

Final Reflective Report for Software Engineering Group Project

2021/2022 Semester 2

Leisure Town

2022.5.22

Lincheng Shi

1927978

A-14

Table of Contents

Introduction	3
System Design	4
System Overview	4
System Architecture.....	8
High-level Database Design	10
Functional Design	11
Individual Work.....	13
Overview of Previous	13
Sprint 4 Overview	13
Personal Contribution to Sprint 4	14
Sprint 5 Overview	14
Personal Contribution to Sprint 5	14
Software Testing	14
Unit Test.....	14
Acceptance Test.....	15
Change Management	15
Teamwork.....	16
Challenges of The Teamwork.....	17
Legal, Social, Ethical, and Professional Issues.....	17
Conclusion.....	17
Lesson Learnt and Future Work.....	18
Appendix.....	19

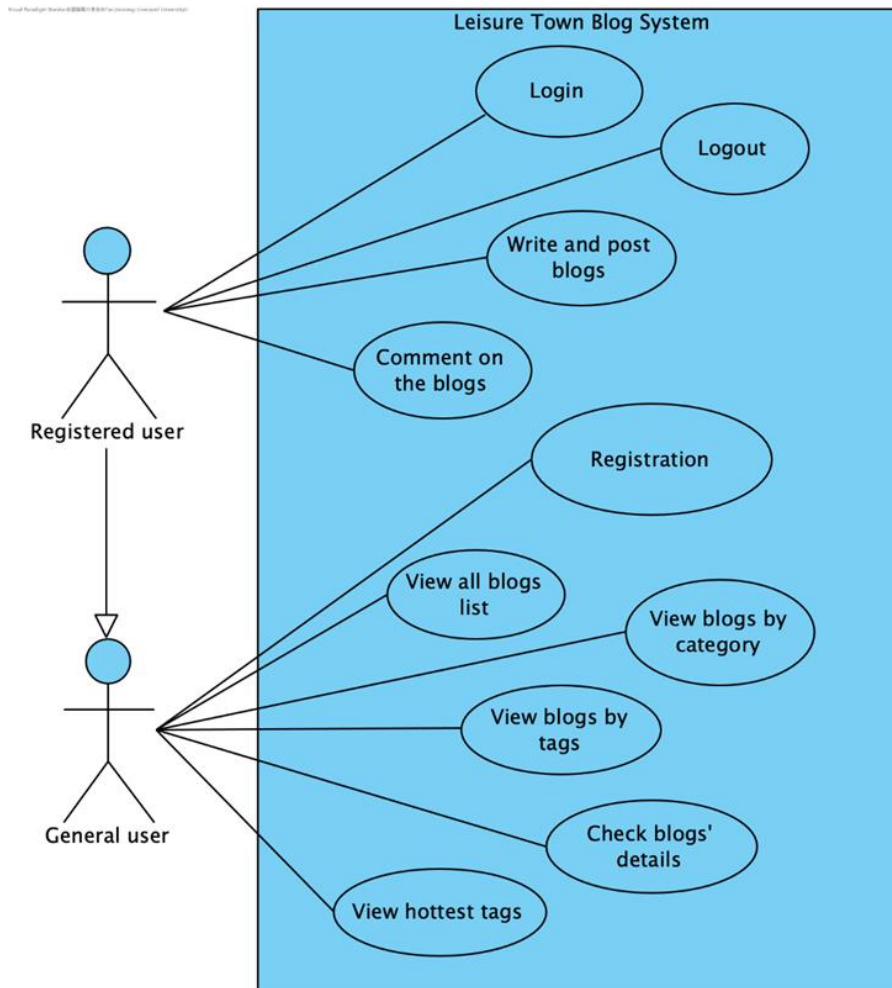
Introduction

Our software requirements can be divided into two parts: requirements on blogs and requirements on accounts. Initially, requirements on blogs include #1.1 viewing blogs, #1.2 expressing users' own ideas (forwarding, thumbs up and comments) on the blogs, #1.3 writing and posting blogs to others, #1.4 searching for blogs. The requirements on accounts include #2.1 registering the account, #2.2 logging in the account, #2.3 managing personal information. With the progress of the project, the requirements changed a lot. We added details to #1.1, that is, we can view blogs in different ways (e.g. categories, tags, timeline). When developing, we added categories and tags to each blog so that users can read the blogs they are interested in and think it's messy to list only blogs. Therefore, we determined how to view these blogs. We also deleted the forwarding in #1.2 because the focus of our products is not on commercial value or chasing others, but on relaxation. Considering that there is too much white space on the main page and we want to enrich functions, we also added a demand: check the hottest tags and fashion articles. In addition, we also added exit login in #2.2 to ensure the security of the account.

Some software requires users to add friends to view and comment on their blogs, so we set the home page as "town" and all blogs in the town are public. I think this can not only echo our software names "leisure" and "town", but also solve the above problem. "Town" is a place where everyone can come and make people feel relaxed. Users can view and share blogs here without worry. What's not so good about this page is that the blog doesn't have pagination and uses scroll bars instead. Some software. Users can't post blogs according to category and tags, so we set up category and tags about life for users to choose. Users can only select one category, but can select multiple tags. I think this will enable user with the same interest to find relevant articles at a glance. However, what needs to be improved is that after users select a category, they should only provide tags under the corresponding category instead of all tags. Users can't view the most popular tags, so we set the hottest tag. I think it's very convenient for users to understand most people's preferences. By clicking on the tag, you can also view all the articles under the tag. Users can't view all the articles under their favorite categories and tags, so we added these two pages. I think the hand drawn icons on these two pages make our software no longer monotonous. Although our software advocates simplicity, it can't make users feel boring. Users can't choose some functions to post blogs, so we added a markdown editor. I think the rich functions make users' articles more vivid and interesting. In particular, the immersive reading function greatly increases the user experience. Users can't read the rough ideas of bloggers, so we set up the blog summary section. I think this not only enriches the home page, but also allows users to clearly understand the content of the article.

System Design

System Overview



Use Case UC1: Login

Primary Actor: Registered user

Preconditions: The user has an account

Post conditions: Functions of writing blogs, posting blogs and comment on the blogs can be used

Main Successful Scenario:

1. Registered user enters the website to the home page
2. Registered user clicks the "Login" button on the navigation bar
3. The page switches and a login box with two input boxes of user name and password is displayed
4. Registered user inputs user name and password
5. Registered user clicks the "Login" button and login successfully
6. The web page skips from login page to the home page
7. The original place of "Login" is replaced by the avatar of users

Exception Scenarios:

1. If the username or password is incorrect, a message "username or password is not existed" is displayed, and the login cannot be successful.

2. If the user clicks login button without username, a message "Please enter the username" is displayed and the login cannot be successful.
3. If the user clicks login button without password, a message "Please enter the password" is displayed and the login cannot be successful.
4. If the user enters a user name or password with more than 10 characters, a message "Cannot be greater than 10 characters" is displayed and the login cannot be successful.

Use Case UC2: Log out

Primary Actor: Registered user

Preconditions: Registered user has logged in with an account

Post conditions: Registered user logs out the system

Main Successful Scenario:

1. Registered user enters the home page
2. Registered user clicks their avatar of user on the navigation bar
3. A drop-down list with item writing "Log out" is displayed from the avatar of user
4. Registered user clicks "Log out" item
5. Registered user logs out successfully

Use Case UC3: Write and post blogs

Primary Actor: Registered user

Preconditions: The user who has registered and logged in clicked the "Write an article" button at the top of Leisure Town's home page to write a new blog with the editor. Finishing editing the blog, the user clicked the "Post" button to post the blog

Post conditions: A new article with title, author, time, article content, article summary, article category and article tags will be published successfully in Leisure Town

Main Successful Scenario:

1. Registered user logs in with their account
2. Registered user clicks the "Publish an article" button of Leisure Town navigation bar in main page to switch the page to write.
3. Registered user writes the title of the article and writes the content of the article, and user can use the editor to complete their blogs.
4. Registered user can according to their own requirements to beautify or design layout, including content in bold, italics, different level title, underline, marking, tag, the Angle, the Angle of standard, paragraph references, ordered list, unordered list, add images, the code block, open title navigation, preview, full-screen editor, immersive reading, markdown syntax help (can add tables, formulas, etc.), step up, step down, empty.
5. Registered user clicks the "publish" button, switch to a pop-up window, write a summary of the article, select the article category, check the article tags.
6. Registered user clicks the "publish" button, the article is published successfully and a pop-up window shows that "Published successfully".

Exception Scenarios:

1. If the title is more than 20 characters long, an error message is displayed and the publication cannot be successful.

2. If the title or article content is not written yet, directly click press conference prompt "title cannot be empty", "content cannot be empty".
3. If the article summary or article category or tags is not filled before, directly press conference prompt "please input summary" or "please select article category" or "please select tags", summary cannot be more than 80 characters
4. Click "Cancel publication" on the page of writing an article and a prompt box will appear. Select "Cancel" or "x" to go back to the page of write post and select "OK" to enter the main page of Leisure Town. The system will restore the article, summary, tag and categories before user cancel the publication.

Use Case UC4: Comment on the blogs

Primary Actor: Registered user

Preconditions: The user who has registered and logged in can check blog's details

Post conditions: A new comment is added to the blog successfully, and user can see his/her comment after reloading the page

Main Successful Scenario:

1. Registered user logs in with an account
2. Registered user clicks a blogs and check blog's detail
3. Registered user enters some comments in the box which below the blog's content.
4. Registered user clicks "Post" button
5. A comment is added to the blog successfully

Exception Scenarios:

1. If the user name has not login and post a comment, a message "Please login" is displayed and the comment posting cannot be successful.

Use Case UC5: Check blog's detail

Primary Actor: General user

Preconditions: The user has clicked Leisure Town's home page.

Post conditions: All of the details of this blog have been shown on the blog's page.

Main Successful Scenario:

1. General user enters website turns to the main page
2. General user clicks any article title to view the details of the blog, the page to jump to the blog page.
3. The blog page displays the title, content, date of publication, author profile picture, number of views, tags used, category of posts and comments.

