) and manage watched list and set other user visible or not.
Pre-condition	User is viewing "My page"
Event flow	1.User clicks "Edit Profile" button 2.Change User's personal information 3.The profile will be updated after user clicks "Update" button
Post-condition	User's profile changed
Triggers	User clicks "Edit Profile" button

ID	UC7
Name	View other's Profile
Description	User can view other user's profile (avatar, username, user description, gender, watched list with rates and review)
Pre-condition	User has registered and already login.
Event flow	1.User clicks other user's Avatar or username 2.User views other user's profile (avatar, username, user description, gender, watched list with rates and review) 3.The watched list could be linked to a movie
Post-condition	None
Triggers	User clicks other user's Avatar or username.

ID	UC8
Name	View Actors
Description	User can view Actors' profile (name , gender, date_birth, introduction, photo) and a list of movies they act in.
Pre-condition	User has registered and already login.
Event flow	1.User clicks actors' name 2.User views Actors' profile (name , gender, date_birth, introduction, photo) and a list of movies they act in. 3.The list could be linked to movies.
Post-condition	None
Triggers	User clicks actors' name

ID	UC9
Name	View recommended movies
Description	User can view a list of movies recommended to him/her individually based on their taste.
Pre-condition	User has registered and already login.
Event flow	1.User view a list of recommended movies 2.The list could be linked to a movie
Post-condition	None
Triggers	None

ID	UC10
Name	Score and review the movie
Description	User can give a score and write a review for a specific movie.
Pre-condition	User is viewing a movie's info.
Event flow	1.User clicks "Score and review" button 2.User give a score and write a review. 3.The score and review will be updated after user clicks "Submit " button
Post-condition	A new score and review is added to the movie, and User's profile (watched list) changed.
Triggers	User clicks "Score and review" button

ID	UC11
Name	Make a comment
Description	User can make a comment to other user's review.
Pre-condition	User is viewing a movie's reviews.
Event flow	1.User clicks "Comment" button 2.User give a comment to other user's review 3.The comment will be send after user clicks "Submit " button
Post-condition	User's receive their comments.
Triggers	User clicks "Comment" button

ID	UC12
Name	Messages notify
Description	User can be notified by a new message.
Pre-condition	Other user leave a comment to the movie review Or administrator leave a messages to the user.
Event flow	1.message reminding button present a messages token. 2.User click the messages reminding button. 3.The messages will be list after user clicks "messages reminding" button
Post-condition	messages reminding token disappear.
Triggers	User receive a new Message.

ID	UC13
Name	Login/Retry
Description	User use his/her username and password to login
Pre-condition	User has registered.
Event flow	1.User click Login button 2.User enter his/her username and password to login 3.Login successfully or retry to login
Post-condition	User can do from UC5 to UC12 with his/her account
Triggers	User click "Login" button

ID	UC14
Name	Reset password
Description	User use username and email to reset his/her password
Pre-condition	User has registered.
Event flow	1.User click Reset Password button. 2.User enter his/her username and email 3.Enter the reset code and new password
Post-condition	User can use username and new password to login.
Triggers	User click "Reset password " button

ID	UC15
Name	Maintain the database
Description	Administrator can maintain the database(regular backup management, etc.)
Pre-condition	The administrator account has been verified
Event flow	1.Monitoring database 2.Regular backup 3.Regularly modify important user passwords
Post-condition	The database is normal or even optimized
Includes	Authenticate

ID	UC16
Name	Update the movie
Description	Administrator can update, add and delete the movie
Pre-condition	The administrator updates the databased
Event flow	Use SQL statements to update, insert or delete data
Post-condition	New movies are on the shelves, eliminated movies are deleted
Includes	Authenticate

