BBQ everything Knowledge base

Welcome to BBQ everything documentation portal

Here are a few suggestions to get started::

- Quick links to external tools and internal material for quick start:
- User story: https://docs.google.com/spreadsheets/d/1BHR9LcNoEyqS7PgytB-nVLzf6WqQ4P8DjbZbEJL4I7k/edit?skip_itp2_check=true#gid=0
- Design of general idea: https://lucid.app/lucidchart/1582e1e7-196c-43fd-bd68-ac6b242491a0/edit?page=0_0&invitationId=inv_1c53ccb1-eeb9-491b-8dbd-970b0359ecfe#
- Specific design of architecture General architecture design of project
- Database module: Design of the database
- Spring Boot document: https://spring.io/projects/spring-boot/
- VUE document: https://vuejs.org/guide/introduction.html
- Motivational Model: Motivational Model
- Job assignment: 2022-08-22 Meeting notes
- Sprint Artefacts Sprint Artefacts
- Coding standard:Coding Standard
- Deployment: Deployment instruction
- API protocol: API Documentation
- Testing plan: Integration Testing

Documentation overview | User guide | Most viewed documents | Recently updated

Documentation overview

This documentation portal records all the documents for the developing purpose and internal communication of this project.

User guide

It is mainly divided into serval characters includes the Sprint Artefacts , Front End Development and Back End Development , internal instruction on how to accessing other tools, internal communication Meeting notes , progress report Internal progress tracking

Most viewed documents

Sprint Artefacts This is the quick link to Artefacts.

Recently updated

You'll see the 5 most recently updated pages that you and your team create.

BBQ everything Knowledge base
Nov 05, 2022 • contributed by Jiajun MAO

Integration Testing
Nov 03, 2022 • contributed by Yuzhi Yan

DevLog of Backend
Nov 03, 2022 • contributed by Yuzhi Yan

API Documentation
Nov 03, 2022 • contributed by Yuzhi Yan

API Documentation for Sprint 0
Nov 03, 2022 • contributed by Yuzhi Yan

Sprint Artefacts

Product backlog and requirements Requirements and arrangements

General design of the project General Design Pages with front end and back end included.

As it is a computing subject, we use 0 as the first number so our sprint 0 is similar to most of the other group's sprint 1, which is a normal sprint working on the coding and other standard project stuff.

Sprint0 planning and checklist Sprint 0 Planing

Sprint0 review Sprint 0 Review

Sprint1 planningSprint 1 Planning

Spring1 reviewSprint 1 Review

Sprint 0 Planing

Story Point Division:

23 out of 27 user stories are aiming to be completed for sprint 0. This is a high percentage of the user stories, however, the remaining 4 user stories are highly difficult tasks and are disproportionately time consuming functionalities that will be best implemented at a later date.

Scope Overview:

The goal for sprint 0 is to have 23 of the user stories implemented and in a functional state, along with a working server that is accessible remotely.

We assume that all aspects of the system's architecture will be able to interact freely and are appropriately interconnected.

Sprint 0 plan:

Front end:

- research on the related resources
- understand the requirements and the UI design
- define and decide the protocol with backend
- complete the development of UI and the initial set up of the framework
- finish the optimization and compile the webpack
- development on the css
- unit testing and optimizing
- finish sprint 0

Backend:

- research on the backend development resources
- understand the requirements and finish the design of the framework
- complete the design of database and interface
- complete the initial set up of the development framework
- initial the database and interface of the development
- complete the functional requirements of backend
- unit test and functional test
- finish sprint 0

Sprint 0 Review

The main goals for Sprint 0 were to complete 23 out of the 27 user stories. This was an overshot by a large margin. Most likely due to an overestimate of our efficiency to complete the smaller tasks one by one quickly. The initial setup of the site proved to be more time consuming and complex than at first imagined, resulting in a bulk of time being spent on preparing that aspect.

User Stories

- Completed 7 user stories
- 20 user stories still to potentially implement

Overview of completed jobs

- Organized fundamental architecture of the project

- Sorted important information into appropriate sections on confluence
- Set up a basic home page
- Created a server that can be accessed remotely at all times
- Successfully deployed the back-end database to the cloud so that it may be accessed remotely

Overview of not completed jobs

- Integration testing between front-end and back-end
- Additional functionalities need to be implemented

Experiences during Sprint 0

- We had to overcome difficulties regarding network connection and time differences as our members live far apart
- Improved documentation over the course of sprint 0
- Got familiar with the agile development process
- Using different working tools to increase the group working efficiency

More points that need to be improved on

- Struggled with decision making regarding which programming tools to utilize to complete the task
- Difficulties using git as a collective resulting in the creation of 2 new repositories
- Problems with integration testing and the design of the testing plan
- We should more frequently communicate our status and progress on the project
- More sharing is needed so that everyone can know what others are working on, and what needs to be worked on
- We should work more as a group to overcome issues faster and more efficiently

Sprint 1 Planning

Story Point Division:

In the previous routine, we completed 7 out of the total 27 user stories that we have.

The schedule of this sprint is to complete the rest 13 out of 27 and complete the rest part of the user stories as many as possible. We might cancel a few of the user story based on the time and resources that some of it might take far more than our exception.

Scope Overview:

The goal for sprint 1 is to have 13 of the user stories implemented and in a functional state, along with a working server that is accessible remotely.

Fixing and optimising the feedback from the previous stage is also considered as one of the most important job for this turn.

We assume that all aspects of the system’s architecture will be able to interact freely and are appropriately interconnected.

We also want to make everything user friendly and should be able to access easily without taking too much of the time on reading through the instructions.

Sprint 1 plan:

Front end:

- [x] optimising the user interface
- [x] access the amount of the work for some of the user story
- [x] finalising the function of comments
- [x] finish the insertion of slogan and other accessibility options
- [x] optimising the interface with the backend to increase the responsibility
- [x] finishing the maintenance document
- [x] finish the user instruction
- [x] unit testing and optimising
- [ ] finish sprint 1

Backend:

- [x] optimising the response speed of the server
- [x] access the amount of the work for some of the user story
- [x] increasing the reliability of the backend
- [x] finishing the comments and like function

- [x] finish the user management function
- [x] finish the user manual and instruction on maintenance
- [x] compiling other important functions after accessing
- [x] unit test and functional test
- [ ] finish sprint 1

Sprint 1 Review

The main goals for Sprint 1 were to complete 13 out of the 27 user stories that haven't been completed. Despite some of the functionalities still needing to be optimised, generally most of the stories have been completed with some of them bein cancelled considering the extraneous amount of work.

User Stories

- Completed 10 user stories
- Other user stories are cancelled (considered as 'will not' priority) as they take too much time and effort under the circumstances for this subject.

Overview of completed jobs

- Deeply optimised the product to make it more functional
- Optimised the structure of the confluence and continued updating the necessary documents
- A more completed homepage has been set up for the project
- Deployed the project to a remote server which allows all-day access of the product
- Implemented important features including the comment and management function of the forum

Overview of not completed jobs

- internal direct messaging system and the recommendation system
- Few of the functionalities are cancelled

Experiences during Sprint 1

- We had to overcome difficulties regarding the unstable connection between different locations as well some of the uncertain policy issues in China which might impact the efficiency of the team
- A better team work model has been created based on everyone's advantages
- more comprehensive understanding of the agile process
- Learnt how to give a presentation for the product

More points that need to be improved on

- Struggled with the time assessment as some of the tasks consumed more time than expected
- Difficulties in applying auto CI/CD Tools to increasing the efficiency of testing
- A better and more frequent communication among the whole group is still needed
- The documentation ability of all group members need to be improved.
- the design abilities of the group still leaves much to be desired.

Requirements and arrangements

"Barbeque Everything" is an online forum space that allows for users to share and discuss various different forms of entertainment, especially movies, comics, anime, TV series etc. Each user can create their own profile on the forum and create posts, as well as comment and share other' s posts. Each post has a title, body, and an optional space for media links related to the post. These posts are also connected to the content included in the post to aid in searches and analysis.

As our client did not provide their own requirements to us, we decided to create something that would be useful to all members of the group. The idea was to create a sharing space for people that wish to review and talk about popular entertainment. It serves as a place for people to share their own theories and reviews, as well as view what others have to say about the topic.

Our first meetings attempted to define clear requirements for the product. The core requirements agreed upon are:

1. Clean looking interface that is visually appealing
2. Simple for user to navigate the site and find what they are looking for
3. Ensures the users privacy and safety while using
4. Easy to maintain as well as expand in the case of a large increase in user count

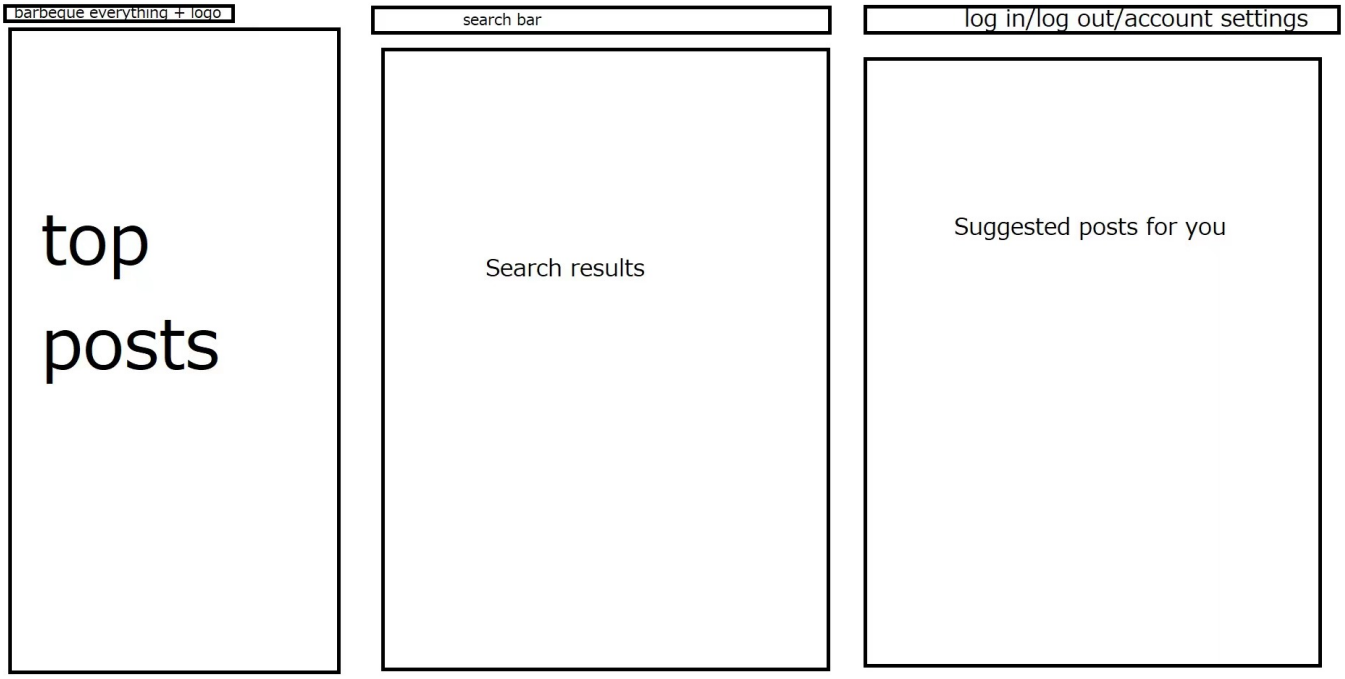
As a	I want to	so	Priority	Sprint num
guest	register	i can become a user	1 Must	sprint0
guest	browse the content	i can obtain the information that i want	1 Must	sprint0