Use Case UC6: View blogs by tags

Primary Actor: General user

Preconditions: There are blogs with tags by users, and they can be classified according to tags. And one blog will be post with at least one tag. The user can click the "town tag" button.

Post conditions: The blogs with the same tag are put together and displayed in time order.

Main Successful Scenario:

1. General user enters the website to the home page
2. General user clicks "town tag" button which is in the navigation bar of the home page

3. All tags of blogs are displayed in the screen
4. General user selects one tag
5. All blogs with the same tag are displayed in publication time order

Use Case UC7: View hottest tags

Primary Actor: General user

Preconditions: The user has clicked Leisure Town's home page and some blogs have been published.

Post conditions: Display the six hottest tags in the database on the main page.

Main Successful Scenario:

1. Users, both registered and unregistered, clicked on the Leisure Town home page.
2. On the right side of the home page, there is a module that displays the six most popular tags already in the database, based on the number of tags used for articles.
3. Click "View All" button to go to the Town's Tag page and view all the tags.

Use Case UC8: View blogs by category

Primary Actor: General user

Preconditions: There are tags classified by registered users. There are blogs post by registered users. There is at least one blog with only one category post by registered users.

Post conditions: The blogs with the same categories are put together and displayed in time order.

Main Successful Scenario:

1. General user enters the website to the home page
2. General user clicks "town category" button which is in the navigation bar of the home page
3. All categories of blogs are displayed in the screen
4. General user selects one category
5. All blogs with the same tag are displayed in publication time order

Use Case UC9: View all blogs list

Primary Actor: General user

Preconditions: General user and registered user enter the website. There is at least one blog post by registered users.

Main Successful Scenario:

1. General user enters the website to the home page
2. The website display all blogs lists in home page

Use Case UC10: Registration

Primary Actor: General user

Preconditions: General user has not registered before.

Post conditions: General users can login the website with account and become the registered users.

Main Successful Scenario:

1. General user enters the website to the home page
2. General user clicks the "Registration" button on the navigation bar

3. The page switches and a login box with three input boxes of user name, nick name and password is displayed
4. General user inputs user name, nick name and password
5. General user clicks the "Registration" button and register successfully
6. The web page skips from login page to the home page

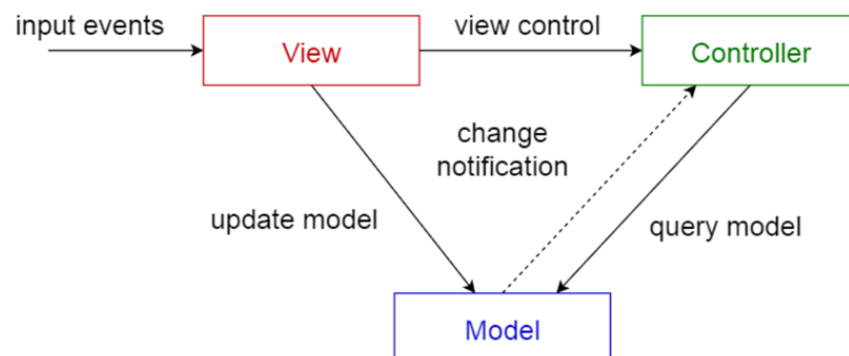
Exception scenario:

1. If the username already exists, a red message "Account already exist" is pop up on the top of the page and the registration is not successful
2. If the user cannot input the username, nick name and password all to the input box, the red message "Please enter username" "Please enter nick name" "Please enter password" will be displayed below each input box and the registration is not successful
3. If the user enters a user name or nickname or password with more than 10 characters, a message "Cannot be greater than 10 characters" is displayed and the registration cannot be successful.

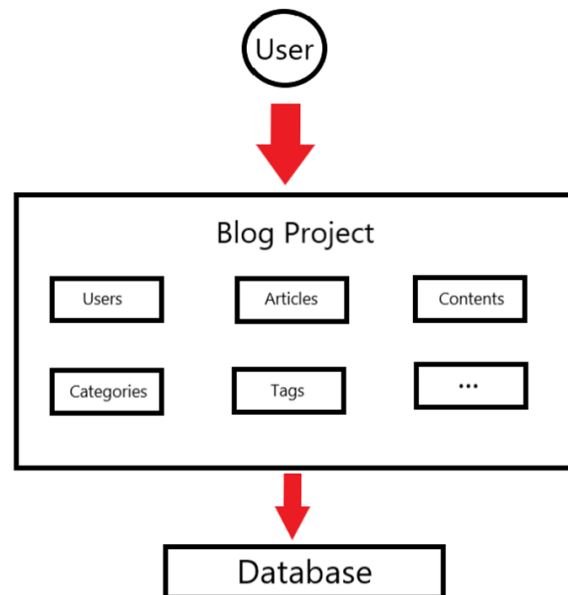
I think there are some things that are done well and some things that are done badly about the system overview. The good thing is that the use cases are listed clearly, which well reflects what users who are logged in and not logged in can do respectively. There is also a generalization of the relationship between the primary actor. This can reflect that the logged in users can do all the things that the not logged in users can do. The bad thing is that the relationship between use cases is not clear. For example, there is a use case called "write and post blogs". But in fact, we must write a blog before we can enter the publish dialog box. Therefore, I think it can be expressed here by publishing a blog, including writing a blog. Another example is "check blogs' details" can extend "view all blogs list", "view blogs by category" and "view blogs by tags". Because when we view these blogs list in different order, we can choose to click the title to see the blog details, or we can choose not to click the title. In addition, I think the use cases of "search blog by title" and "trendy blogs" can be added in the general user section. These two use cases are done in sprint 4.

System Architecture

In this project, our overall framework is MVC (Model-View-Controller) framework. "M" stands for Model. Models represent business rules. "V" stands for View. A View is the interface that the user sees and interacts with. "C" stands for Controller. A controller is a controller that accepts input from the user and invokes models and views to fulfill the user's requirements.



MVC is a monomer architecture as well as a back-end separation architecture. Thus, we use different architectural techniques on the front end and the back end.



The main front end we use is vue.js framework. Vue is a set of progressive JavaScript frameworks for building user interfaces. Unlike other large frameworks, Vue is designed to be applied layer by layer from the bottom up. Vue's core library focuses only on the view layer, making it easy to get started and integrate with third-party libraries or existing projects. On the other hand, when combined with modern toolchains and various supporting libraries, Vue is perfectly capable of providing drivers for complex single-page applications (SPA).

The backend we use is mainly composed of SpringBoot + Mybatisplus + Redis +MySQL framework.

SpringBoot: SpringBoot integrates common frameworks such as Mybatis+ SpringMVC, eliminating complex configuration.

Mybatisplus: Mybatisplus is a MyBatis enhancement tool, designed to enhance MyBatis on the basis of only do not change, to simplify development, improve efficiency

Redis: Redis supports data persistence, saving data in memory to disk, which can be reloaded for use upon restart. Redis not only supports simple key-value type data, but also provides the storage of list, set, zset, hash and other data structures.

MySQL: Back-end storage of various data needs to design and use the database; MySQL is an open-source relational database management system.

I think our system architecture is MVC architecture, which is reasonable. Because the front end and the back end are separated. At the same time, it is also a single architecture. Because users

can directly access our project, after our project is developed into each module, we can directly access the database. We make everything into a web project, package it into a jar package and integrate it on Tomcat.

High-level Database Design

Tables: Article, Article_body, Article_tag, Tag, Category, Comment, Permission, Sys_log, Sys_User.

Details of the database:

Article contains article information: Id, Create_Time, Comment count, view count, title, etc

Article_body: Contain the main body of the article.

Article_tag: Connect Table Article and Table Tag.

Tag: Article tags.

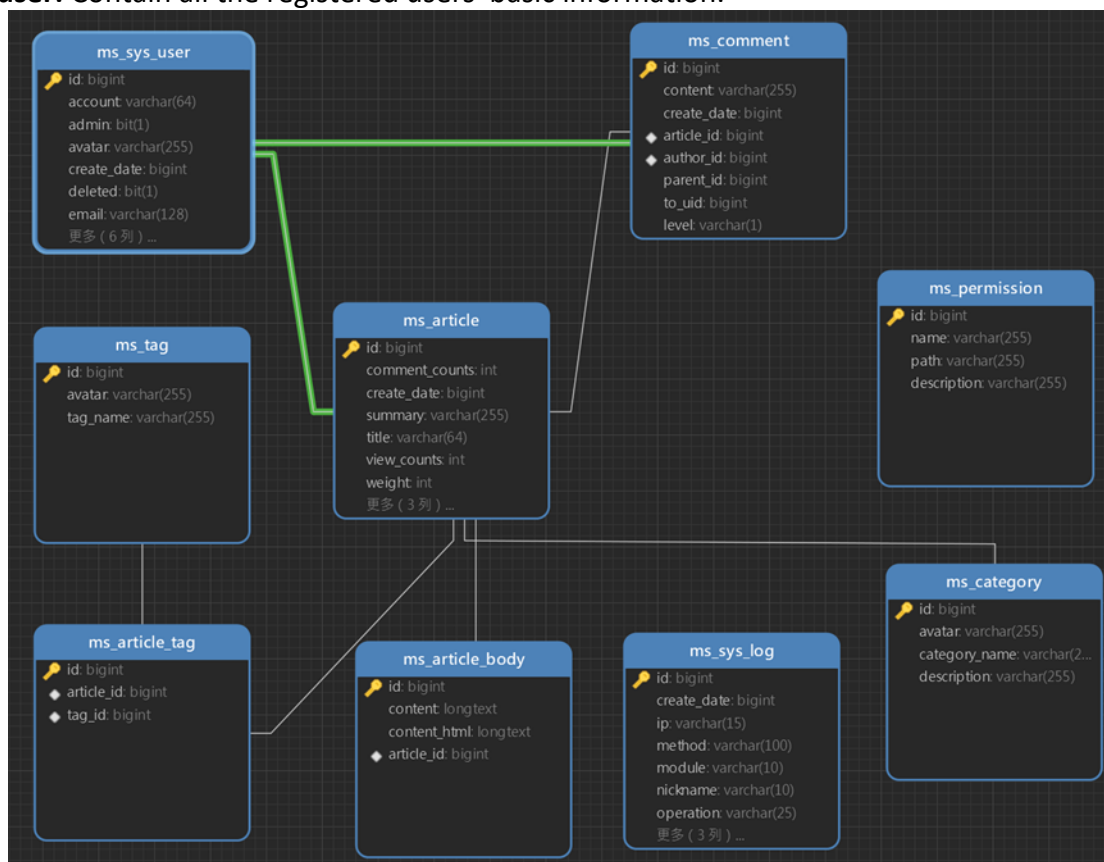
Category: Article categories.