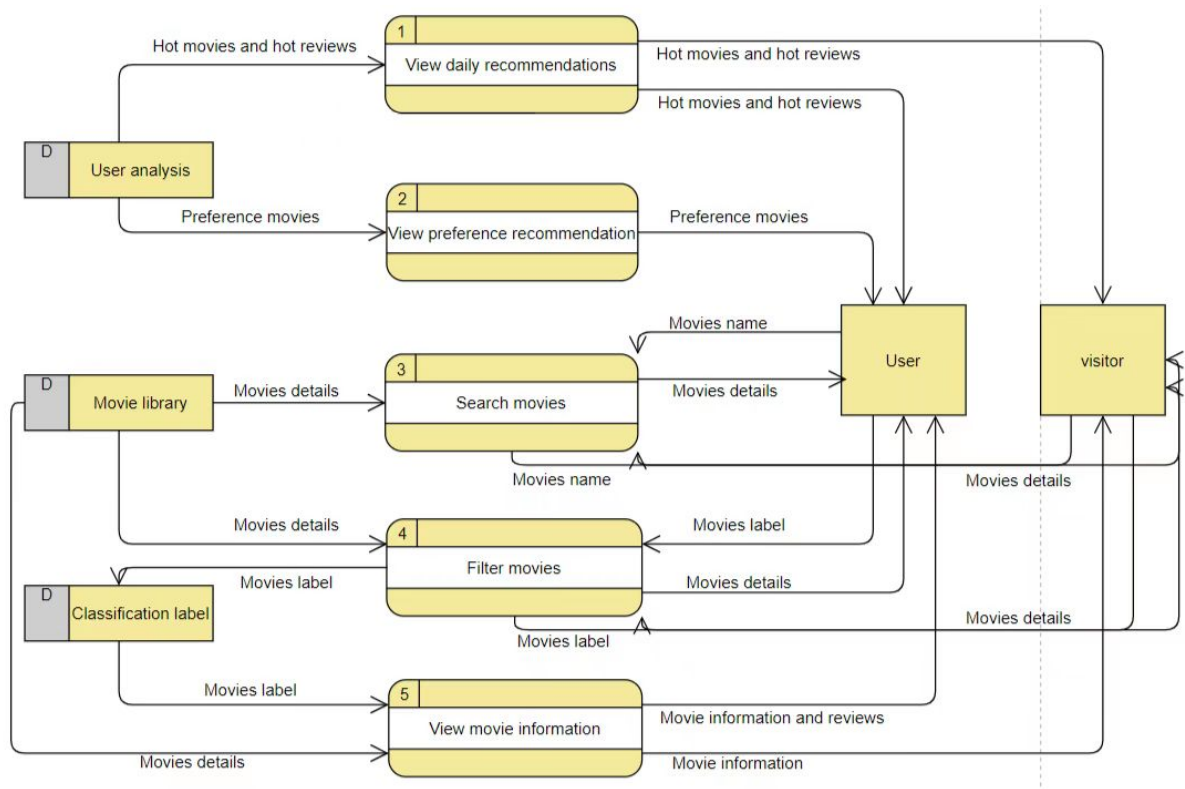
ID	UC17
Name	Update the label
Description	Administrator can update the label in real time
Pre-condition	The administrator updates the database
Event flow	Use SQL statements to update, insert or delete data
Post-condition	Movie classification label update
Includes	Authenticate

ID	UC18
Name	Maintain code and handle potential damage
Description	Administrator can handle potential damage by monitoring the warning log of the database
Pre-condition	The administrator account has been verified
Event flow	Develop a database backup plan to restore database information in the event of a disaster
Post-condition	The database is normal
Includes	Authenticate

ID	UC19
Name	Adjust user's permission
Description	Administrator can increase or decrease the permissions of users and tourists
Pre-condition	The administrator account has been verified
Event flow	Provide different access rights for users
Post-condition	The database is protected from unauthorized access and destruction
Includes	Authenticate