user	leave the comments on the product	share my ideas and get feedback from other people	1 Must	sprint1
user	update the current post	user can make changes if they made an error	1 Must	
user	upvote/downvote button on comments/posts	support ideas i agree with	1 Must	sprint1
user/guest	search for key words	i can filter the posts to find what i want	1 Must	sprint1
user/guest	share link to post/comment	I can show others who may be interested	1 Must	sprint1
user	add media to post	show others content that is being discussed	1 Must	
manager	delete some unfriendly comments	keep the forum fair and legal	2 Should	sprint1
manager	manage the users and their activities	provide people with good experience	2 Should	sprint1
manager	change the status of user	remove inactive user or those always leaving illegal information	2 Should	sprint1
user/guest	View different categories of posts	I can see posts im interested in	2 Should	sprint1
user	change my password/email	update my security information	3 could	sprint1
user	Logout	i can maintain privacy, especially if on a shared computer	3 could	sprint1
user	format my text	I can convey my ideas clearer to readers	3 could	sprint 1
user	bookmark posts/comments	to comeback to interesting posts/threads	3 could	sprint1
user	edit my profile information	display my interests /information to others	3 could	sprint0
user	get recommendation from forum	i can find other content i like	4 will not	Cancel
user	avoid spoilers by hiding text /images	myself and others dont get spoiled by posts	4 will not	Cancel
user	report illegal comments	keep the forum fair and legal	4 will not	Cancel
manager	make advertisements on the site	I can earn money for maintenance of the website	4 will not	Cancel
user	delete my account	I can delete all my comments and posts instantly	4 will not	Cancel
user	create community for a certain entertainment type	certain content is concentrated to 1 area and easy to find	4 will not	Cancel
user	get the comment from other people with email	keep me update	4 will not	Cancel
user	send message to other user	users can communicate with each other	4 will not	Cancel
user	purchase premium membership	Avoid ads and earn customization options	4 will not	Cancel
user	add other users as community moderators	others can help manage busy forums	4 will not	Cancel

User Interface Requirements

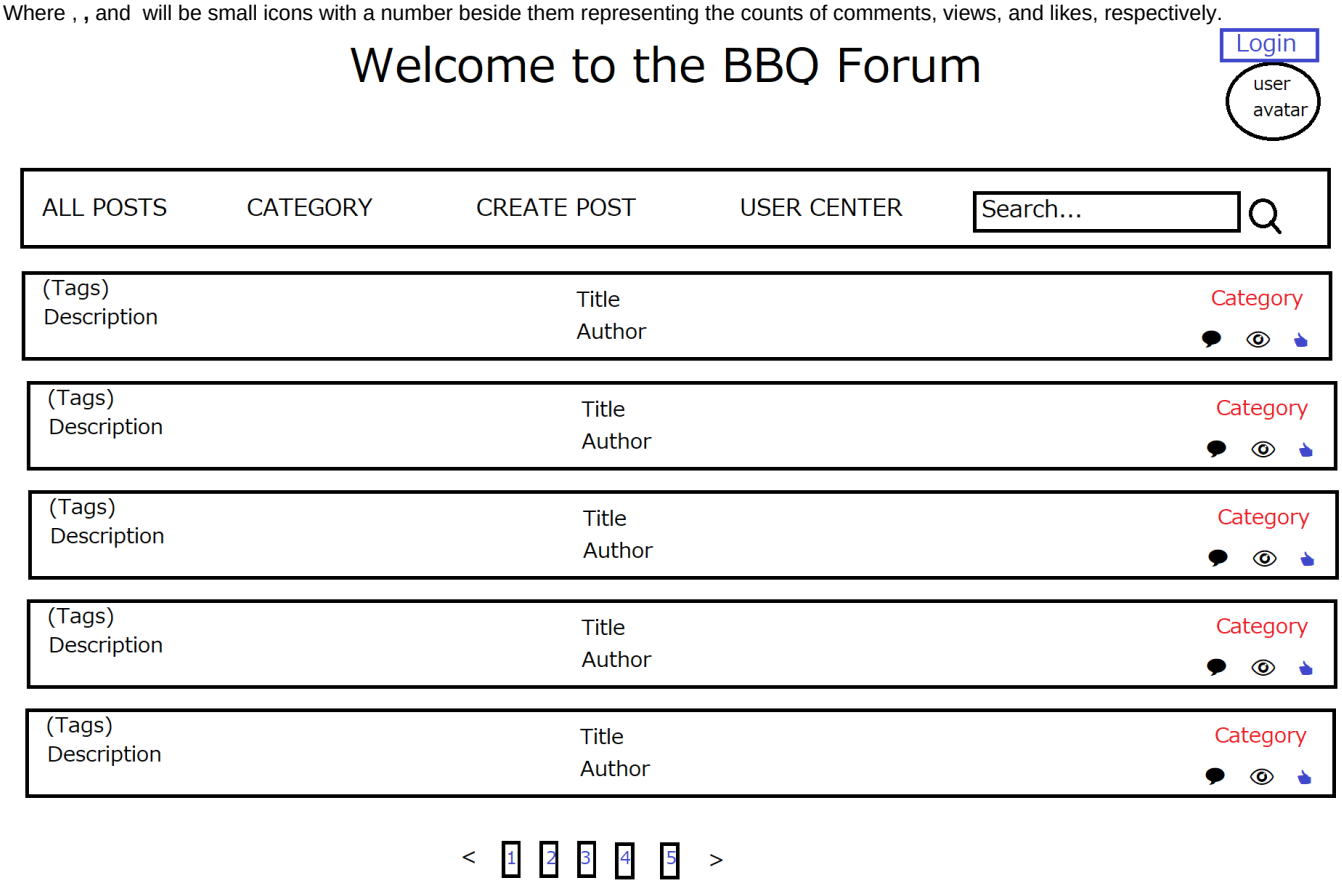
The user interface should have a simple layout with core areas of the page to allow for ease of access for users of the site. It is split into 3 columns, each acting as a major part of the site. The left most column has the website name and logo, along with the current top trending posts listed. The center column is for searching for content and contains a search bar and corresponding space to display results for a given query. The

right most column contains the personalized sections including targeted content that a similarity algorithm predicts the user will be interested in, as well as a small area used to log in/out and access account details. The user profile page and sign up / log in pages should be simple interfaces with new rows for each data entry point.



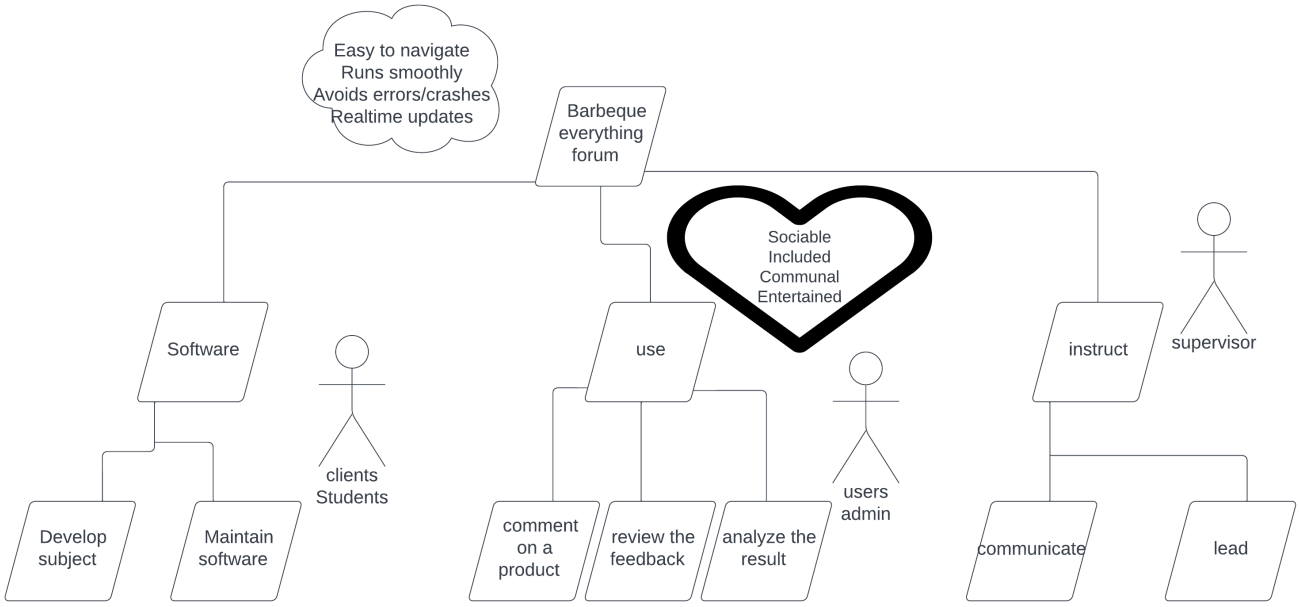
After discussing what will and will not be implemented, as well as restrictions regarding spacing, coloring, and general home page structure, a new, simpler design has been developed, emitting the previous top posts and suggested posts sections, focusing purely on the posts as they are sorted by the filtering systems.

Where , , and will be small icons with a number beside them representing the counts of comments, views, and likes, respectively.



Motivational Model

The motivational model for this project is as follows:



Who	Do	Be	Feel
Users	Use the product	Courteous	Connected
Admins	Monitor app	Attentive	Authoritative
Supervisors	Assist	Supportive	Helpful
Clients	Give requirements	Clear, concise	Satisfied
Students	Develop Software	Efficient	Accomplished

Meeting notes

Error rendering macro 'create-from-template' : Failed to find net.sf.hibernate.Session from the current thread

Incomplete tasks from meetings

Description	Due date	Assignee	Task appears on
A new repository was set up for this case just in case			2022-10-13 Meeting notes
@ gin is going to be in charge for the testing plan later especially on the unit test part		gin	2022-08-30 Meeting notes
@ Ke HU update the front end development schedule and start developing		Ke HU	2022-08-29 Meeting notes (framework)
@ Jiajun MAO update the new design diagram and re-organise the documentation		Jiajun MAO	2022-08-29 Meeting notes (framework)
@ Yuzhi Yan Update the new back end development schedule and start developing		Yuzhi Yan	2022-08-29 Meeting notes (framework)

☐ Issue the design of the project and the developing procedure. @ Jiajun MAO	Jiajun MAO	2022-08-23 Meeting notes
☐ @ gin python database management as well work together with @ Yuzhi Yan on back end	gin	2022-08-22 Meeting notes
☐ @ Ke HU learn how to do front end design	Ke HU	2022-08-22 Meeting notes
☐ @ Yuzhi Yan handling the requirement from back-end side	Yuzhi Yan	2022-08-22 Meeting notes
☐ @ Jiajun MAO Document the user story and requirement with @ Alex Ozoline together.	Jiajun MAO	2022-08-22 Meeting notes