Comment: Contain all comment information.

Permission: User's rights on the app.

Sys_log: Record users' operations on the app. (Just create the table. It will not be used until the app is certainly open to the users.)

Sys_user: Contain all the registered users' basic information.



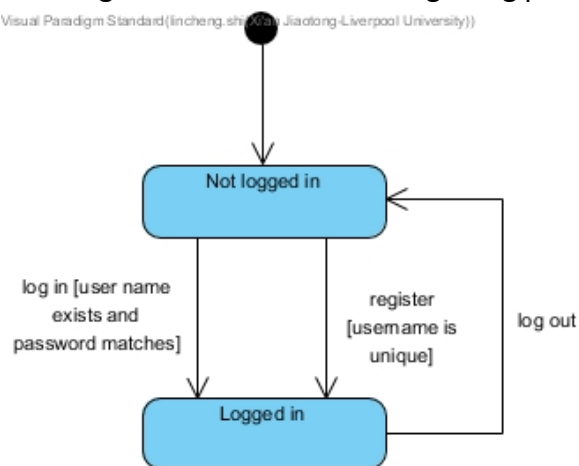
I think the design of this database is very good on the whole. Taking the “ms_article” as the primary table, the association tables of “ms_category”, “ms_article_body”, “ms_article_tag”, “ms_tag”, “ms_sys_user” and “ms_comment” are established. The main contents of these

association tables will be displayed in different pages. For example, “avatar” and “tag_name” will be displayed in “town tags” page. The “ms_sys_user” table is not only the author information of the blog, but also the information of registered users’ information will be created in this table. Considering the avatar and nickname of relevant users will be displayed when users make a comment, “ms_sys_user” table and “ms_comment” table is also associated. In addition, two independent tables are designed: “ms_permission” and “ms_sys_log”. Initially, these two tables are designed for future work, so they are still empty in MySQL at present. One is used to set the administrator user. The administrator user can use some permissions, such as setting the top. The other is used to record login information. However, due to limited time and capacity, we may not develop these two lower priority functions.

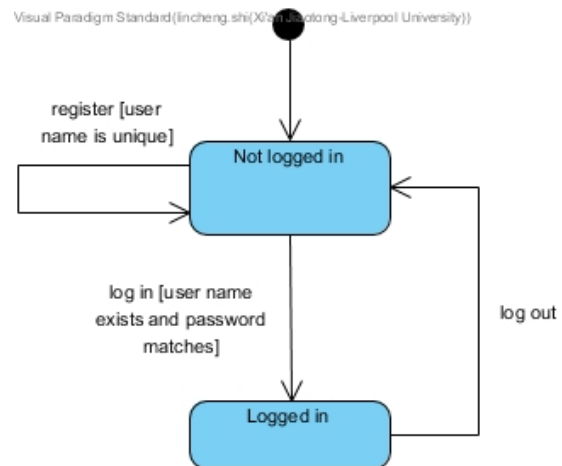
Functional Design

“Login and registration” is one of the core functions of our system. When users enter the website, they are not logged in. Users can enter the logged in state by registering an account or logging in to an existing account. At the same time, in the login status, users can click exit login to return to the non login status. The state machine diagram is shown in the left figure below. I have some ideas about the design of this core function. I think that when a user is not logged in, he is still not logged in through registration. To reach the logged in status, users must log in. The state machine diagram of login and registration in my idea is shown in the right figure below. This is mainly to deepen the user's memory of user name and password. After all, our members often forget the account and password they created when trying the system. In addition, I think there should be a back to home button on the login and registration pages. Otherwise, it is too troublesome to rely on right clicking in the appropriate position to select the return button. I also think login lacks a function of forgetting password.

Visual Paradigm Standard (lincheng.shi@xjtu.edu.cn, Jiaotong-Liverpool University)

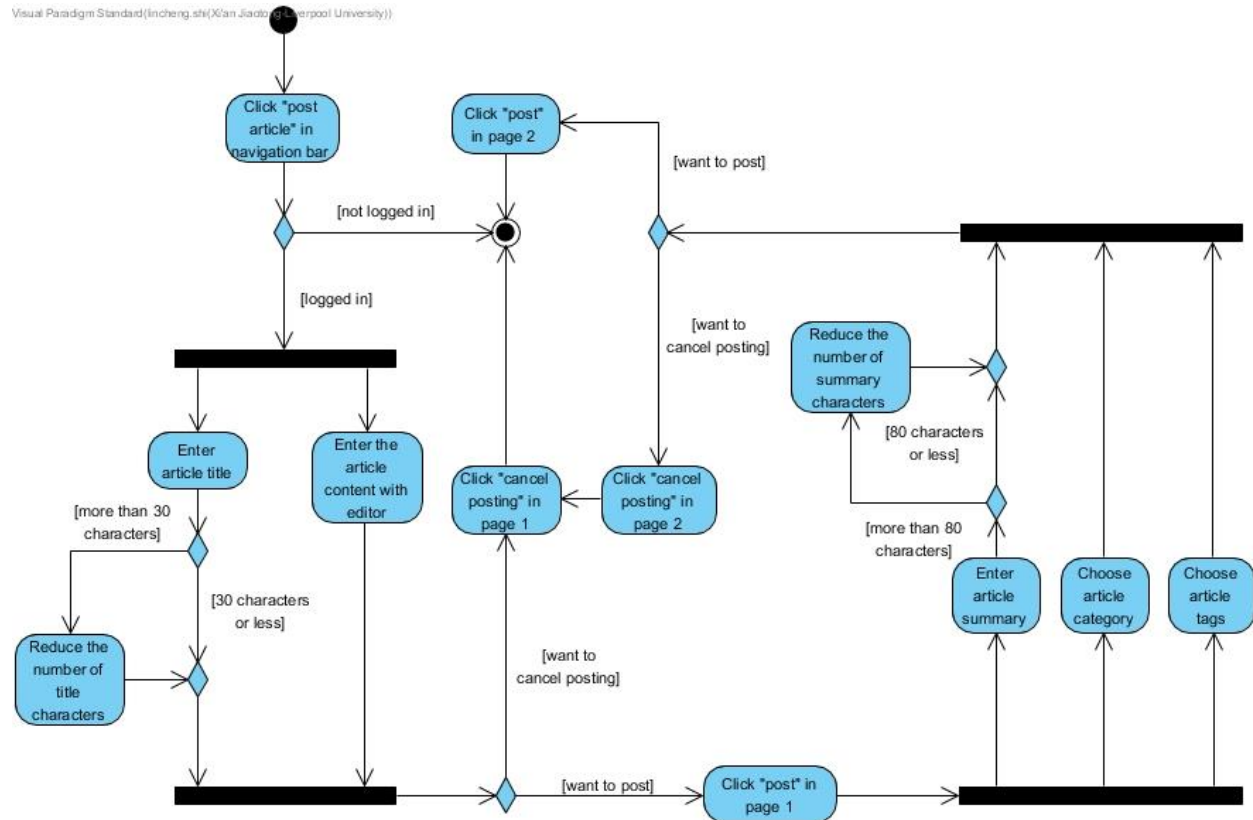


Visual Paradigm Standard (lincheng.shi@xjtu.edu.cn, Jiaotong-Liverpool University)

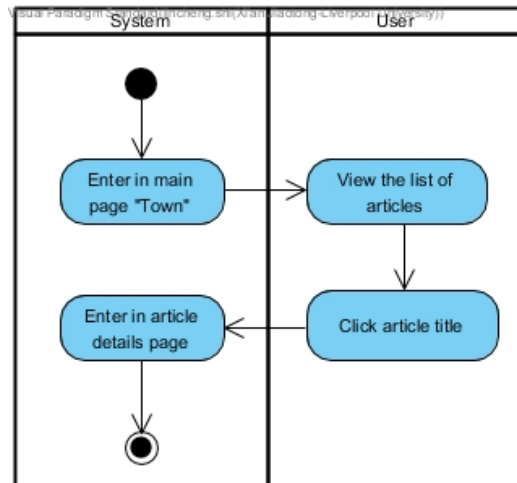


“Post articles” is also one of the core functions of our system. This function can only be used when the user is logged in. In the login status, users need to enter the article content and a title with less than 30 characters after they click "post article" in navigation bar. When entering article content, users can use markdown editor functions such as bold font, subtitle, immersive reading. The user can then choose to cancel posting or post the article. If you choose to post an article, a dialog box will pop up. After users fill in the article summary with less than 80 characters, article category and article tags, they still have two choices. One is to close the dialog box, the other is

to post the article. If users choose to post articles, they will successfully post articles to the home page. The activity diagram of this core function is shown below. The object of this activity is the user. I think the design of this core function is very comprehensive. Markdown editor is a good proof. It is also a good choice to divide the article title, article content and article summary, category and tags into two pages. In this way, users can clearly see the main part of the article and have a clear hierarchy. I also have another idea: merge all the contents to be written on one page. Completing all the contents on one page can save some trouble, which also corresponds to the theme of our product "leisure". The activity diagram of this idea can be seen in the appendix 3.



"View articles" is another core function. The system should first enter the home page "town". Then users can view the list of articles. Then the user clicks the title of the article he wants to see, and the system will enter the article details page. The activity diagram of this core function is shown below and the objects include system and users. In fact, users can view articles not only by clicking the article title on the home page, but also by clicking the title of trendy articles on the right side of the home page. In addition, users can also click the article title classified in the town article category and town tags to view article. I think the only deficiency of the design of this core function is the lack of return button. Users can only return to the home page by clicking "town" in the navigation bar. It's not very comfortable for me to use. I want to delete the navigation bar of the article details page and change it to the back page button. In general, the design of this core function is simple, clear and easy to operate.