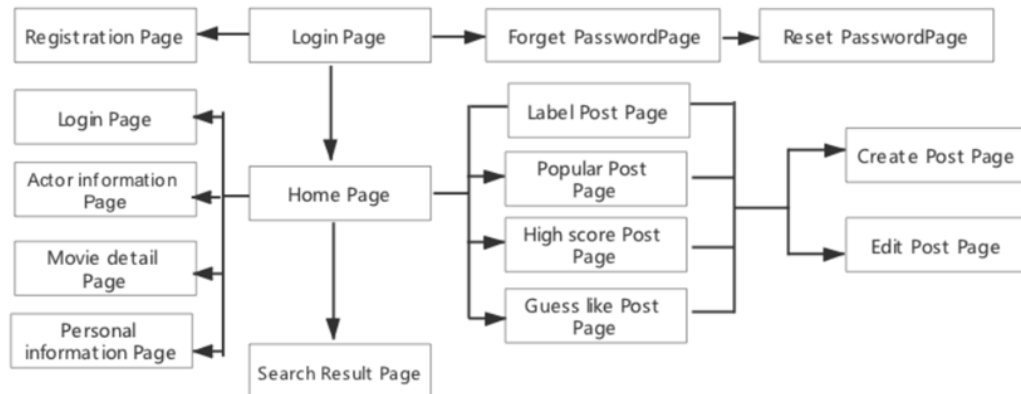
ID	UC20
Name	Authenticate
Description	Check if the administrator inputs are the same as recorded in the database
Pre-condition	Administrator has already registered an account
Post-condition	Administrator account authentication succeeded

6.3 Data Flow Diagram



Data from the input of users and visitors flows to the movie's data repository and tag data repository through search and filtering. The repository feeds movie information back to users and visitors. The collaborative filtering algorithm flows the movie data that users may like to users, and flows popular movies to users and visitors.

6.4 Navigation structure diagrams



In an actual HTML file, the header sections would be the same as the POST page, and they would serve as the navigation bar for the page. On the home page, users can choose to go to each of the four sections. Pages that have a search bar if the user. Enter the keyword in the search bar. In addition, there is a sidebar on the label page, popular page, high score page and guess sample page. The first is the registration page, you can choose to register/login or directly enter the page, you can also go to the registration page, registration or directly login successfully can choose three parts of the page, the first is the home page, the second is the rank page, the third is the personal home page. A search on the home page for a key actor or a detailed movie will lead to two pages. The ranking page includes some key tabs for corresponding ranking, and also summaries the introduction of each movie. Personal homepage includes personal information, history of movies watched. When you register, you need to use a valid email address and a password you set to log in successfully.