Decisions from meetings

Page Title	Decisions
2022-08-15 Meeting notes (no clients)	We decide to develop the web app
2022-08-16 Meeting notes	Finish the user story first and then start to study Django.
2022-08-22 Meeting notes	@ gin Will mainly in-charge for the back end part of the project including the database and the final deployment of the whole project
	@ Ke HU Will mainly in-charge for the front end design for the project
	@ Yuzhi Yan Will mainly do the back end function implementation
	@ Jiajun MAO Will do the architecture design, procedure checking and documentation as well the trouble shooting for any issue happened during the development of the project.
	@ Alex Ozoline Will assist Ke in the front end part as well help Will with the document part of the project.
2022-08-23 Meeting notes	We r going to call our product BaeBecue-everything.
2022-08-29 Meeting notes (framework)	Using spring boot for the back end developing
	Using VUE for the front end developing

New team working rules apply right now

All meeting notes

Title	Creator	Modified
2022-10-18 Meeting notes	Jiajun MAO	Oct 18, 2022
2022-10-13 Meeting notes	Jiajun MAO	Oct 13, 2022
2022-10-11 Meeting notes	Jiajun MAO	Oct 11, 2022
2022-10-05 Meeting notes	Jiajun MAO	Oct 05, 2022
2022-10-02 Meeting notes	Jiajun MAO	Oct 01, 2022
2022-08-22 Meeting notes	Jiajun MAO	Sep 28, 2022
2022-08-25 Meeting notes	Jiajun MAO	Sep 28, 2022
2022-09-06 Meeting notes	Jiajun MAO	Sep 27, 2022
2022-08-30 Meeting notes	Jiajun MAO	Sep 27, 2022
2022-09-15 Meeting notes	Jiajun MAO	Sep 27, 2022
2022-09-13 Meeting notes	Jiajun MAO	Sep 27, 2022
2022-09-21 Meeting notes	Jiajun MAO	Sep 22, 2022
2022-08-29 Meeting notes (framework)	Jiajun MAO	Aug 29, 2022
2022-08-23 Meeting notes	Jiajun MAO	Aug 23, 2022
2022-08-18 Meeting notes	Jiajun MAO	Aug 18, 2022
2022-08-16 Meeting notes	Jiajun MAO	Aug 16, 2022
2022-08-15 Meeting notes (no clients)	Jiajun MAO	Aug 15, 2022

2022-10-18 Meeting notes

🗓️ Date

18 Oct 2022

👥 Participants

- @ Jiajun MAO
- @ Alex Ozoline
- @ gin
- @ Ke HU
- @ Yuzhi Yan
- @ Samuel Tumewa

🥅 Goals

- stand up
- demo presentation

🗣️ Discussion topics

Time	Item	Presenter	Notes
22:00	presentation	everyone	• working on the preparation of presentation
22:10	Trail presentation	everyone	feedback see below

✅ Feedback

deployment & delivery practise more

spend more time on live demo

talk more concisely

deployment part need to be more specific

no testing content more formal content

What is delivery? – like real, GitHub, instruction,

⤴️ Decisions

2022-10-13 Meeting notes

🗓️ Date

13 Oct 2022

👥 Participants

- @ Jiajun MAO
- @ gin
- @ Ke HU
- @ Alex Ozoline

🥅 Goals

- Get ready for the incidents in China like sudden quarantine request

🗣️ Discussion topics

Time	Item	Presenter	Notes
14:30	New repository	@ gin	• Vin is the previous admin of the previous one, but he might not be able to reply us on time because of the sudden accident in China which request him to be isolated in the given hotel by gorverment

✅ Action items

- [ ] A new repository was set up for this case just in case

⤴️ Decisions

2022-10-11 Meeting notes

🗓️ Date

11 Oct 2022

👥 Participants

- @ Jiajun MAO
- @ gin
- @ Alex Ozoline
- @ Ke HU
- @ Yuzhi Yan
- @ Samuel Tumewa

🥅 Goals

- feedback on the week9 checklist
- general schedule for sprint 1

🗣️ Discussion topics

Time	Item	Presenter	Notes
22:04	Project feedback	@ Samuel Tumewa	• see below

✅ Action items

Update the motivational model with the latest user story

Update the Front end design with more specific design includes the colour and other details

Update the Back end design with more specific functions integrated into the design diagram

More communicate evidence need to be shown in the discord

More testing plan on different functionalities need to be updated to the testing plan document

More commits wanted on Github, with pull requests being applied

Auto CI/CD integrated with Github needed to be considered

Simplify the confluence document

⤴️ Decisions

2022-10-05 Meeting notes

🗓️ Date

05 Oct 2022

👥 Participants

- @ Jiajun MAO
- @ Alex Ozoline
- @ gin
- @ Ke HU
- @ Yuzhi Yan

🥅 Goals

- Sharing some feedback on sprint0
- Sharing what had happened in last few weeks
- get some general ideas

🗣️ Discussion topics

Time	Item	Presenter	Notes	
22:00		Jiajun Mao	General feedback and review of sprint 0	
22:10		Alex Gin Yan ke Jiajun	Front-end: Currently evaluating and optimizing the ui effect of user operation, beautifying the color, layout, etc. of the ui, and at the same time increasing the operation animation effect displayed on the front-end, enhancing the user experience, system fluency and user experience workload, combined with the sprint0 feedback to make changes Back end: 1. Normal coding, evaluation on other functionalities, API documentation update, development log on back end	

✅ Action items

⤴️ Decisions

2022-10-02 Meeting notes

🗓️ Date

02 Oct 2022

👥 Participants

- @ Jiajun MAO
- @ Alex Ozoline

🥅 Goals

- Finalise the Sprint 0 review report
- List all the bugs that is discovered during the review session

🗣️ Discussion topics

Time	Item	Presenter	Notes
12:30am	Sprint0	@ Alex Ozoline @ Jiajun MAO	• Finalise the sprint 0 report • double check all the documentation
01:30am	debugging	@ Jiajun MAO @ Alex Ozoline	• list all the issues with the platform

✅ Action items

⤴️ Decisions

2022-09-21 Meeting notes

🗓️ Date

21 Sep 2022

👥 Participants

- @ Jiajun MAO
- @ Ke HU
- @ Yuzhi Yan
- @ gin
- @ Alex Ozoline

🥅 Goals

- Weekly standup for this week
- sharing the current progress of coding
- Trouble shooting

🗣️ Discussion topics

Time	Item	Presenter	Notes	
22:00	Confluence & design	Will Alex	We have finished the document of design on deployment aspect. More editing will be added to confluence later	
22:12	Deployment	Gin	issue regards to backend in deployment	
22:22	Development	Ke, Yan	Work on those bugs that can potentially causing issues	

✅ Action items

⤴️ Decisions

2022-09-21 Meeting note

The code has serious issue need to be fixed

1.public List<Admin> login(String username) {
Session session = getSession();
String sql = "from Admin admin where admin.userName=?";
Query query = session.createQuery(sql);
query.setString(0, username);
List list = query.list();
session.flush();
session.close();
return list;
}

If there are more than one user in the database, then it will be a list and I don't know which user is logged in.
You should not find a list of users when logging in, you should only find one user, because the user name is unique.
Modify the return value to Admin to make it easier to operate.

2.public void publish(Notice notice){
Session session = getSession();
Transaction transaction = session.beginTransaction();
session.save(notice);
transaction.commit();
session.flush();
session.close();

}
Now the problem is that when posting, I don't know who posted the notification, which causes the user to try to find the user who posted it.
When posting a notification, you need to put the login user's id inside the notification to indicate who posted this notification.

3.String queryString = "from MainForum as model where model."
+ propertyName + "= ?";
Session session = getSession();
Query queryObject = session.createQuery(queryString);
manipulating data in a way that is too old.
For example, writing sql to java code inside may lead to code confusion and sql is not well maintained.
Later to change to a more convenient way to manipulate the database.

4.The checks inside the code are not very complete, and some checks are not yet completed.

2022-09-15 Meeting notes

🗓️ Date

15 Sep 2022

👥 Participants

- @ Jiajun MAO
- @ gin
- @ Ke HU
- @ Alex Ozoline
- @ Yuzhi Yan

🥅 Goals

- confirming the the target for the next turns on the design of the project
- test and review for sprint 0 need to be finished at next week.

🗣️ Discussion topics

Time	Item	Presenter	Notes
13:25	Deployment	Gin	• once the backend and frontend is finished then the deployment shall begin
14:30	Design	Yuchi Yan Ke Hu	• design and code standard will be published next week, with the basic functions of sprint 0 available
14:52	Motivational model	Alex Ozoline Jiajun Mao	• Motivational model has been published and the structure of the confluence has been edited. Feedback will be followed.

✅ Action items

- [ ] Backend design and code standard need to be finished by the end of next Tuesday @ Yuzhi Yan
- [ ] Frontend design need to be finished by the end of this weekend. @ Ke HU

- [ ] General design updates need to be finished by next Tuesday @ Jiajun MAO
- [ ] Home page of confluence need to be more specific @ Alex Ozoline
- [ ] @ gin Deployment need to be ready for the next Tuesday.

Decisions

2022-09-13 Meeting notes

Date

13 Sep 2022

Participants

- @ Jiajun MAO
- @ Ke HU
- @ Alex Ozoline
- @ Yuzhi Yan
- @ gin
- @ Samuel Tumewa

Goals

- week6 checklist feedback

Discussion topics

Time	Item	Presenter	Notes
22:00	Progress and documentation	@ Jiajun MAO @ Alex Ozoline	• Progress check and checklist fixing and procedures for the backend development
22:05	Front end	@ Ke HU	1. Front-end: At present, the modular development of JS and CSS has been completed. I am currently working on the last stage of component-based development, developing the pages of the client and the administrator in layers, which is convenient for maintenance, and the interface of the administrator is basically completed. Currently, I am developing the relevant data pages of the client. After the development is completed, the front-end and back-end debug and test the system as a whole. After the debugging and test, my front-end work is completed.
22:07	Back end	@ Yuzhi Yan	1. finished almost interfaces and the basic interaction with data is working, but still exceptions need to be handled, and more test may be applied next week.
22:09	Deployment	@ gin	docker will be used for the deployment stuff
22:11	Areas of Improvement	@ Samuel Tumewa	1. Clean up confluence pages. 2. Improve code structure and clearly define coding standards. 3. Avoid adding excess amounts of files. 4. Add more prototypes particularly for front end to aid in the tracking of development. 5. Communicate via discord or other mediums more consistently and lend aid to other members when needed.

Action items

- [ ] Front end design (and diagram) @ Ke HU @ gin
- [ ] Back end design document @ Yuzhi Yan @ gin
- [ ] General design document and diagram @ Alex Ozoline @ Jiajun MAO
- [ ] Motivational modle @ Alex Ozoline @ Jiajun MAO

⤴️ Decisions

2022-09-06 Meeting notes

🗓️ Date

06 Sep 2022

👥 Participants

- @ Jiajun MAO
- @ gin
- @ Ke HU
- @ Alex Ozoline
- @ Yuzhi Yan
- @ Samuel Tumewa

🥅 Goals

- weekly updates on the current progress report

🗣️ Discussion topics

Time	Item	Presenter	Notes
22:04	Deployment	Bucheng Liu	• research on the deployment part of the project, research on the unit test.
22:06	Functional test	Alex Ozo	• research on the scope of the functional test.
22:10	Backend	Vin Yan	• database module, normal coding and paper work
22:15	Front end	Ke HU	• Front end development
22:18	Procedures and workflow	Will	• make sure everything is on track and provides trouble shooting

✅ Action items

- [ ] front end and back end development schedule needs to be upload @ Jiajun MAO @ Ke HU @ Yuzhi Yan

⤴️ Decisions

2022-08-25 Meeting notes

🗓️ Date

25 Aug 2022

👥 Participants

- @ Jiajun MAO
- @ Samuel Tumewa
- @ gin
- @ Yuzhi Yan
- @ Ke HU
- @ Alex Ozoline

🥅 Goals

- the design frame work and the template for front end
- user authorisation

🗣️ Discussion topics

Time	Item	Presenter	Notes
14:20	User authorisation	@ Jiajun MAO @ samuel.tumewa @ Yuzhi Yan	• It is suggested to use third party tools for the verification for safety pouppose
14:50	User interface	@ Ke HU	• the general idea of user interface for front end.