Individual Work

Overview of Previous Works

From Sprint 1 to sprint 3, my main task is back-end development. In addition, I also did many tests, fixed the bugs in my tasks and part of front-end bugs, and made some modifications to the contents of the database. I combined some of my contribution in same bullet points, details are as follows:

- Learn basic knowledge
- Make the front-end page of login and registration, including jump function
- Write back-end code for article list and article details
- Create part of database tables related to the article list and add an account and an article in database
- Test almost all functions and list them
- Fix all bugs in my own tasks and author disappeared bug in front-end

Sprint 4 Overview

In sprint 4, our aims and objectives are as follows:

1. fix all bugs in the first three sprints
2. complete all 4 PBIs
3. find out where to optimize in future

We have four PBIs in this sprint. They are #12 Trendy blogs, #13 Tag-classified blog list, #14 Category-classified blog list and #15 Search by blog title. Other details can be seen in the appendix 4.

#12 can be divided into 6 tasks: #12.1 UI design, #12.2 Front-end script, #12.3 Front-end api, #12.4 Front-end router, #12.5 Back-end api coding + database initialization, #12.6 Testing. #13 can be divided into 9 tasks: #13.1 UI – page, #13.2 UI – components, #13.3 Front-end script, #13.4 Front-end API, #13.5 Front-end router, #13.6 Back-end api coding, #13.7 Database initialization of icon, #13.8 Back-end api coding - access the data, #13.9 Testing. #14 can be divided into 9 tasks:

#14.1 UI – page, #14.2 UI - components of page, #14.3 Front-end script, # 14.4 Front-end API, #14.5 Front-end router, #14.6 Back-end api coding, #14.7 Database initialization of icon, #14.8 Back-end api coding - access the data, #14.9 Testing. #15 can be divided into 6 tasks: #15.1 UI – page, #15.2 Front-end script, #15.3 Front-end API, #15.4 Front-end router, #15.5 Back-end API coding, #15.6 Testing.

Personal Contribution to Sprint 4

In sprint 4, I mainly completed three things. I wrote the back-end code of list of article category and tags, insert the drawn picture in the front end and database, test functions in this sprint. In back-end code, interface url is “/categorys/detail” and “/tags/detail”, request method is “get”, request parameter is nothing. It returns json data. The general process is similar to that in sprint 2. This function is actually similar to the principle of displaying tags and categories in the PBI#9 Blog full text. Take the category as an example. In #9, the annotation for “categories()” method is @GetMapping in CategoryController, “findAll()” method only selects two fields “Id” and “CategoryName” in “CategoryServiceImpl”. But now, the annotation for “categoriesDetail()” method is @GetMapping(“detail”) in CategoryController, “findAllDetail()” method selects all fields in “CategoryServiceImpl”. In short, system accesses all data in “ms_category” table. Inserting the drawn picture is an easy thing. I just find the right folder and insert it. I do acceptance test by clicking buttons one by one.

In this sprint, my task was completed quickly, because the functions of this part are similar to those I wrote earlier to some extent. I also helped the team members solve some problems about trendy blogs. For example, we can sort trendy articles by reading times. After completing the test, I summarized some places where our software can be optimized and actively communicated with the team members. The bad thing is that everyone feels that this sprint is not as formal as before, so we all type and discuss in WeChat group instead of having a meeting.

Sprint 5 Overview

We have implemented the basic functions in the first four sprints. Therefore, we focus on individual reports instead of doing sprint 5.

Personal Contribution to Sprint 5

We have implemented the basic functions in the first four sprints. Therefore, we focus on individual reports instead of doing sprint 5.

Software Testing

Unit Test

I mainly do unit tests through postman, modifying parameters and viewing running results in console. In sprint 1, my task is mainly to realize the jump function, so my unit test only needs to observe whether the jump between two pages can be successful. In sprint 2, my task is mainly the back-end code of the article list. In postman, my interface url is “http://localhost:8888/articles”, request method is “post”, request parameters(json) in body-

raw are "page": 1, "pageSize": 10. It returns json data (all the correct return results can be seen in appendix 5). In sprint 3, my task is mainly the back-end code of the article detail. In postman, my interface url is "http://localhost:8888/articles/view/{id}", request method is "post", request parameter is "id(long)" which means article id(path parameter). I give an example results for id=1001 in appendix 5. In sprint 4, my task is mainly the back-end code of categories and tags list pages. In postman, my interface url is "http://localhost:8888/categorys/detail" and "http://localhost:8888/tags/detail", request method is "get". Therefore, there are no request parameter.

When doing unit test in sprint 3, I spend too much time in postman. Because I don't know where the path parameter "id" should be put. At first, I wrote the id in the form of {"id": 1001} in the body raw, but the "code" never showed 200. This shows that some of my knowledge is not solid enough. I also have something good to do. I solved above problem and fixed all the bugs in my task by searching a lot of information and Thinking on the Internet. For example, add @JsonSerialize (using = ToStringSerialize.class) to the id attribute of ArticleVo to ensure the id accuracy obtained by the snowflake algorithm. Otherwise, the article will not be displayed.

Acceptance Test

I did too many acceptance tests, even though some of the tests were assigned by people other than me. Here I describe the acceptance criteria for my own main tasks. For #3 View blogs (main page interface), users can see a series of article lists when they enter the website. These article lists show the author, abstract and other information. In addition, when users click the "town" button on other pages, they can enter the home page and see the list of these articles. For #9 Blog full content, when viewing the article list, users can click the article title to see the details of the article. This includes article content, category, tags, comments, etc. For #13 Tag-classified blog list and #14 Category-classified blog list, users click "town article categories" or "town tags" when they want to view all categories or tags, all categories and tags with icons and introductions can be selected.

All the acceptance criteria I described can be seen in the appendices of two reports. I think there are many areas that need to be improved in the PBI. Firstly, these acceptance criteria lack some scenarios. For example, in #3, we should add a scenario: "**GIVEN** users are in login or registration page **WHEN** users click "return" **THEN** the system should display main page." Secondly, the **GIVEN** statement of some acceptance criteria is too simple and repetitive. They are all "**GIVEN** the user is visiting the website". Take the acceptance criteria scenario 1 of #13 as an example, I think "**GIVEN** the user wants to view all tag" is better. Finally, some words in acceptance criteria are imprecise. Take the acceptance criteria scenario 1 of #13 as an example again, our navigation doesn't have "tag list". We only have "Town tags". We need to precise the description.

Change Management

Recording the user's login information during login was originally a user requirement of our software. We have designed related database table for this part. Due to our slow sprint 1

schedule and technical difficulties in obtaining IP addresses, we temporarily abandoned this function to work on higher priority tasks. Filling in basic information such as mobile phone number when registering is also the original user requirement. Considering that our theme is "leisure", we have only retained three indispensable pieces of information to fill out. Developing a section called "latest articles" was originally a user requirement. We were already done with this section, but then we realized that the articles on our home page are in chronological order. Therefore, we changed it to "trendy articles." We handled these changes in two main ways. One is the meeting before each sprint, where everyone gives feedback on the tasks of the previous sprint. The other is to communicate through WeChat groups.

What we did well in this section was replace the latest articles with popular articles. This not only fills in the blanks on our home page, but also solves the changes and improves the user experience. The bad part is that the function of recording user login information is deleted. In fact, we shouldn't remove the entire function because of a few technical difficulties. Although IP addresses are the important data for this function, we can remove only this part of data temporarily. It was a mistake in our decision-making.

Teamwork

Our team's formal organization includes the meeting before every sprint. This is a good way to summarize the issues left over from the previous sprint and plan for the next one. In addition, flexible task completion is another point. We add tasks that were not completed in the previous sprint to the next sprint. We also start the next sprint early if we finish the sprint early. Informal organization is the Daily Scrum. This activity is usually carried out by sending messages in WeChat groups. Since everyone's routine is different, we don't do it every day.

In this project, our team roughly divided 4 people to do the front end and 5 people to do the back end, but the division of labor is not fixed. After the members received their own tasks, they began to develop. During the development, we helped each other. Then I do the testing mostly. I would find out bugs and dissatisfaction with the system and then communicate with members. Finally, Zhaoyu uploaded the code to GitHub. Our main ways of communication are online meetings and WeChat groups. In the meeting, the general planning and direction are mainly discussed, while details are discussed in the WeChat group.

As for the integration work, we originally intended to use GitHub. However, in practice, we found that after the code update, we need to download the whole zip file again. It's too much trouble. We will directly transfer the updated front-end, back-end and database files in the WeChat group. Everyone should record their updated places, and then insert them into the documents sent by the first person to complete. Finally, Zhaoyu integrate all the files into GitHub.

I have a clear goal in every sprint. On the basis of completing my task, I try my best to help others. I also actively initiate discussions and tell members what I think. When I argue about a point, I will take the initiative to ask many people for their opinions, so that everyone has a sense of participation. Then use the principle of "the minority obeys the majority" to make the final

decision. At the same time, I also treat people sincerely, trust members and respect team decisions.

Challenges of The Teamwork

The difficulty of our team work is to integrate the code. Whenever there is an updated file, we need to import it again and in application Modify the URL, account and password of the database in properties. This is troublesome. Another difficulty is that there are often many differences in team communication, resulting in slow progress. For example, what elements should be included in the article list of the home page. Everyone has different ideas.

I think the reason for the first challenge is that we pay too much attention to code development and don't have much time to study how to synchronize code with GitHub. We simply uploaded the code with GIT. Also, we didn't have a unified database schema and password before the whole project started. This leads us to modify this content every time we update the file. I think the second challenge is natural. Everyone naturally has different ideas. This also fully shows that each team member is actively participating and thinking. However, the reason for the slow progress can be said to be our poor decision-making. We should be able to vote anonymously. Then discuss the two schemes with the highest number of votes and implement the principle of "the minority obeys the majority". Instead of discussing all options aimlessly and repeatedly.