6.5 Interface Design

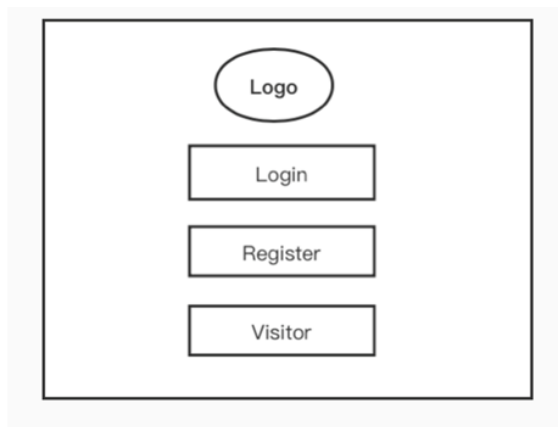


Figure 1 login

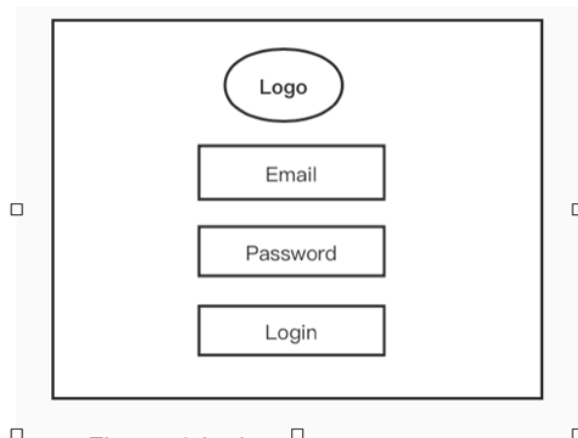


Figure 2 login

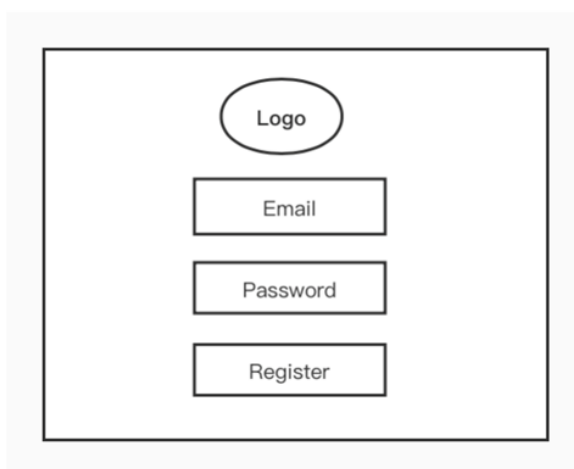


Figure 3 login

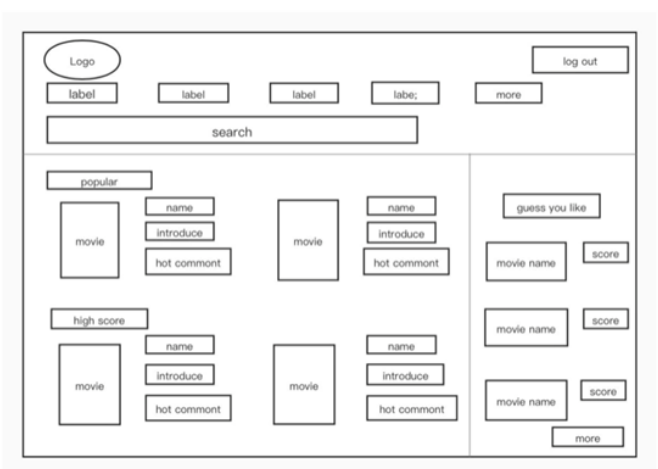


Figure 4 Home page