✅ Action items

- [ ] finish the user interface requirement document @ Alex Ozoline
- [ ] @ Jiajun MAO finish the design of the project by the end of this week.

⤴️ Decisions

2022-09-03 Meeting notes

🗓️ Date

03 Sep 2022

👥 Participants

- @ gin
- @ Yuzhi Yan

🥅 Goals

- Since the theme of the project has been settled up, to connect with this topic: “Barbeque Everything” and improve the quality and productivity of the development, an alteration of database design has been made in this meeting

⤴️ Decisions

- Change name of data tables

- Delete manager table and merge into user table
- Add create_time and update_time to each tables
- Replace black_list table by using state attribute in user table
- Create separate tables for tags, operation logs and messages

2022-08-30 Meeting notes

🗓️ Date

30 Aug 2022

👥 Participants

- @ Jiajun MAO
- @ Ke HU
- @ gin
- @ Yuzhi Yan
- @ Alex Ozoline
- @ Samuel Tumewa

🥅 Goals

- weekly stand up on the current process

🗣️ Discussion topics

Time	Item	Presenter	Notes
22:00		@ Jiajun MAO	• share the progress of week 6 checklist
22:10		@ Alex Ozoline	• document and summarise the weekly progress
22:15		@ Yuzhi Yan	• progress on backend
22:20		@ Ke HU	• progress on frontend, using Ajax
22:25		@ gin	• research on the backend

✅ Action items

- [ ] @ gin is going to be in charge for the testing plan later especially on the unit test part
- [x] @ Alex Ozoline can help on the function tests later.

⤴️ Decisions

2022-08-29 Meeting notes (framework)

🗓️ Date

29 Aug 2022

👥 Participants

- @ Jiajun MAO
- @ Ke HU
- @ Yuzhi Yan

🥅 Goals

- Updates on the changing of the architecture and the framework of the project
- Updates on the rule of group work

🗣️ Discussion topics

Time	Item	Presenter	Notes
21:00	Group work model	@ Jiajun MAO	• Everyone is in charge for the part that is assigned to him, but also be able to ask someone who is free to help and the one can't refuse the help without a reasonable reason hence to balance the workload
21:15	Backend framework update	@ Yuzhi Yan @ Ke HU	• sprint boot + VUE are used as the front end and back end rather than the Django. Spring boot and VUE can provide a seperate working environment for front end and back end that can allow our team to work at the same time. A separated model also have a higher flexibility on each part of it and decrease the cost of time on the developing and maintenance compare with the Django which has a integrated model on front end and back end. • Also due to the high cohesion of the Django, it is difficult to apply the third party library like Spring Boot. The built-in ORM of Django is also not as powerful as Mybatis. • Spring boot is also a Java based language framework, and we have just finished the SWEN30023 SMD which is a Java based course, so all of us are more familiar with Java on the advanced developing as well when comparing it with Python. • VUE also provides a good connection to Spring Boot. It is a js based framework which means it is more widely used in the real world developing environment, hence it also decreases the degree of difficulty when doing front end development for us as none of us has any experience in the front end developing before. • the combination of the VUE and spring boot has been widely used these days which provides a more stable experience and it is easy to study and manage.

✅ Action items

- [ ] @ Yuzhi Yan Update the new back end development schedule and start developing
- [ ] @ Ke HU update the front end development schedule and start developing
- [ ] @ Jiajun MAO update the new design diagram and re-organise the documentation

⤴️ Decisions

- Using spring boot for the back end developing
- Using VUE for the front end developing
- New team working rules apply right now

2022-08-23 Meeting notes

🗓️ Date

23 Aug 2022

👥 Participants

- @ Jiajun MAO
- @Sam
- @ gin
- @ Yuzhi Yan
- @ Alex Ozoline
- @ Ke HU

🥅 Goals

- Further question from yesterday need to be confirmed with supervisor:
    1. feedback? compulsory?
    2. design?
    3. decision making ?
    4. week 6 testing ?
- get ready for the week 6 assessment
- progress report for each of the team member

🗣️ Discussion topics

Time	Item	Presenter	Notes
22:10	Late	Sam	• A kind reminder that everyone should try to be on-time to save time.
22:15 Asking all the question for the week 6 checklist	Week 6 check_list	Will	Asking all the question for the week 6 checklist, during the marking process that the online server need to be available for access 7*24.
22:20	Progress report	Will	User story has been finished focus more on the design and architecture of the project later doing more documentation with Alex tonight.
22:22	Progress report	Gin	Database has been set, focus more on the back end and database solutions and providing trouble shooting to front end and back end developer. Going to do the deployment part of the project at the final stage
22:23	Progress Report	Alex	Finish the internal progress logbook and finish the user story table with Will together, going to do more on the documentation part later
22:25	Progress report	Yan	Finish the initial of the framework which allow the basic creating and browsing of the forum. Going to do the authentication of the user.
22:27	Progress report	Ke	Finish the initial of the framework for the front end part. Going to do the design of UI.

✅ Action items

- [ ] Issue the design of the project and the developing procedure. @ Jiajun MAO

⤴️ Decisions

- We r going to call our product BaeBecue-everything.

2022-08-22 Meeting notes

🗓️ Date

22 Aug 2022

👥 Participants

- @ Jiajun MAO
- @ Alex Ozoline
- @ gin
- @ Ke HU
- @ Yuzhi Yan

🥅 Goals

- Job assignment
- Week 6 progress report

🗣️ Discussion topics

Time	Item	Presenter	Notes

✅ Action items

This is a task list for us to finish before the beginning of the next tutorial class meeting.

- [ ] @ Ke HU learn how to do front end design
- [ ] @ Yuzhi Yan handling the requirement from back-end side
- [ ] @ gin python database management as well work together with @ Yuzhi Yan on back end
- [ ] @ Jiajun MAO Document the user story and requirement with @ Alex Ozoline together.
- [ ] @ Jiajun MAO @ Alex Ozoline finish the document for week 6 progress check
- [/] @ Alex Ozoline doing some research on python front end and assist @ Ke HU
- [ ] @ Jiajun MAO do the design of architecture and user story diagram as well get familiar with the developing process.
- [/] @ Alex Ozoline go through the documents on the confluence, write the log for whatever has been finished by us so far.

⤴️ Decisions

- @ gin Will mainly in-charge for the back end part of the project including the database and the final deployment of the whole project
- @ Ke HU Will mainly in-charge for the front end design for the project
- @ Yuzhi Yan Will mainly do the back end function implementation

@ Jiajun MAO Will do the architecture design, procedure checking and documentation as well the trouble shooting for any issue happened during the development of the project.

@ Alex Ozoline Will assist Ke in the front end part as well help Will with the document part of the project.

2022-08-18 Meeting notes

🗓️ Date

18 Aug 2022

👥 Participants

- @ Jiajun MAO
- @ Ke HU
- @ gin
- @ Yuzhi Yan
- @ Alex Ozoline (Will be informed later)

🥅 Goals

- decide the priority of user story
- assign the job to team member

🗣️ Discussion topics

Time	Item	Presenter	Notes

✅ Action items

settle the architecture of the whole project on virtual machine

⤴️ Decisions

2022-08-16 Meeting notes

🗓️ Date

16 Aug 2022

👥 Participants

- @ Jiajun MAO
- @ Alex Ozoline
- @ gin
- @ Yuzhi Yan
- @ Ke HU

🥅 Goals

- user story

- job assignment on the architecture

🗣️ Discussion topics

Time	Item	Presenter	Notes
15min	User story	All	• More suggestions should be mentioned on how this forum is going to react with different roles. Even if any of the member make any mistakes is fine as the whole group is going to have that part of the job verified later on Thursday's tutorial and we are going to discuss about the priority as well.
10min	Job assignment	All	• Since week 6 is coming so everyone should focused on user story first. Gin, Vin and Ke is mainly going to focused on the coding and architecture part while Alex will help with coding standard and user interface design. Will is going to focus on the coding part majorly and be a backup resources for any other cases as he is good at both coding and standardising.

✅ Action items

Django has been decided as the framework for this project hence everyone should start to study the document of the Django as soon as possible. everyone should post any questions regards to that in the group chat.

⤴️ Decisions

Finish the user story first and then start to study Django.

2022-08-15 Meeting notes (no clients)

🗓️ Date

15 Aug 2022

👥 Participants

- @ Jiajun MAO
- @ Vin
- @ Gin
- @ HU
- @Sam
- Alex (not available, will read this document later.)

🥅 Goals

- Discuss about the plan for no clients situation.
- What should we do for the further development.

🗣️ Discussion topics

Time	Item	Presenter	Notes
		All	• Since no responds from clients after a large number of emails been sent to him, after we get the approval from our supervisor
		Sam	• Web app is suggested for the development

✅ Action items

⤴️ Decisions

We decide to develop the web app

The forum is suggested by Bucheng Liu as a product for us to develop. He will also be in charged for the final settle of our service on server.

Since there is no actual client, so everyone can carry out their own ideas on the product and Will and Alex is in charge for the recording and collection of the document.

Since Bucheng has his own server so we decide to deploy on his own server. Further more, a new virtual machine will be created by him for the developing and testing purpose. We appreciate his contribution.

General Design Pages

These are the generally agreed upon designs for the back and front end.

The front end contains designs for website layout including the home page, account registration, and post creation.