Legal, Social, Ethical, and Professional Issues

Considering the legal, social, ethical and professional issues, we used JWT (JSON Web Token) technology when logging in and put the token into redis as double insurance. When logging in for authentication, it will first verify whether the token character is legal, and then go to redis to verify whether it exists. This greatly improves security. Content filtering tools can be deployed in our product. These tools are generally used to analyze HTTP packets contained in network data flows and control the access to IP addresses, URLs, file names and HTTP methods in the data headers. They monitor all information on the network and selectively block connections based on TCP. When blocking, the server sends HTTPFINPUSHACK to client and sends HTTPRST to the server as the client, to prevent and filter users from posting politically sensitive, pornographic and violent, false and deceptive information. When users attempt to post these contents, they will be automatically blocked and warned by the website. In addition, because the markdown editor of our web come from other people's products, we are using someone else's intellectual property rights, and did not communicate with the original developer and authorization, when our products are officially put into use, the infringement of this component is not allowed, we need to obtain permission of the authors to use the product, otherwise, it will cause a series of legal disputes about property rights protection.

Conclusion

In short, this report mainly shows our system, something about our teamwork and my personal opinions. In all sprints, our team members are very united and work together towards the goal

of "Leisure Town". Although we have some conflicts, we are not afraid of them. If this project can be developed offline, I believe our team will complete it better and more efficiently.

Lesson Learnt and Future Work

In the future, we will fix the bugs left over from the first those four sprints. These bugs include the reverse order of comment floors, the article problems displayed in some town tags, and some pictures and Emoji can't be displayed. We will also adjust some elements of the page to make the page look more beautiful. This includes unifying the language of the software, the location of the town slogan, replacing the slide bar of the home page with the number of pages, and the location of the user's Avatar when writing articles. At the same time, add more functions to our software, such as viewing and changing personal information, liking blogs. Based on the evaluation, I will master more knowledge about GitHub to facilitate our code integration, refine our use case diagram, learn more knowledge to develop new functions, find a person who specializes in testing and propose to the team members to assign tasks according to the technical level of the members.

I learned many lessons in all those sprints. I learned how to use GitHub to share code, HTML page design, build back-end projects, how to use postman to test, how to connect front-end and back-end, how to connect the database. I learned to use GitHub through class content, CSDN and Bilibili which shared by team members. HTML page design is that I find the source code of the login registration page, compare it with the generated page, and modify some code to see the effect. I searched on the Internet and compared a lot of materials and classroom content to learn. The construction of the project is also a key learning experience, because it is the most basic step and a troublesome step. I put forward many useful suggestions for project construction, such as looking for the relevant code of cross domain configuration on the official website. I learned postman by class content and exploring for myself. I knew where the path variable should put and what is the reasonable results. I also learned discipline, project management and teamwork in your class last semester. I reviewed it in this project. Through this team project, I learned the basic process of agile development, the importance of actively communicating with team members and increased my practical experience. At the same time, it also improved my programming thinking ability. I think I can include this group project in my resume, whether I apply for graduate school or work. The completion of this project also gives me confidence in related work in the future.

Appendix

1. The content of the additional material in the zip file:

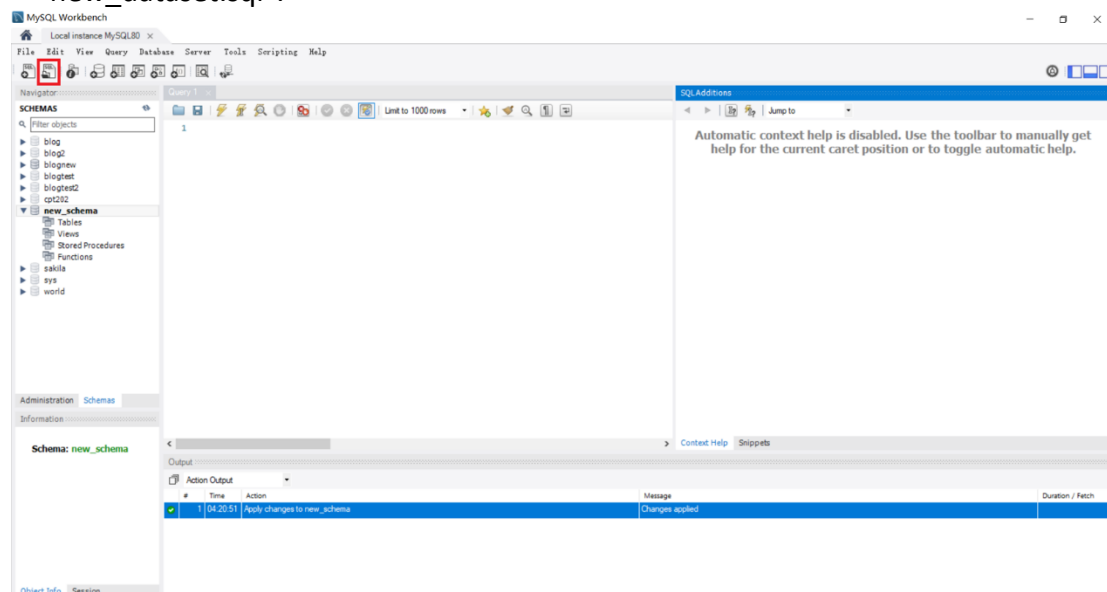
The zip file includes front-end project (folder "blog-app"), back-end project (folder "Leisure_Town"), database (file "new_dataset.sql") and redis (folder "redis1" for Windows, folder "redis2" for MacOS).

2. GitHub link and how to access the source code:

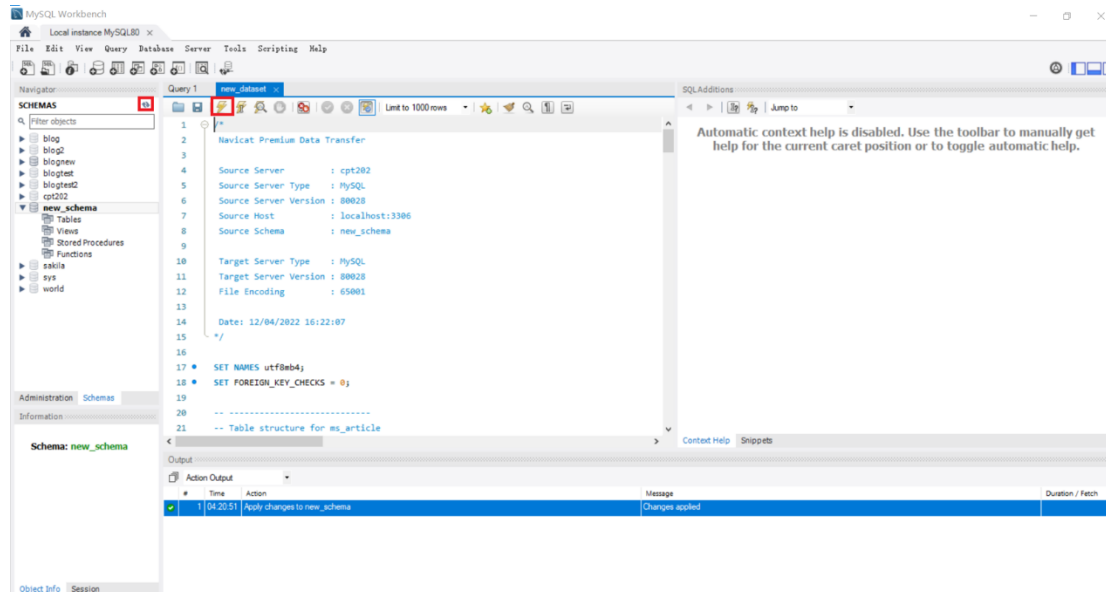
In addition to accessing our code and redis on learningmall, you can also access them from GitHub (https://github.com/ZhaoyuXu/leisure_town). You can clearly see that our back-end folder is "Leisure_Town", the front-end folder is "blog-app", the database file is "new_dataset.sql" and the required redis is in the "redis1" folder (Windows version) or "redis2.zip" (MacOS version). For some unknown reason, we can't upload folder "redis2" completely, so we changed it to "redis2.zip".

Please follow the instructions below to set up the environment:

- Download two main projects (front-end: folder "blog-app", back-end: folder "Leisure_Town"), database file ("new_dataset.sql") and redis (folder "redis1" or "redis2.zip") from GitHub link shown above.
- Install Node.js from <https://nodejs.org/en/download/>
- Start MySQL Workbench
 - o Create a new schema called whatever you like (we recommend calling "new_schema") and double-click to select the schema.
 - o Click "Open a SQL script file in a new query tab" and open our database file "new_dataset.sql".



- o Click "Execute the selected portion of the script or everything, if there is no selection" and then refresh schema.

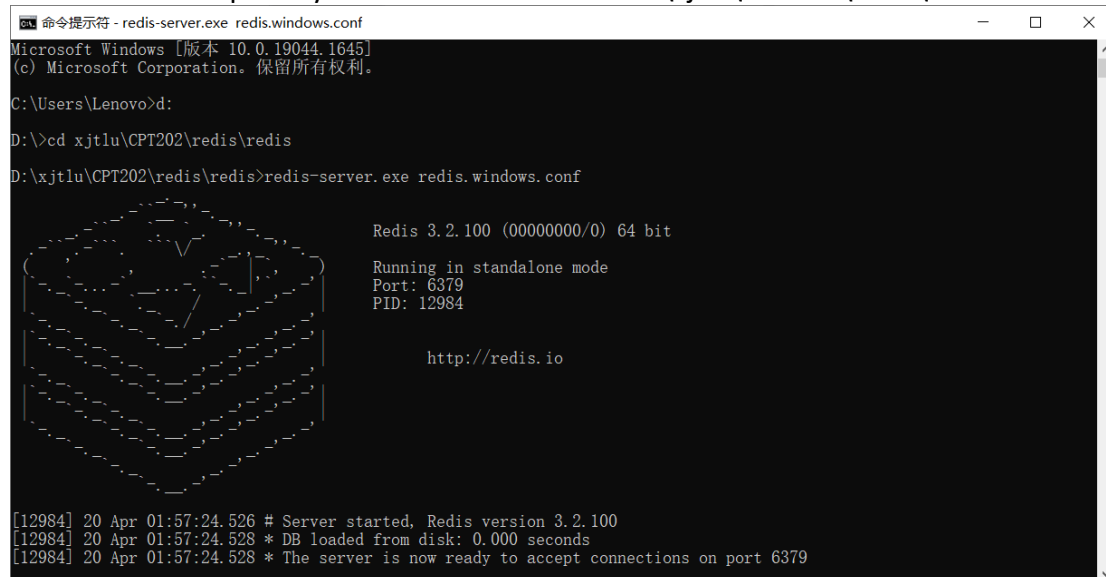


- Start "redis-server.exe" in our folder "redis1" (Windows version) or folder "redis2\src" (MacOS version) which is decompressed from our "redis2.zip". (Don't forget this step, otherwise many functions will be unavailable if you can't register and log in!!!)