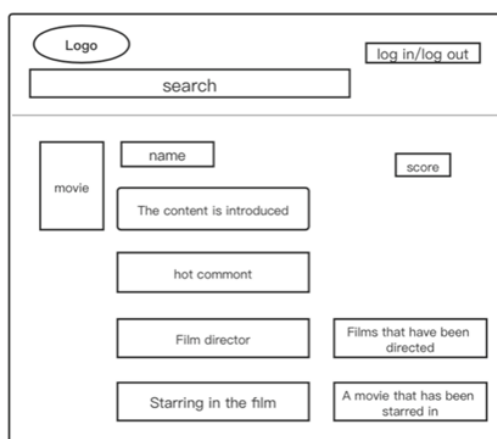


Figure 5 Movie Details Page

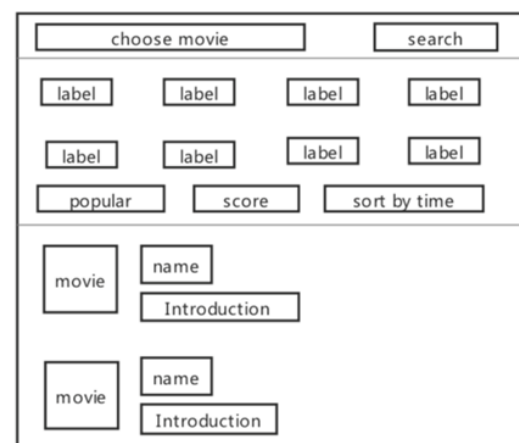


Figure 6 Movie ranking

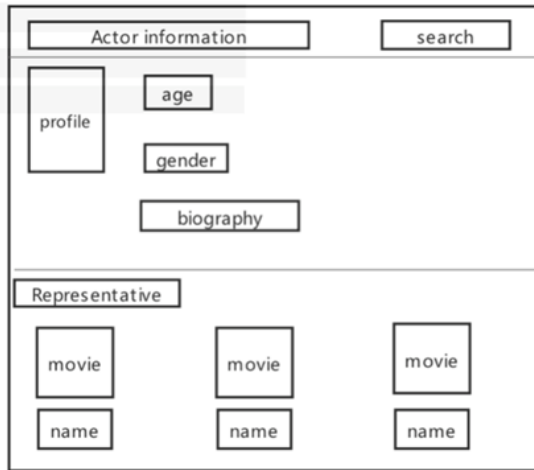


Figure 7 Actor Information

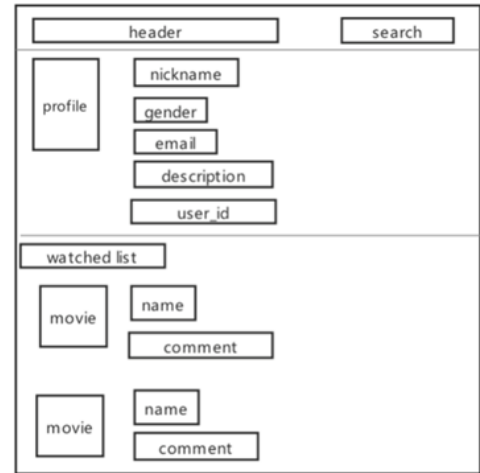
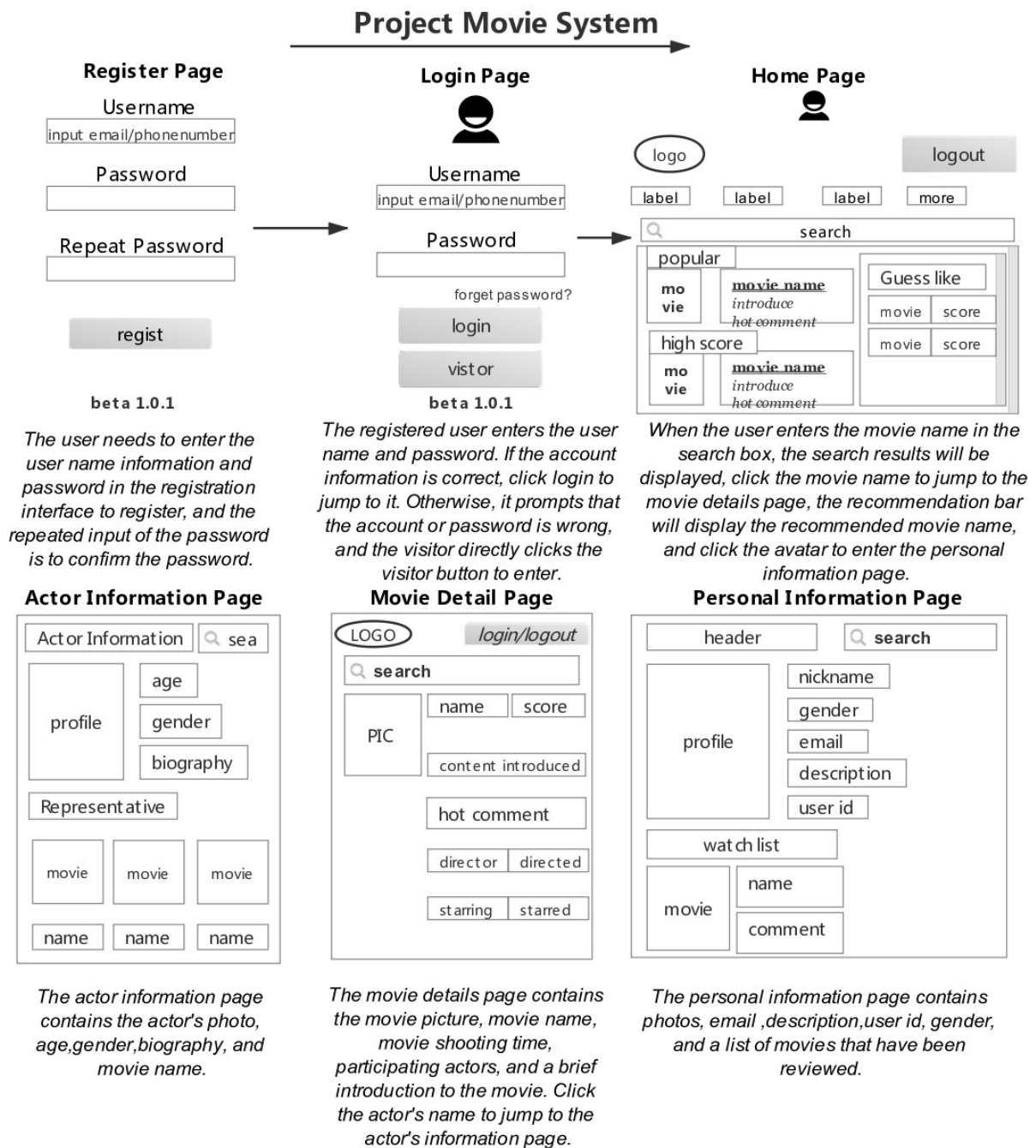