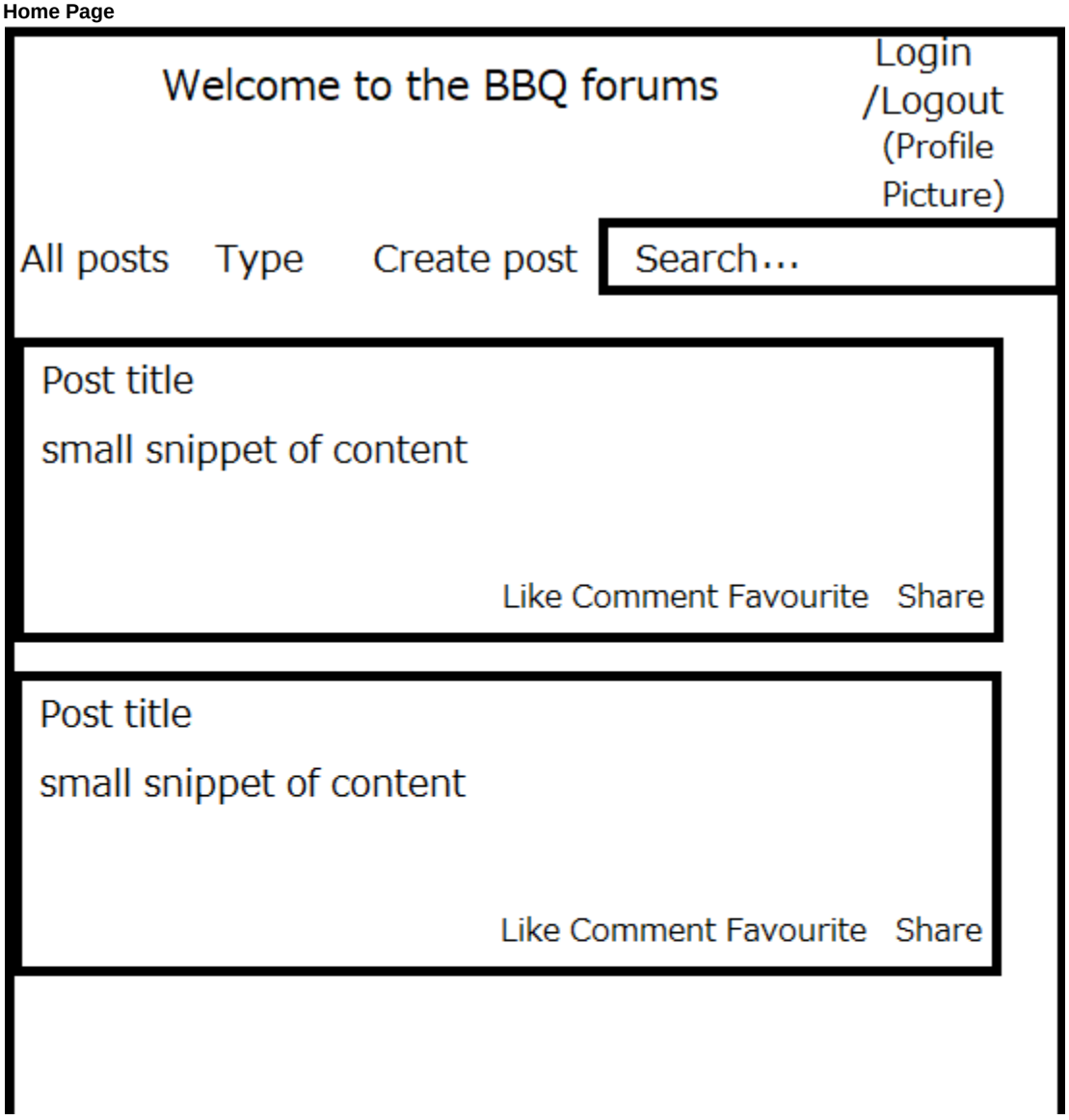
The back end contains design ideas for the database structure, and sequence diagram, which briefly introduces the logic behind how the system functions.

Front end design - Front end design specification

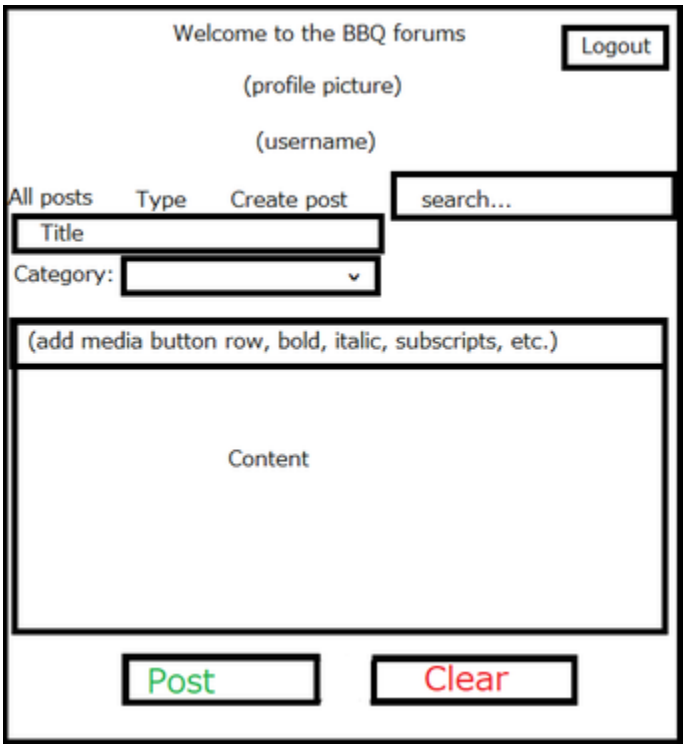
Back end design - General architecture design of project

Front end design specification

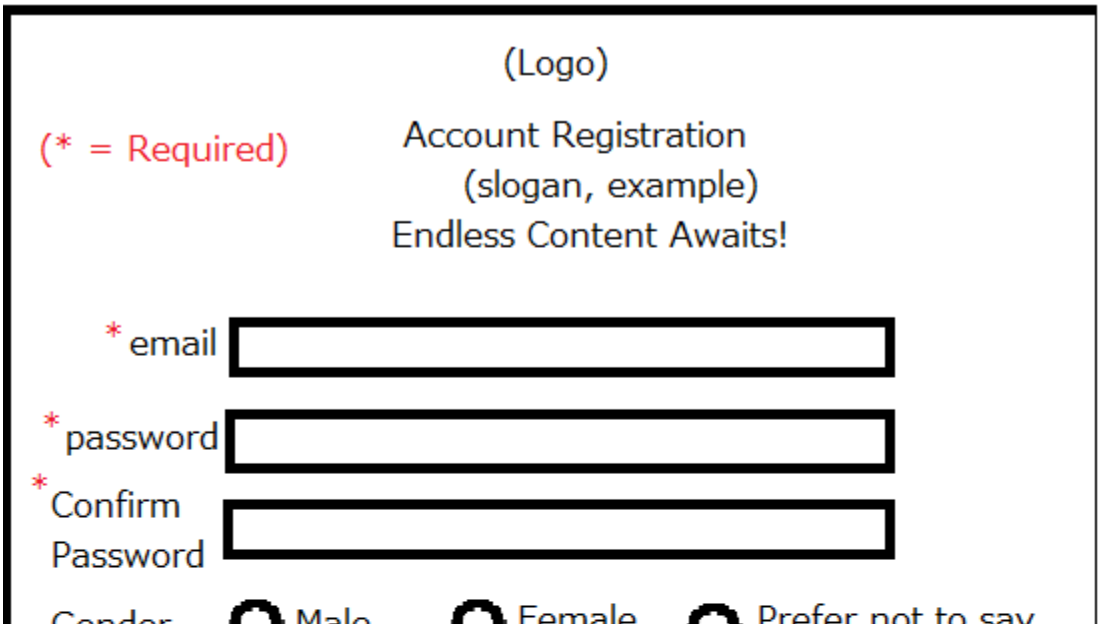
Home Page

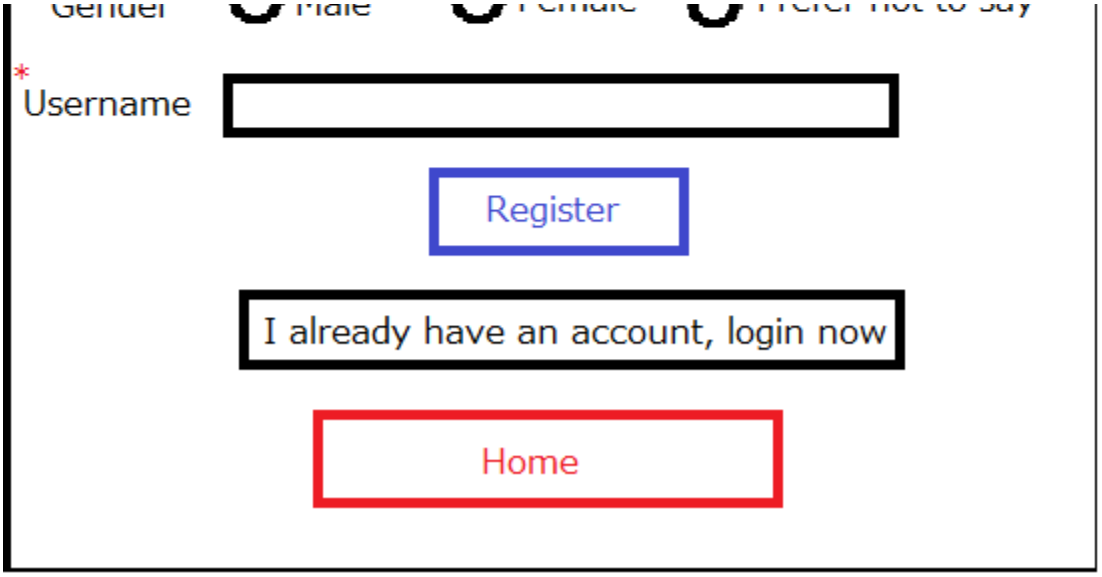


Post creation



Account creation

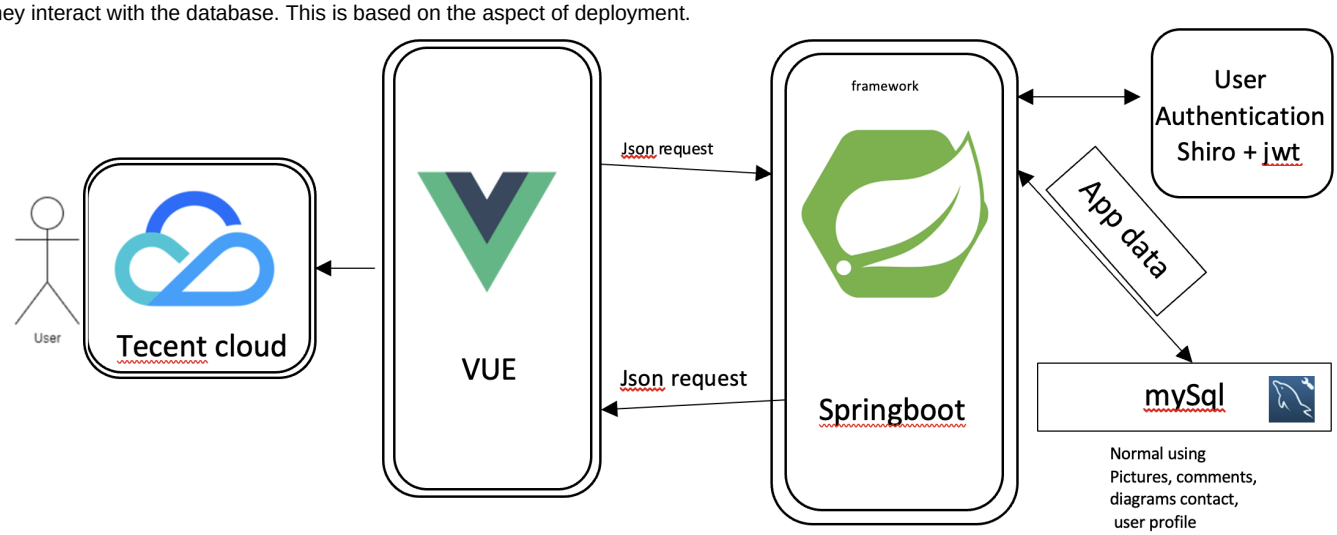




These are guidelines subject to alter as more changes are made.

General architecture design of project

This diagram indicates the general design of this project by explaining the relationship between VUE, springboot and tecent cloud and the way they interact with the database. This is based on the aspect of deployment.