If you can't open redis successfully by clicking "redis-server.exe" directly:

For Windows:

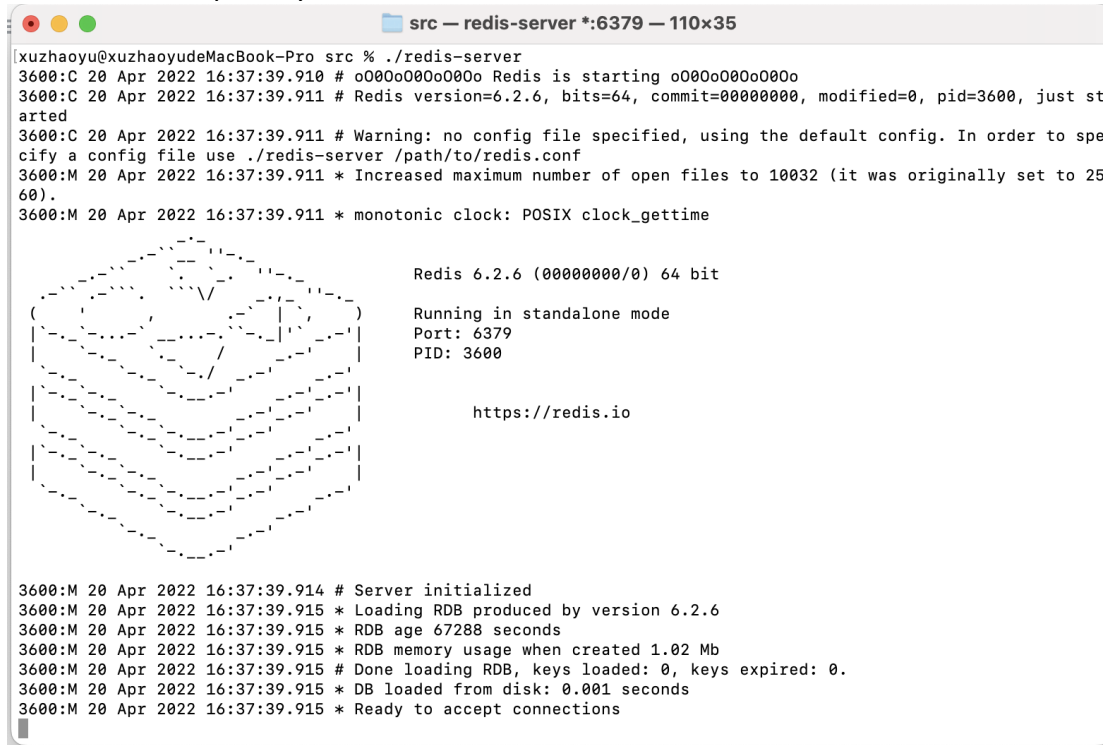
- o Open "cmd".
- o Use the command to enter the folder "redis1" you downloaded from GitHub.
- o Enter the command "redis-server.exe redis.windows.conf" and execute it.
- o Here is an example: my "redis-server.exe" is in D:\xjtl\CPT202\redis\redis



For MacOS:

- o Open "Terminal".
- o Use the command to enter the folder "redis2\src" which is decompressed from "redis2.zip".
- o Enter the command "./redis-server" and execute it.

o Here is an example: my "redis-server.exe" is in /usr/local/redis-6.2.6/src



```
src — redis-server *:6379 — 110x35
[xuzhaoyu@xuzhaoyudeMacBook-Pro src % ./redis-server
3600:C 20 Apr 2022 16:37:39.910 # 000000000000 Redis is starting 000000000000
3600:C 20 Apr 2022 16:37:39.911 # Redis version=6.2.6, bits=64, commit=00000000, modified=0, pid=3600, just started
3600:C 20 Apr 2022 16:37:39.911 # Warning: no config file specified, using the default config. In order to specify a config file use ./redis-server /path/to/redis.conf
3600:M 20 Apr 2022 16:37:39.911 * Increased maximum number of open files to 10032 (it was originally set to 2560).
3600:M 20 Apr 2022 16:37:39.911 * monotonic clock: POSIX clock_gettime

Redis 6.2.6 (00000000/0) 64 bit

Running in standalone mode
Port: 6379
PID: 3600

https://redis.io

3600:M 20 Apr 2022 16:37:39.914 # Server initialized
3600:M 20 Apr 2022 16:37:39.915 * Loading RDB produced by version 6.2.6
3600:M 20 Apr 2022 16:37:39.915 * RDB age 67288 seconds
3600:M 20 Apr 2022 16:37:39.915 * RDB memory usage when created 1.02 Mb
3600:M 20 Apr 2022 16:37:39.915 # Done loading RDB, keys loaded: 0, keys expired: 0.
3600:M 20 Apr 2022 16:37:39.915 * DB loaded from disk: 0.001 seconds
3600:M 20 Apr 2022 16:37:39.915 * Ready to accept connections
```

If you still have some problems about it in MacOS, please follow this blog:

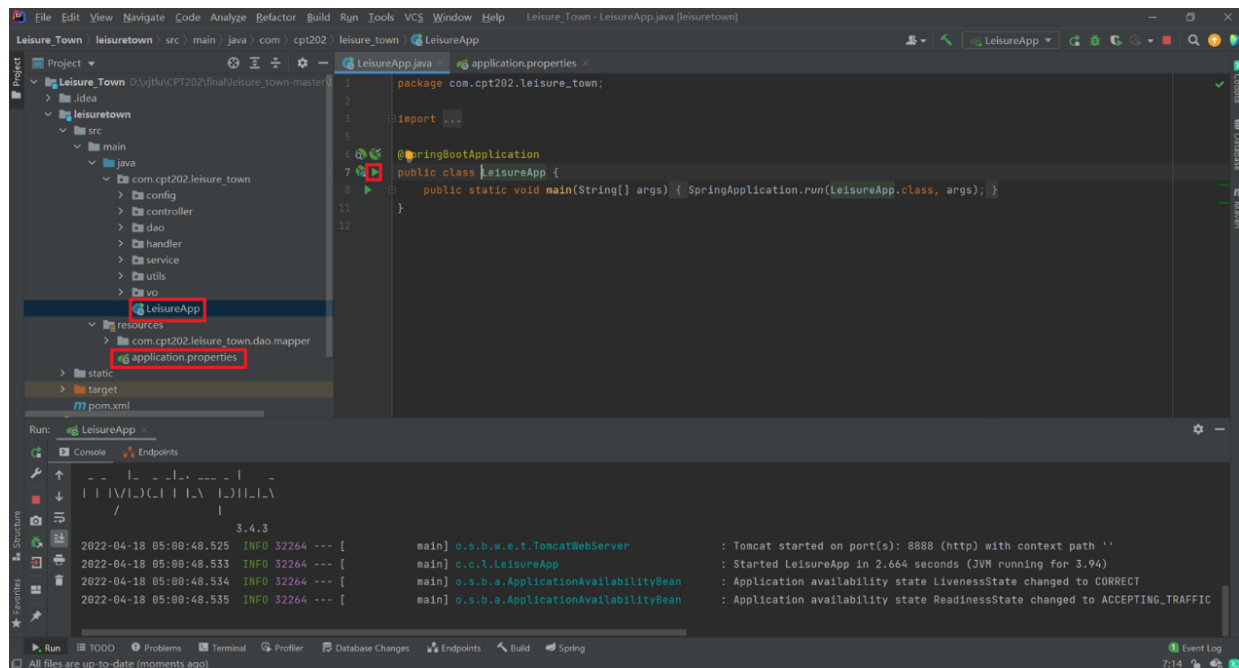
https://blog.csdn.net/weixin_45037570/article/details/118500615

For Linux:

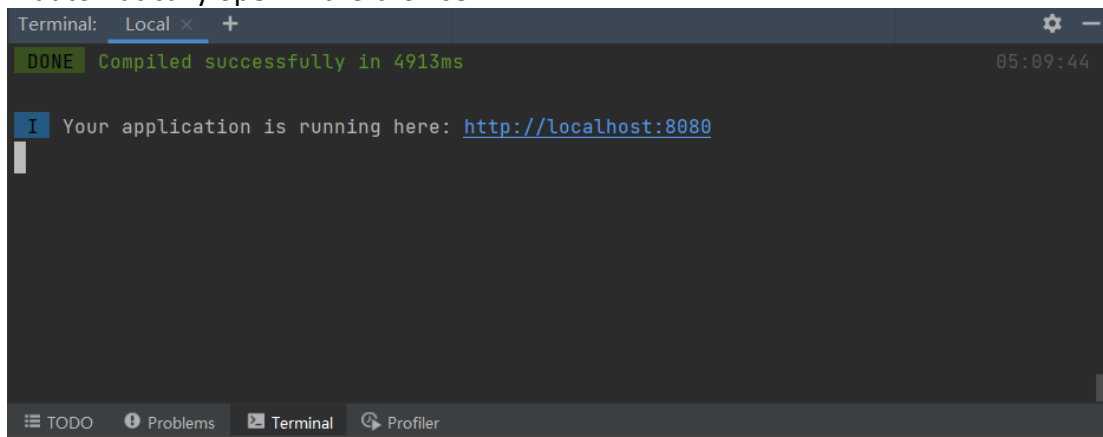
The steps are similar to those of MacOS. You can also check out with this blog:

https://blog.csdn.net/qg_38728790/article/details/82703308?spm=1001.2101.3001.6661.1&utm_medium=distribute.pc_relevant_t0.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1.pc_relevant_default&depth=1-utm_source=distribute.pc_relevant_t0.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1.pc_relevant_default&utm_relevant_index=1

- Run back-end project "Leisure_Town"
 - o Use IntelliJ IDEA to open the project.
 - o Install related libraries and plugins "MyBatisX" and "Lombok".
 - o Go to "application.properties" file and change the "username" and "password" to your own MySQL account and change the "url" to your own schema which is created earlier. For example, my localhost is "3306" and my schema is called "new_schema", so url is "jdbc:mysql://localhost:3306/new_schema".
 - o Run the SpringBoot project by "LeisureApp.java".