Figure 8 Personal home Page

6.6 Story Board



7 Testing Design

7.1 Testing Strategy

Three phases of testing is required for testing stage: unit testing, integrated testing and system&acceptance testing. A testing document will be used to record the bugs and errors found within the whole implementation stage. In addition, a test report will be delivered, which will include functional testing, performance testing and security testing.

unit testing will cover every individual unit of the system, on how they perform separate from each other. The strategy for the unit testing will be white-box testing. This will allow the testing members to check the code, input the test data and see exactly what's going on and how it's affecting the output.

Integration testing will test how the unit work together as a sub-system. Bottom-up testing is used in this phase. Testing members will test different aspects of sub-systems starting with the lower levels of the system. Integration testing is a key phase of testing since many bugs can be found and early release versions of the overall software or hardware can be developed.

System&acceptance testing will test the whole system. The strategy for system&acceptance testing is black-box testing. Testing team will give test data and cases based on the specification for functional testing and they will use WebPageTest to deliver performance testing and security testing report. Finally, once demo version of our project delivered, an acceptance and satisfaction survey will be made. The group will change settings and develop further functionality based on the survey.

7.2 Test Form

Testing team is aimed at finding bugs and defects within the system and to ensure that it matches the requirements of the project.

Test_ID	Test_Group	Test_Case	Description	Error/Bug	Analysis

Figure 1: Unit Testing form

Function_ID	Test_Function	Description	Input(Behavior)	Actual_Output	Analysis	Success/Fail

Figure 2: Functional testing form

Test a website's performance


Advanced Testing

Simple Testing


Visual Comparison


Traceroute

Test Location

Dulles, VA USA (Desktop, Android, iOS)  [Select from Map](#)

Browser

Chrome 

Advanced Settings 

3 runs, First View only, Cable connection

START TEST

Performance testing and security testing WebPageTest: <https://www.webpagetest.org>

8 Planning and Risk Assessment

8.1 Planning

The basic idea for planning is that, every members could take part in different part of our Project, the multi-cross assignment make our project motivated and enhance the efficient communication between different groups.

We chose to be using Agile methodology for developing software. Agile development make our project has the advantages of adaptive planning, evolutionary development, early delivery, and continual improvement.

The task and code will be developed modularly, which means they are easy to manage and we can work on different task simultaneously.