🟦 Key points

Here is something might need to be changed later

1. It could be possible for us to replace the json request to other format because base on the previous tests there are some issue in the communication between front end and back end
2. Currently we considering 2 separate database for different purpose and we need to encrypt the user authentication database. this may leads to the slow response. We are currently using Shiro + jwt for user authentication and mysql for app data storage.

ℹ️ this design is also a semi-complete version, and might need to be edited later

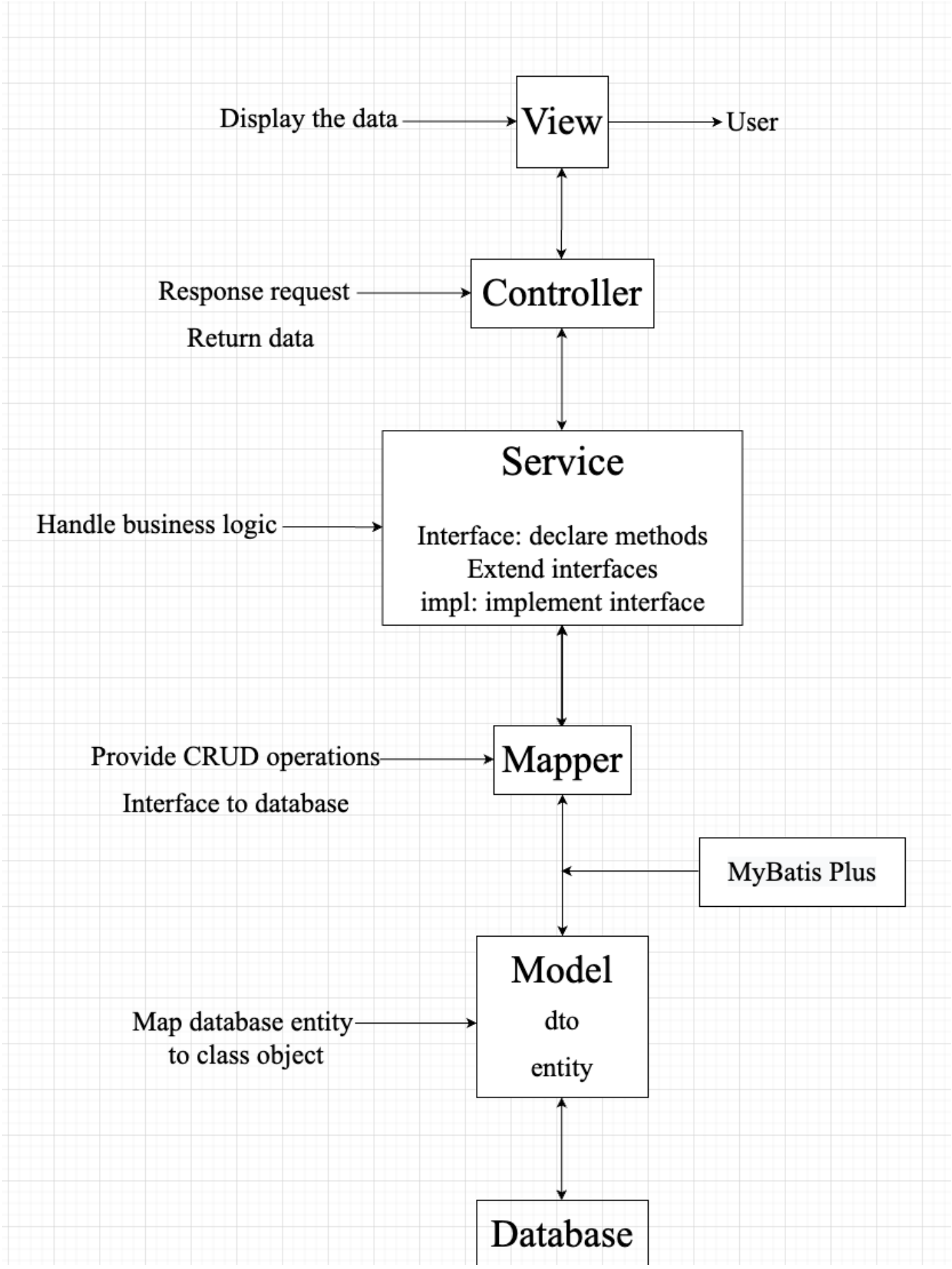
📋 Related articles

here is the link to the design draft of the previous version.

https://lucid.app/lucidchart/1582e1e7-196c-43fd-bd68-ac6b242491a0/edit?page=0_0&invitationId=inv_1c53ccb1-eeb9-491b-8dbd-970b0359ecfe#

📋 Specific Code structure

The code structure of back end in this project



Coding Standard

Indentation

The most important aspect of indentation is consistency within each class

Additionally, some form of indentation must be used to ensure code readability and coherence.

- Newline after each brace open('{'). E.g. function definitions, loops, if statements.
- Following each brace there should be an indent of 1 tab and any additional lines inside the braces must also posses the same indent.
- Ensure a space is present after placing a comma between two function arguments.

Length

- Ensure that lines of code do not go to an extreme length, there is no set limit but if a line feels too long it probably is.
- Simply go to a new line and indent 1 tab to continue the line.
- Avoid excessively lengthy functions to prevent difficult to functions, make them perform small amounts of work and combine them together in separate places.

Documentation

- Comments should be present at regular intervals to explain the main functionalities of each function defined.
- This includes descriptions for each input variable, as well as what the function returns (if applicable).
- No use of ‘magic numbers’ without a clear comment explaining the usage of the number.
- Include a doc string for each main class or function.
- Include a short description that states how to use a certain function to achieve something (to be collected in a manual for later).

Naming

- Variable and function names should be meaningful and help the reader understand their purpose in the code
- Classes should begin with a capital letter and follow camel case rules
- Variables and functions should begin with a lower case letter and also be in camel case
- Avoid using excessive global / static variables

Other advice to follow

- Ensure that code functions even when provided with malicious input by a user
- Don’t delete/edit other’s coding without discussing the intended changes with them beforehand
- Record large changes or errors on confluence inside the respective development logs.
- Make regular commits with a meaningful commit message describing your chanages/additions

Example of code that follows the standards

```
public class UserInterceptor extends AbstractInterceptor {
  /**
   * This is an example of a docstring that would describe how the
function works
   * This function recieves a username / admin username
   * and tries to log them in to the site
   */
        @Override
        public String intercept(ActionInvocation arg0) throws Exception
{
                String username = (String) ActionContext.getContext().
getSession().get("username");
                String adminName = (String) ActionContext.getContext().
getSession().get("adminName");
                if (username == null && adminName == null){
                        //Fail to login
```

```
                        System.out.println("Failed");
                        // Go back to login page
                        return "login";
                }
                else {
                                //Successful
login
                                return arg0.invoke();
                }
        }
}
```

Maintenance and Hand Over

This part of the documents will mainly record the maintenance and hand over process of this project

Maintenance:

The Auto deployment function has been enabled. See below for the latest deployment instruction with auto deploy enabled.

1. deploy the front end.
   a. push the latest update
   b. edit the local hots of the following files to 43.159.34.72

axios.js baseurl

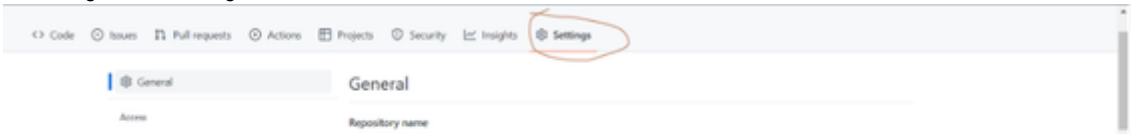
UserEditDia.vue action

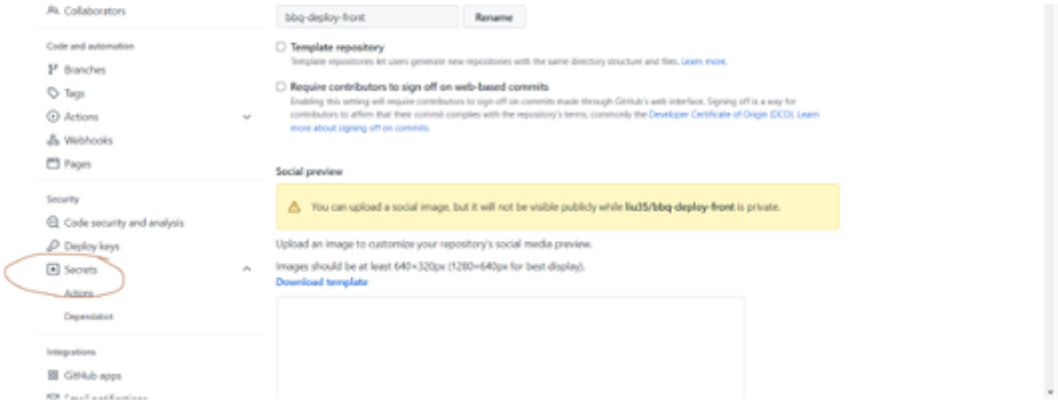
Register.vue action

2. edit the Github settings

   a. enter the repository of front end, go for settings and set up the hidden settings

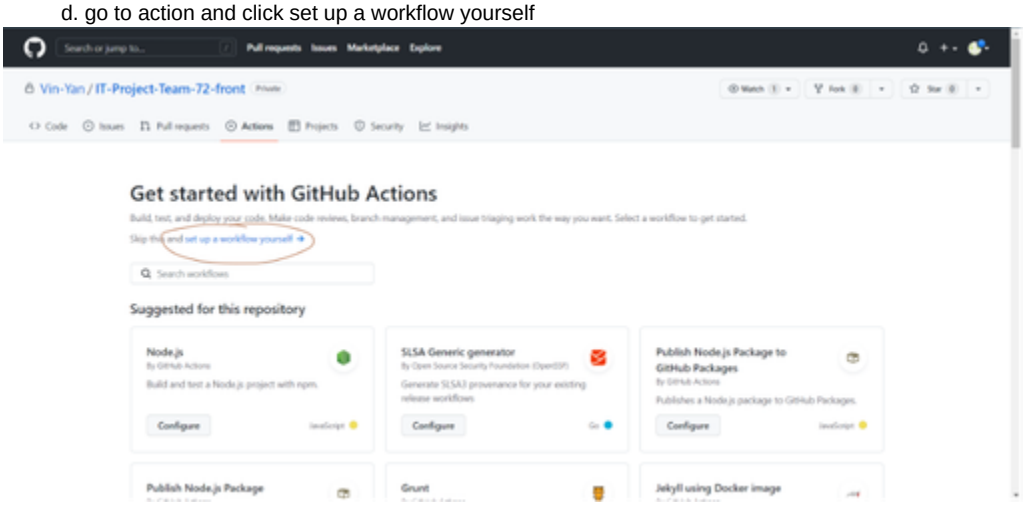
   b. go to the settings > secrets > actions





c. add the host and password with the matching ip and password

d. go to action and click set up a workflow yourself



e. copy the following information

```
name: CI

on:
  push:
    branches: [ main ]


jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout
      uses: actions/checkout@v2.5.0

    - name: install nodeJs
      uses: actions/setup-node@v2.5.1
```