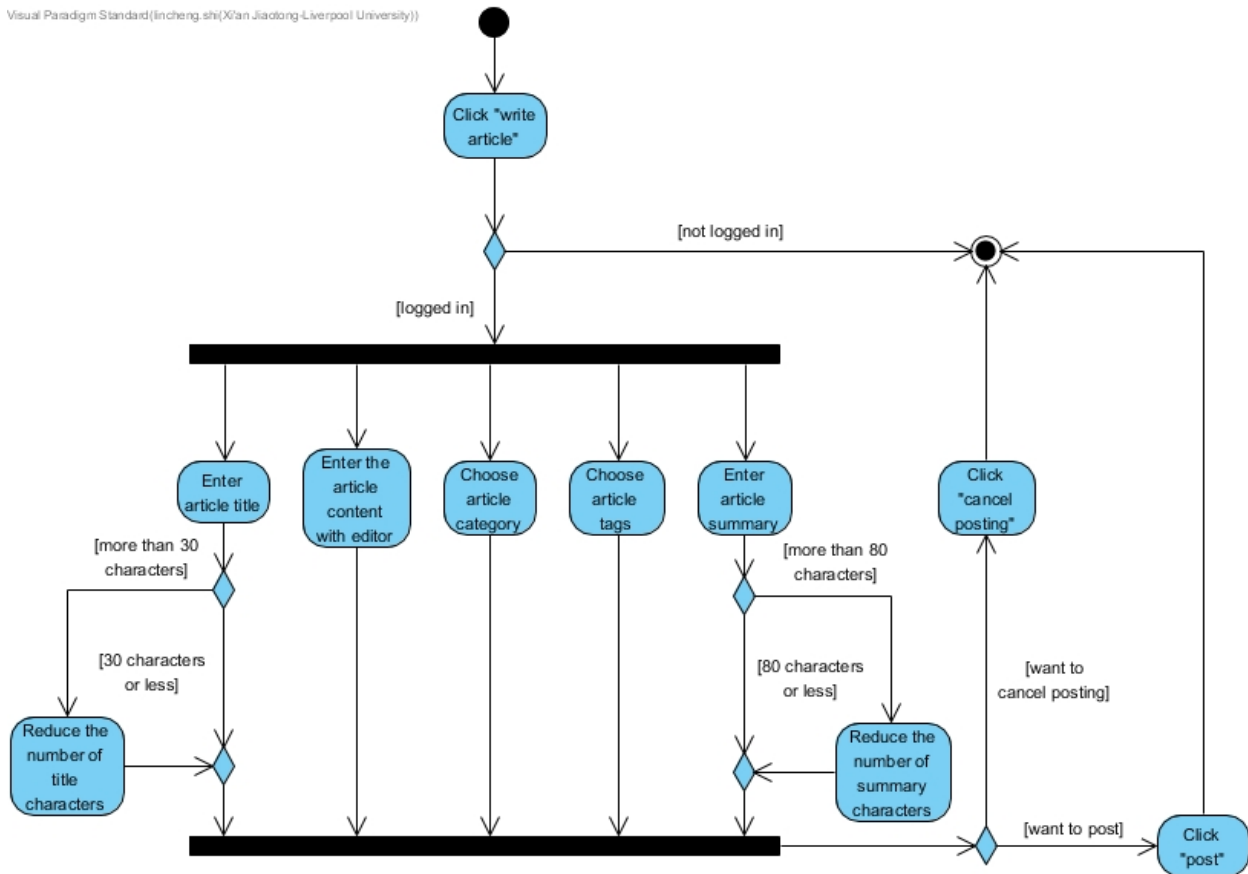


- Run front-end project "blog-app"
 - o Use IntelliJ IDEA to open the project in a new window.
 - o Enter "npm install" in terminal to install related libraries.
 - o Enter "npm run build" in terminal and then enter "npm run dev" in terminal to run the project.
 - o You will see a link in terminal and just click it, the web page "Leisure Town" will automatically open in the browser.



- You can use this account (account: user, password: 123) or register a new account to login to use more functions.

3. The activity diagram of another idea (Post articles):



4. Details of PBIs:

#12 Trendy blogs

PRODUCT BACKLOG ITEM 82

82 Trendy blogs

Unassigned

0 comments Add tag

State	Done	Area	Leisure Town
Reason	Work finished	Iteration	Leisure Town\Sprint 4

Description

The function to arrange the most frequently viewed blogs together in one section.

As a user, I want to see the trendy blogs **so that** I can quickly get access to the blogs that are popular on the website.

Q: How to define the trendy blogs?

A: The blogs that have the most view count in the list of posted blogs.

Q: How many blogs do we need to show in the trendy blog section?

A: 5 blogs would be appropriate.

Acceptance Criteria

Scenario 1: A user wants to see trendy blogs

GIVEN the user is visiting the website **WHEN** the user tries to see trendy tags **THEN** the system should display current trendy tags in the particular position on the main page.

Details

Priority

4

Effort

4

Business Value

4

Value area

Business

#13 Tag-classified blog list

PRODUCT BACKLOG ITEM 207

207 Tag-classified blog list

Unassigned

0 comments

Add tag

State

Done

Area

Leisure Town

Reason

Work finished

Iteration

Leisure Town\Sprint 4

Description

The function to arrange all blogs that share one tag together in one section.

As a user, I **want to** see blogs under each tag **so that** I can quickly get access to all blogs with similar content.

Q: What should the tag list look like?
A: Four tags in a row, with an icon clearly showing the topic of the tags, the tag name and a short description under it. Also a button in the navigation bar should be designed, next to the main page button.

Q: What should the blog list under a certain tag look like?
A: The 3 tag elements in the above question should be put on the top to show the tag those blogs belong to. And a list of blogs just like the UI of the main page should be listed under it.

Details

Acceptance Criteria

Scenario 1: A user wants to see the tag list
GIVEN the user is visiting the website **WHEN** the user clicks on the 'Tag list' button in the navigation bar **THEN** the system should redirect to the tag list page showing all tags.

Scenario 2: A user wants to see blogs under one certain tag
GIVEN the tag has some blogs **WHEN** the user clicks on one tag in the tag list page **THEN** the system should redirect to the page showing all blogs using the specific tag.

#14 Category-classified blog list

PRODUCT BACKLOG ITEM 208

208 Category-classified blog list

Unassigned

0 comments

Add tag

State

Done

Area

Leisure Town

Reason

Work finished

Iteration

Leisure Town\Sprint 4

Description

The function to arrange all blogs that share one category together in one section.

As a user, I **want to** see blogs under each category **so that** I can quickly get access to all blogs that have a similar topic.

Q: What is the difference between 'Category' and 'Tag'?
A: 'Category' covers more widely than 'Tag'. 'Category' is the field that the blog belongs to, and 'Tag' shows the detailed characteristics of the blog. E.g., 'Cooking' --> 'Category', 'Desserts' --> 'Tag'.

Q: What should this section look like?
A: Same with the tag-classified blog list to maintain consistency.

Details

Acceptance Criteria

Scenario 1: A user wants to see the category list
GIVEN the user is visiting the website **WHEN** the user clicks on the 'Category list' button in the navigation bar **THEN** the system should redirect to the category list page showing all categories.

Scenario 2: A user wants to see blogs under one certain category
GIVEN the category has some blogs **WHEN** the user clicks on one category in the category list page **THEN** the system should redirect to the page showing all blogs belonging to the specific category.

#15 Search by blog title

PRODUCT BACKLOG ITEM 46

46 Search by blog title

Unassigned

0 comments

Add tag

State

Done

Area

Leisure Town

Reason

Work finished

Iteration

Leisure Town\Sprint 4

Description

The function to search for wanted blogs.

As a user, I **want to** search for blogs **so that** I can quickly get access to the blog that meet my need/interest.

Q: Where should the search bar be placed in the navigation bar?

A: On the left of the login/registration function.

Q: What should be responded after users type in the search bar?

A: Blog titles that are associated to the input should be listed under the search bar while typing. And after the user clicks on one of them, the website should be redirected to the detail page of the chosen blog.

Acceptance Criteria

Scenario 1: During typing: The input has some associated posted blogs

GIVEN the input has some associated posted blogs **WHEN** the user is typing **THEN** the system should list all blogs that has a title including the input for the user to choose.

Scenario 2: After typing: The user chooses the wanted blog based the input

GIVEN the corresponding blog is valid **WHEN** the user clicks on one result in the blog list under the search bar **THEN** the system should redirect to the detail page of the blog.