Stage	ID	Task	Start Date	End Date	Taskmaster	Duration
Specification	1	Proposal and Background Searching	7-Feb	12-Feb	Everyone	6
	2	Decide A Project idea	9-Feb	12-Feb	Everyone	4
	3	Requirement Analysis	13-Feb	18-Feb	Everyone	6
	4	Requirement Documentation	16-Feb	22-Feb	Everyone	7
Design	5	Architecture Design and Documentation	23-Feb	2-Mar	Everyone	7
	6	Interface Design	25-Feb	4-Mar	Everyone	2
	7	Front-end Web Design	5-Mar	15-Mar	QianQian,HengYu	11
	8	Back-end Web Design	5-Mar	15-Mar	ZiYang,Hang	11
	9	Database Design	5-Mar	15-Mar	YiBing,Wuwei	11
	10	Recommendation Algorithm Design	5-Mar	15-Mar	Wuwei,ZiYang	11
	11	Testing Plan and Design	9-Mar	15-Mar	Hang,Yibing	7
	12	Design Review	16-Mar	19-Mar	Everyone	4
Implementation	13	Plan Review	20-Mar	20-Mar	Everyone	1
	14	Front-end Web Implementation	21-Mar	21-Apr	QianQian,HengYu	31
	15	Back-end Web Implementation	21-Mar	21-Apr	ZiYang,Hang	31
	16	Database Implementation	21-Mar	21-Apr	YiBing,Wuwei	31
	17	Recommendation Algorithm Implementation	21-Mar	21-Apr	Wuwei,ZiYang	31
Test	18	Unit Testing	21-Mar	21-Apr	Hang,Yibing	31
	19	Integrated Testing	21-Mar	21-Apr	Hang,Yibing	31
	20	System & Acceptance Testing	22-Apr	28-Apr	Everyone	6
Submission	21	Demo Documentation	22-Apr	4-May	Everyone	12
	22	Portfolio Submission	5-May	11-May	Everyone	7
	23	Individual Submission	5-May	14-May	Everyone	10

8.2 Required Skills

- Git command and Github is considered as an important skill, which will improve the efficiency of coding as a group work.
- PHP, JavaScript , MySQL for back-end implementation. • HTML, CSS for front-end Implementation.
- Python is used to build recommendation algorithm (Item based Collaborative Filtering Algorithm etc.).
- Except White-box testing, black-box testing, WebPageTest will be used for Functional and performance testing.
- Communication skill is required, especially for the communication between the back-end group and front-end group.

8.3 Challenge

8.3.1 General Challenge

- Due to the pandemic virus, two of team members are outside the UK, so it could be a challenge for team members to host efficient communications online.
- Team members should learn related coding environment at the beginning of implementation, they also have to get to know how each part cooperation.
- Sets of new skill required, teams must learn and Implement over very short periods of time.

8.3.2 Technological Challenge

- For collaborative filtering algorithm, as the number of users continues to increase, "nearest neighbor search" within a large number of users will become the bottleneck of the entire algorithm, and the amount of calculation is large.
- It's a challenge transfer XML data from TMDb data to SQL format.
- It's hard for Front-end design to deliver a user-friendly interface.

9 References

- [1] Hameed, M.A, O. Al Jadaan and S. Ramachandram, Collaborative filtering based recommendation system: A survey. International Journal on Computer Science and Engineering, 4(5), 859, 2012.
- [2] G. Author, "8 Major Advantages of Using MySQL", Datamation.com, 2016. [Online]. Available: <https://www.datamation.com/storage/8-major-advantages-of-using-mysql.html>.
- [3] "The Movie Database (TMDb) API", TMDb, 2021. [Online] Available: <https://developers.themoviedb.org/3/getting-started/introduction> [Accessed: March-2021].
- [4] "The Movie Dataset", Kaggle, 2021. [Online] Available: <https://www.kaggle.com/rounakbanik/the-movies-dataset> [Accessed: March-2021]
- [5] "TMDB 5000 Movies Dataset", Kaggle, 2021.[Online] Available: <https://www.kaggle.com/tmdb/tmdb-movie-metadata> [Accessed: March-2021]