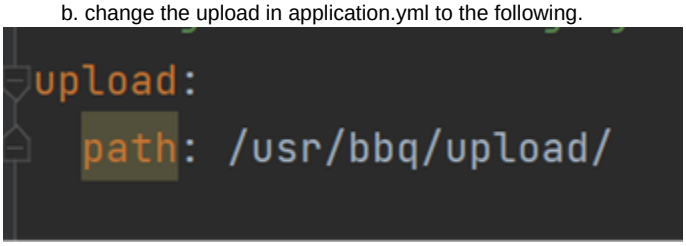
```
    with:
      node-version: "14.X"

- name: install deps
  run: |
    cd bbq-fore
    npm install

- name: build app
  run: |
    cd bbq-fore
    npm run build

- name: copy dist file with scp
  uses: appleboy/scp-action@master
  with:
    host: ${{secrets.HOST}}
    username: root
    password: ${{secrets.PASSWORD}}
    port: 22
    source: "bbq-fore/dist"
    target: "/root"
    rm: false

- name: deploy fore
  uses: appleboy/ssh-action@master
  with:
    host: ${{secrets.HOST}}
    username: root
    password: ${{secrets.PASSWORD}}
    script: |
      cp -r /root/bbq-fore/dist /root/nginx
      rm -rf /root/bbq-fore
      rm -rf /root/nginx/html
      cd /root/nginx/
      mv dist html
      cd /home/docker/
      docker-compose restart nginx
```

f. enter the following information and hit submit

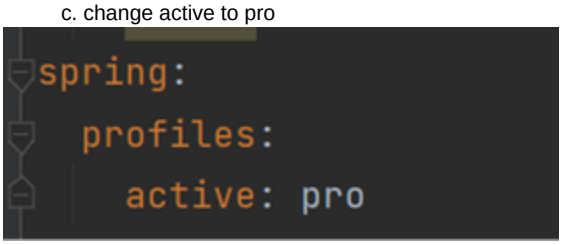
3. Back end deployment:

a. repeat the same operation for the secrets settings.

b. change the upload in application.yml to the following.



c. change active to pro



e. repeat the following operation, with the different config files. Copy and paste the text box content and implement to the GitHub.

```
name: CI

on:
  push:
    branches: [ main ]


jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout
      uses: actions/checkout@v2.5.0

    - name: Set up JDK 1.8
      uses: actions/setup-java@v3
      with:
        java-version: 8.0.345+1
        distribution: 'adopt'

    - name: cache maven dependences
      uses: actions/cache@v2
```

```
    with:
      path: ~/.m2/repository
      key: ${{ runner.os }}-maven-${{ hashFiles('**/pom.xml') }}
      restore-keys: |
        ${{ runner.os }}-maven-

- name: compile
  run: |
    cd bbq-back-end
    mvn compile

- name: package
  run: |
    cd bbq-back-end
    mvn -B package --file pom.xml -Dmaven.test.skip=true

- name: copy jar file with scp
  uses: appleboy/scp-action@master
  with:
    host: ${{secrets.HOST}}
    username: root
    password: ${{secrets.PASSWORD}}
    port: 22
    source: "bbq-back-end/target"
    target: "/root"
    rm: false

- name: deploy back end
  uses: appleboy/ssh-action@master
  with:
    host: ${{secrets.HOST}}
    username: root
    password: ${{secrets.PASSWORD}}
    script: |
      rm -rf /home/docker/bbq-1.0.0-SNAPSHOT.jar
      cp -r /root/bbq-back-end/target/bbq-1.0.0-SNAPSHOT.jar /home/docker/
      rm -rf /root/bbq-back-end
      cd /home/docker/
      docker-compose kill bbq
      docker-compose rm -f
```

```
docker rmi -f $(docker images bbq:latest -q)

docker-compose up -d --no-recreate
```

4. if any issue popup during the process, re-run it. Bugs do appear.

Knowledge base management & Documentation

1. Please remember to write step-by-step indication here about the software that you have designed
2. don't delete other people's code before asking

Quick links to external tools and internal material for quick start:

User story: https://docs.google.com/spreadsheets/d/1BHR9LcNoEyqS7PgytB-nVLzf6WqQ4P8DjbZbEJL4I7k/edit?skip_itp2_check=true#gid=0

Design of general idea: https://lucid.app/lucidchart/1582e1e7-196c-43fd-bd68-ac6b242491a0/edit?page=0_0&invitationId=inv_1c53ccb1-eeb9-491b-8dbd-970b0359ecfe#

Specific design of architecture General architecture design of project

Database module: Design of the database

Spring Boot document: https://spring.io/projects/spring-boot/

VUE document: https://vuejs.org/guide/introduction.html

Motivational Model: Motivational Model

Job assignment: 2022-08-22 Meeting notes

VM use guide

Use ssh software to login to the VM

Ip address: gin-and-tonic.asuscomm.com

port number: 72

account name: team72

password:team123

Git Command Dictionary

git clone

1. It is common to join a team with an ongoing project which does source code management using Git. We use the git clone command to get a copy of existing code stored on an external Git server.
2. Browse to the directory you want to set up our project, run command to copy the existing code. command: **git clone** https://github.com/Vin-Yan/IT-Project-Team-72.git
3. Note that a new directory has been created with the name IT-Project-Team-72 . Browse

   to this directory, view files. Use the **git status** and **git log** commands to note that the full

   history of the repository has been downloaded.
4. Alternatively you could just download the GitHub Desktop app in your PC and directly clone the project without typing command.

git pull

1. The command is used to fetch and download content from a remote repository and immediately update the (current) local repository to match that content
   Command: **git pull origin branch_name**

A file can exist in three stages in a Git repository: Working, Staging and Commit.

- Any changes not explicitly added to the Staging area reside in the Working Directory.
- Changes can be added to the Staging Area (also known as Index) using the **git add** command. This marks the changes that should go into the next commit
- The **git commit** command adds the marked changes from the step above to the (local) repository

git add

1. If you have modified multiple files and want to add all of them to the staging area at once, you can use command: **git add .** (don't forget the dot)

2. For single files using command :**git add filename**

git commit

After adding our changes to the staging area. Committing the changes stores a snapshot of the current state of the files. This allows you to roll back to this version if needed in the future. Note that this snapshot is stored *locally* on your machine.

1. Command: **git commit -m "commit message"**

   Every commit needs to be accompanied by a commit message, which is what we specified using

   the **-m** parameter.

2. Inspect the current status of the repository using command: **git status**

3. You can also combine the add and commit steps in one operation using the

   command: **git commit --all -m "commit message"**

   (Note that this combined operation only commits changes to files added earlier, and does not commit **new** files.)

git push

1. Note that every git operation up to now has been done *locally*. That is, our changes to the files have

not been sent to a remote server.

2. Push your changes to the server by using the command: **git push**

3. You do not need to push after every commit.

It is appropriate to push after every major change (e.g. a new feature, a bug fix, some refactoring etc.)

git branch

1. Git allows you to organize code in branches. For example, you might create a new branch for a new

   feature of your application. You can then keep continued development on the core of the application separate from the new feature development.
2. When initializing the repository, we specified the default branch to be named **'main'**

   Use the **git branch** command to list the current branches.

   (The currently active branch is shown with an asterisk) ---> * main
3. The most common way to create a new branch in Git is to create a new branch and switch to it in

   the same command.

   Command: **git checkout -b new-feature** or **git switch -c new-feature**
4. Switch back to the main branch.

   Command: **git checkout main** or **git switch main**
5. Push new branch to remote:
   Command: **git push -u origin new-feature**
6. Listing all branch
   Command: **git branch -a**
7. Delete a remote branch
   Command: **git push origin -d new-feature**
8. Delete a local branch (need to switch to another branch)
   Command: **git branch -D new-feature**
9. When we have finished developing the feature, we will want to bring those changes into the main

   branch. We do that using the git merge command

   Command: **git merge new-feature**

git tag

1. Git is a Version Control System. It allows the labelling of different versions of code.

2. You can add a tag to a commit using
   command : **git tag -a v0.72 -m "Release version 72"**
   The **-a** parameter is the tag label or 'annotation'. The **-m** argument allows the addition a message to the tag.
3. Delete a local tag using command: **git tag -d tagName**
4. Delete a remote tag using command: **git push origin :refs/tags/<tagName>**
5. Show all local tags using command: **git tag**
6. Tags in Git need to be explicitly pushed. You can push a single tag using
   command: **git push origin v0.72**

   or all created tags using command: **git push --tag**

New testing platform for all

I just buy a new Tencent cloud server.

ip:`43.159.34.72`

username: root

password:Team123@

- ***if you have difficulty to login please contact me on discord***
- **NOTE: If you want to login Please leave a message on discord. And If you make any change on the server Please make a simple note. Thanks.**

Update on Tencent server

I just install Bt control platform on the server if you want to use the server you can use the Bt control platform directly.

username:odlgrpk5

password:2fa0f7a5

Internal progress tracking

- Week 1 25/07
- Formed the team

- Set up mediums for communication and task management (Discord, Jira, etc)
- Discussed each members strengths and weaknesses with regards to software design/development/implementation etc
- Week 2 01/08
- Created GitHub repository and invite everybody on the team
- Held meeting to delegate management tasks among group members and formulate overall project plan
- Attempted to contact client
- Selected task information to work on
- Organized upcoming meeting dates and recorded schedules to assist in future planning
- Week 3 08/08
- Decided to create own task as client is unresponsive
- Made decisions on which programming languages to employ for the task
- Stated product requirements and assigned priorities
- Set deadlines for each key phase of development
- Chose roles for each member to take up during the production process
- Week 4 15/08
- Created user story table - https://docs.google.com/spreadsheets/d/1BHR9LcNoEyqS7PgytB-nVLzf6WqQ4P8DjbZbEJL4I7k/edit?skip_itp2_check=true#gid=0
- Assigned priority levels to user story table
- Chose to use Django as python is the most comfortable language for most of the group, and it has in- built forum support
- Idea comes up to create forum page to discuss popular entertainment
- Implemented 0.0 model. Local forum page
- Added ability to create forum posts with a title, description, name and link to 3rd party website.
- Updated user story table with more features
- Week 5 22/08
- Agree upon the architecture of the project
- Chose potentially final name of product, "Barbeque Everything"
- Planned what the user interface should look like
- Added some back end functionalities such as logging in
- Week 6 29/08
- Finalized documentation leading up to the progress check
- Reviewed user stories, priorities.
- Changed platform to sprint foot + VUE for the front end due to a multitude of reasons further discussed in meeting notes on 29/08
- Partially implemented user authentication
- Week 7 05/09

- Created plan for Sprint0
- Working on tasks using Jira board
- Working on implementing additional user stories leading up to spring0
- Negotiations with front end regarding back end developments on developing the interface
- Week 8 12/09
- Bulk of implementation of website
- Discussion of deployment method (purchase server vs local host)
- Design ideas for main website layout, post editing interface, and login / registration plan
- Implement proper coding standard for back end and front end use
- Restructure confluence to be more easily traversed
- Continued to update dev logs
- Week 9 19/09
- Ran unit and integration testing, recording results
- Created new coding repositories to split front and back end
- Created deployment instruction document
- Purchased server to run the project on (http://43.159.34.72)
- Finished sprint 0

Front End Development

Contains all Information related to front end development of the project

1. User Interface RequirementsUser Interface Requirements + General Requirements Requirements
2. Prototypes and specific design Front end design specification
3. Development LogDeve log for front end
4. Code repository https://github.com/liu35/it-bbq

Deve log for front end

Below is a list includes all the development logs for front end sorted by date.