5. Return results in postman:

```
post http://localhost:8888/articles {"page": 1, "pageSize": 10}:
```

The screenshot displays a REST client interface with the following details:

- URL Bar:** `http://localhost:8888/articles`
- Method:** `POST`
- Request Body:**

```
{  "page": 1,  "pageSize": 10}
```
- Status Bar:** Status: 200 OK, Time: 101 ms, Size: 2.22 KB
- Response Body:**

```
{  "success": true, "code": 200, "msg": "success", "data": [{    "id": "1516084926579511297", "title": "Hero", "summary": "Introduction about DC heroes", "commentCounts": 1, "viewCounts": 27, "weight": 0,    "createDate": "2022-04-19 08:03", "author": "user", "body": null, "tags": [{"id": "15", "tagName": "Movie", "avatar": "/static/tag/movie.jpg"}], "category": null, "id": "1081", "title": "Jay Chou",    "summary": "Stories about Jay Chou.", "commentCounts": 2, "viewCounts": 11, "weight": 0, "createDate": "2022-04-12 16:13", "author": "\u5f9e\u56fd", "body": null, "tags": [{"id": "9", "tagName": "Music", "avatar": "/static/tag/music.jpg"}, {"id": "11", "tagName": "Artist", "avatar": "/static/tag/artist.jpg"}], "category": null, "id": "1095", "title": "Taylor Swift", "summary": "Know more about Taylor Swift.", "commentCounts": 0, "viewCounts": 8, "weight": 0, "createDate": "2022-04-12 16:12", "author": "\u5f9e\u56fd", "body": null, "tags": [{"id": "9", "tagName": "Music", "avatar": "/static/tag/music.jpg"}, {"id": "11", "tagName": "Artist", "avatar": "/static/tag/artist.jpg"}], "category": null, "id": "1084", "title": "League Of Legends", "summary": "MOBA!", "commentCounts": 0, "viewCounts": 6, "weight": 0, "createDate": "2022-04-12 16:11", "author": "xjtlu", "body": null, "tags": [{"id": "3", "tagName": "E-sports", "avatar": "/static/tag/e-sports.jpg"}], "category": null, "id": "1083", "title": "Pizza", "summary": "Delicious Pizza!!!", "commentCounts": 0, "viewCounts": 4, "weight": 0, "createDate": "2022-04-12 16:10", "author": "xjtlu", "body": null, "tags": [{"id": "1", "tagName": "Food", "avatar": "/static/tag/food.jpg"}], "id": "2", "tagName": "Recipe", "avatar": "/static/tag/recipe.jpg"}, {"id": "1992", "title": "Eiffel Tower", "summary": "Travel to Paris.", "commentCounts": 0, "viewCounts": 17, "weight": 0, "createDate": "2022-04-11 19:04", "author": "xjtlu", "body": null, "tags": [{"id": "12", "tagName": "Tourist attraction", "avatar": "/static/tag/tourist attraction.jpg"}, {"id": "13", "tagName": "Travel strategy", "avatar": "/static/tag/travel strategy.jpg"}], "id": "14", "tagName": "Scenic spot introduction", "avatar": "/static/tag/scenic spot introduction.jpg"}, {"category": null}]}
```

```
Body Cookies Headers (8) Test Results Status: 200 OK Time: 71 ms Size: 2.22 KB Save Response
Pretty Raw Preview Visualize JSON
1 {
2   "success": true,
3   "code": 200,
4   "msg": "success",
5   "data": [
6     {
7       "id": "1516084926579511297",
8       "title": "Hero",
9       "summary": "Introduction about DC heroes",
10      "commentCounts": 1,
11      "viewCounts": 27,
12      "weight": 0,
13      "createDate": "2022-04-19 00:03",
14      "author": "user",
15      "body": null,
16      "tags": [
17        {
18          "id": 15,
19          "tagName": "Movie",
20          "avatar": "/static/tag/movie.jpg"
21        }
22      ],
23      "category": null
24    }
25  ]
26 }
```

post http://localhost:8888/articles/view/1001 :

http://localhost:8888/articles/view/1001

POST http://localhost:8888/articles/view/1001

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

```
Body Cookies Headers (8) Test Results Status: 200 OK Time: 24 ms Size: 1.92 KB Save Response
Pretty Raw Preview Visualize
{"success":true,"code":200,"msg":"success","data":{"id":"1001","title":"Jay Chou","summary":"Stories about Jay Chou.","commentCounts":2,"viewCounts":12,"weight":0,"createDate":"2022-04-12 16:13","author":"李四","body":{"content":"Jay Chou was born in Xinbei City, Taiwan Province, and his ancestral home is Yongchun County, Quanzhou City, Fujian Province. When he was 4 years old, his mother, Ye Hui-mei, sent him to tania Yamaha Kindergarten music class to learn piano. When he was in the second grade of junior high school, his parents divorced due to personality differences, and Jay returned to his mother, Ye Hui-mei, to raise him. Middle school, did not take an examination of ordinary high school, the same year, because good at piano and was admitted to the first music class danjiang middle school. After graduating from high school, I failed to get into the music department of Taipei University twice, so I started working in a restaurant. \nIn September 1997, encouraged by his mother, Jay signed up for the Taipei Starlight TV entertainment show \"Super Newcomer\" and invited people to sing his own song \"Dreams Have Wings\". When host Wu Zongxian saw the score of the song, he invited Chou to work as a music assistant at Alfa Music Company. In 1998, she composed the song \"Tears Know\", which the company gave to Andy Lau and was rejected. Her songs \"Nunchucks\" and \"Ninja\" (later included in Jay Chou's album \"Fantexi\") were also rejected.\"},"tags":[{"id":9,"tagName":"Music","avatar":"/static/tag/music.jpg"}],"id":11,"tagName":"Artist","avatar":"/static/tag/artist.jpg"},"category":{"id":4,"avatar":"/static/category/art.jpg","categoryName":"Art","description":"Art comes from life."}}}
```

```
Body Cookies Headers (8) Test Results Status: 200 OK Time: 11 ms Size: 1.92 KB Save Response
Pretty Raw Preview Visualize JSON
1 {
2   "success": true,
3   "code": 200,
4   "msg": "success",
5   "data": {
6     "id": "1001",
7     "title": "Jay Chou",
8     "summary": "Stories about Jay Chou.",
9     "commentCounts": 2,
10    "viewCounts": 17,
11    "weight": 0,
12    "createDate": "2022-04-12 16:13",
13    "author": "李四",
14    "body": {
15      "content": "Jay Chou was born in Xinbei City, Taiwan Province, and his ancestral home is Yongchun County, Quanzhou City, Fujian Province. When he was 4 years old, his mother, Ye Hui-mei, sent him to tania Yamaha Kindergarten music class to learn piano. When he was in the second grade of junior high school, his parents divorced due to personality differences, and Jay returned to his mother, Ye Hui-mei, to raise him. Middle school, did not take an examination of ordinary high school, the same year, because good at piano and was admitted to the first music class danjiang middle school. After graduating from high school, I failed to get into the music department of Taipei University twice, so I started working in a restaurant. \nIn September 1997, encouraged by his mother, Jay signed up for the Taipei Starlight TV entertainment show \"Super Newcomer\" and invited people to sing his own song \"Dreams Have Wings\". When host Wu Zongxian saw the score of the song, he invited Chou to work as a music assistant at Alfa Music Company. In 1998, she composed the song \"Tears Know\", which the company gave to Andy Lau and was rejected. Her songs \"Nunchucks\" and \"Ninja\" (later included in Jay Chou's album \"Fantexi\") were also rejected."
16    },
17    "tags": [
18      {
19        "id": 9,
20        "tagName": "Music",
21        "avatar": "/static/tag/music.jpg"
22      }
23    ]
24  }
25 }
```

get http://localhost:8888/tags/detail :

http://localhost:8888/tags/detail

Save



OET http://localhost:8888/tags/detail

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (8) Test Results

Status: 200 OK Time: 12 ms Size: 1.32 KB Save Response

Pretty Raw Preview Visualize



```
{
  "success": true,
  "code": 200,
  "msg": "success",
  "data": [
    {
      "id": 1,
      "tagName": "Food",
      "avatar": "/static/tag/food.jpg"
    },
    {
      "id": 2,
      "tagName": "Recipe",
      "avatar": "/static/tag/recipe.jpg"
    },
    {
      "id": 3,
      "tagName": "E-sports",
      "avatar": "/static/tag/e-sports.jpg"
    },
    {
      "id": 4,
      "tagName": "Ball game",
      "avatar": "/static/tag/ball game.jpg"
    },
    {
      "id": 5,
      "tagName": "Athletics",
      "avatar": "/static/tag/athletics.jpg"
    },
    {
      "id": 6,
      "tagName": "Other sports",
      "avatar": "/static/tag/other sports.jpg"
    },
    {
      "id": 7,
      "tagName": "Daily life",
      "avatar": "/static/tag/daily life.jpg"
    },
    {
      "id": 8,
      "tagName": "OOTD",
      "avatar": "/static/tag/oootd.jpg"
    },
    {
      "id": 9,
      "tagName": "Music",
      "avatar": "/static/tag/music.jpg"
    },
    {
      "id": 10,
      "tagName": "Fine arts",
      "avatar": "/static/tag/fine arts.jpg"
    },
    {
      "id": 11,
      "tagName": "Artist",
      "avatar": "/static/tag/artist.jpg"
    },
    {
      "id": 12,
      "tagName": "Tourist attraction",
      "avatar": "/static/tag/tourist attraction.jpg"
    },
    {
      "id": 13,
      "tagName": "Travel strategy",
      "avatar": "/static/tag/travel strategy.jpg"
    },
    {
      "id": 14,
      "tagName": "Scenic spot introduction",
      "avatar": "/static/tag/scenic spot introduction.jpg"
    },
    {
      "id": 15,
      "tagName": "Movie",
      "avatar": "/static/tag/movie.jpg"
    }
  ]
}
```

Body Cookies Headers (8) Test Results

Status: 200 OK Time: 9 ms Size: 1.32 KB Save Response

Pretty Raw Preview Visualize

JSON



```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
{
  "success": true,
  "code": 200,
  "msg": "success",
  "data": [
    {
      "id": 1,
      "tagName": "Food",
      "avatar": "/static/tag/food.jpg"
    },
    {
      "id": 2,
      "tagName": "Recipe",
      "avatar": "/static/tag/recipe.jpg"
    },
    {
      "id": 3,
      "tagName": "E-sports",
      "avatar": "/static/tag/e-sports.jpg"
    },
    {
      "id": 4,
      "tagName": "Ball game",
      "avatar": "/static/tag/ball game.jpg"
    }
  ]
}
```