30.8.2022: 30 Aug 2022 DevLog of Front-end

6.9.2022: 6 Sep 2022 DevLog of Front-end

30 Aug 2022 DevLog of Front-end

Front-end: The first task I do is to design the UI sketch of the system, and then connect with the back-end for data interaction. Since the back-end decided to use the front-end and back-end separation method to develop, provide an API interface for me to interact with data. I decided to use Ajax now: Asynchronous javascript and xml (asynchronous javascript and xml), because of the advantages of Ajax: update page data without refreshing the page, improve user experience, and achieve asynchronous work.
var1: Request method get/post
var2: Requested backend program address
var3: asynchronous (true)/synchronous (false), optional parameter, default is true
var: There are two cases. If it is a get request, fill in null. If it is a post request, fill in the data to be sent to the backend.
== Send ajax request process

1. Create XMLHttpRequest object
2. Call the open method to prepare the ajax request
3. Call the send method to send the ajax request
   In this way, the cooperation between me and the back-end can be easily solved, and the user experience is also better. Currently, I am waiting for the back-end interface, and at the same time, the front-end UI design is completed. After the back-end interface is completed, I can start the front-end interface according to the specific interface. Formally developed.

6 Sep 2022 DevLog of Front-end

Front-end: The interface protocol was decided with the back-end last week, so I have been using ajax as the framework of the system and reserved the UI of the page. I mainly made several functions, such as function sendAjax() and function callback() The two functions can send ajax interface request and data callback. For the interface, I need to complete 4 stages.
Stage 1 lib/framework selection (react)
Stage 2 Simple build optimization
webpack packaging
Stage 3 JS, CSS modular development
Stage 4 Componentized development. At present, I have completed the selection of the framework and completed the simple construction. Currently, I am completing the modular development of js and CSS. After the development is completed, I will enter the most important component development. We divide the page into several different components. , each independently visible on the page. The interactive area is treated as a component; it is easier to maintain and develop in this way.

13 Sep 2022 DevLog of Front-end

Front-end: At present, the modular development of JS and CSS has been completed. I am currently working on the last stage of component-based development, developing the pages of the client and the administrator in layers, which is convenient for maintenance, and the interface of the administrator is basically completed. Currently, I am developing the relevant data pages of the client. After the development is completed, the front-end and back-end debug and test the system as a whole. After the debugging and test, my front-end work is completed.

21 Sep 2022 DevLog of Front-end

Front-end: At present, the development of the relevant data pages on the user end has been completed. At present, the system is debugged and tested as a whole in the joint back-end. At the same time, there are some data interaction problems, such as the back-end error code is returned, the front-end will not be able to recognize , the front-end has not done small problems such as data format verification, and is currently being optimized. After optimization, the front-end work will be completed after the test is correct again.

04 Oct 2022 DevLog of Front-end

Front-end: Currently evaluating and optimizing the ui effect of user operation, beautifying the color, layout, etc. of the ui, and at the same time increasing the operation animation effect displayed on the front-end, enhancing the user experience, system fluency and user experience workload, combined with the sprint0 feedback to make changes

11 Oct 2022 DevLog of Front-end

Front-end completed functions: filter related articles, article-related functions, follow related functions, search functions, management-side article management all functions, user management all functions, login record management.

Back End Development

1. Requirements Requirements
2. Database design Database Development
3. Code structureCode structure
4. API documentationAPI Documentation
5. Development LogDevLog of Backend
6. Code repository https://github.com/liu35/it-bbq

DevLog of Backend

Used to record development process of backend, summarize errors, plan strategy and document architectures for helping collaborators in this project.

Current arrangement

1. Independently design database and implement functionalities in backend according to user story
2. Supporting frontend to complete designed architectures during the development
3. Testing and deploying

Known Errors

Known errors during development, may include bugs, misuse in programming method, architectural issues etc.

- Retrieving sole object from data models

```
Entry.objects.get(field ="value")
```

`.get()` will raise a `DoesNotExist` exception If there are no results that match the query.

```
list = Entry.objects.filter(field ="value")
if list: list[0]
```

Try use `.filter()` with a slice of [0].

(Django, no longer used)

- Confusions and issues in framework

  2022-09-21 Meeting note

- Improvements that need to be made

  Improvements that need to be made

Development

Content	Status	Issues	Document	Date

Register/Login	Aborted due to change in framework	1. Arguments need to be in agreement with frontend. () 2. More details needs to be completed in login part. ()	/	8/23/2022
Changing in framework	Done	1. Familiar with SpringBoot () 2. Test case ()	2022-08-29 Meeting notes (framework)	8/29/2022
Design in Database	Done	Database tables may vary later (Alterations will be documented)	Database Development	8/30/2022
Developing Interfaces sprint 0	Done	Detail need to negotiate with front end () More API will be added(see sprint 1)	API Documentation for Sprint 0	9/5/2022 to 9/25/2022
Testing	Done	More testing on sprint 1 need to be made(…)	Integration Testing	9/18/2022
Deployment	Done	\	Deployment instruction	9/27/2022
Developing Interfaces sprint 1	Done	A few bugs need to be fixed()	API Documentation	9/29/2022 to 10/13/2022

Database Development

Database : BBS

	Entity	Attributes	Attribute Type
1	`bbq_user` # user info	1. `id` (primary key) 2. `email` (unique key) 3. `username` 4. `password` 5. `role` 6. `state` 7. `gender` 8. `avatar` 9. `signature` 10. `is_delete` 11. `create_time` 12. `update_time`	1. bigint(11) / not null / auto_increment 2. varchar(64) / not null 3. varchar(64) / not null 4. varchar(128) / not null 5. varchar(32) / not null 6. varchar(64) / not null 7. varchar(32) / not null 8. varchar(256) / not null / default '' 9. varchar(1024) / not null 10. tinyint(2) unsigned / not null / default '0' **(1: deleted, 0: not)** 11. datetime / not null 12. timestamp / not null / default current_timestamp on update current_timestamp
2	`bbq_article_type` # article type	1. `id` (primary key) 2. `name` (unique key) 3. `description` 4. `ref_count` 5. `scope` 6. `is_delete` 7. `create_time` 8. `update_time`	1. bigint(11) / not null / auto_increment 2. varchar(64) / not null 3. varchar(1024) / not null 4. bigint(11) / not null / default '0' 5. varchar(32) / not null 6. tinyint(2) unsigned / not null / default '0' **(1: deleted, 0: not)** 7. datetime / not null 8. timestamp / not null / default current_timestamp on update current_timestamp
3	`bbq_comments` # comment	1. `id` (primary key) 2. `user_id` 3. `reply_id` 4. `reply_reply_id` 5. `posts_id` 6. `content` 7. `is_delete` 8. `create_time` 9. `update_time`	1. bigint(11) / not null / auto_increment 2. bigint(11) / not null 3. bigint(11) / default null 4. bigint(11) / default null **(secondary comment id)** 5. bigint(11) / not null 6. longtext / not null 7. tinyint(2) unsigned / not null / default '0' **(1: deleted, 0: not)** 8. datetime / not null 9. timestamp / not null / default current_timestamp on update current_timestamp
4	`bbq_message` #send message	1. `id` (primary key) 2. `channel` 3. `type` 4. `read` 5. `sender` 6. `receiver` 7. `title` 8. `content` 9. `is_delete`	1. bigint(11) / not null / auto_increment 2. varchar(64) / not null 3. varchar(64) / not null 4. varchar(64) / not null 5. bigint(11) / not null 6. bigint(11) / not null 7. varchar(256) / not null 8. longtext / not null 9. tinyint(2) unsigned / not null / default '0' **(1: deleted, 0: not)**

		10. create_time 11. update_time	10. datetime / not null 11. timestamp / not null / default current_timestamp on update current_timestamp
5	bbq_operate_log # operate log	1. id (primary key) 2. type 3. operater_id 4. content 5. is_delete 6. create_time 7. update_time	1. bigint(11) / not null / auto_increment 2. varchar(64) / not null 3. bigint(11) / not null 4. longtext / not null 5. tinyint(2) unsigned / not null / default '0' **(1: deleted, 0: not)** 6. datetime / not null 7. timestamp / not null / default current_timestamp on update current_timestamp
6	bbq_posts # posts	1. id (primary key) 2. audit_state 3. category 4. author_id 5. title 6. content_type 7. markdown_content 8. html_content 9. views 10. approvals 11. comments 12. type_id 13. head_img 14. official 15. top 16. sort 17. marrow 18. is_delete 19. create_time 20. update_time	1. bigint(11) / not null / auto_increment 2. varchar(64) / not null 3. varchar(64) / not null 4. bigint(11) / not null 5. varchar(256) / not null 6. varchar(64) / not null 7. longtext / not null 8. longtext / not null 9. bigint(11) / not null / default '0' 10. bigint(11) / not null / default '0' 11. bigint(11) / not null / default '0' 12. bigint(11) / not null / default '0' 13. varchar(8192) / not null / default '' 14. tinyint(2) unsigned / not null / default '0' 15. tinyint(2) unsigned / not null / default '0' 16. int(4) / not null / default '1000' 17. tinyint(2) unsigned / not null / default '0' 18. tinyint(2) unsigned / not null / default '0' 19. datetime / not null 20. timestamp / not null / default current_timestamp on update current_timestamp 21. (audit_state, category, views)using btree
7	bbq_tag # labels	1. id (primary key) 2. name 3. group_name 4. description 5. ref_count 6. is_delete 7. create_time 8. update_time	1. bigint(11) / not null / auto_increment 2. varchar(64) / not null 3. varchar(255) / not null 4. varchar(1024) / not null 5. bigint(11) / not null / default '0' 6. tinyint(2) unsigned / not null / default '0' 7. datetime / not null 8. timestamp / not null / default current_timestamp on update current_timestamp
8	bbq_follow # user follow	1. id (primary key) 2. followed 3. follower 4. is_delete 5. create_time 6. update_time	1. bigint(11) / not null / auto_increment 2. bigint(11) / not null 3. bigint(11) / not null 4. tinyint(2) unsigned / not null / default '0' 5. datetime / not null 6. timestamp / not null / default current_timestamp on update current_timestamp 7. (followed,followed_type,follower)using btree

1st version - Aborted

	Entity	Attributes	Attribute Type
1	channel	1. id (primary key) 2. title 3. info	1. int / not null / auto_increment 2. varchar(20) / not null 3. varchar(60)
2	zone	1. id (primary key) 2. title 3. info 4. channel	1. int / not null / auto_increment 2. varchar(20) / not null 3. varchar(60) 4. foreign key references channel(id) / not null
3	user	1. id (primary key) 2. username 3. password 4. gender 5. photo_url 6. email 7. type 8. register_date 9. level	1. int / not null / auto_increment 2. varchar(20) / not null 3. varchar(20) / not null 4. varchar(10) 5. tinytext 6. varchar(30) / not null 7. int / not null 8. datetime / not null 9. int / not null / default 0

		10. `state`	10. int
4	`manager`	1. `id` (primary key) 2. `username` 3. `password` 4. `gener` 5. `photo_url` 6. `email` 7. `type`	1. int / not null / auto_increment 2. varchar(20) / not null 3. varchar(20) / not null 4. varchar(10) 5. tinytext 6. varchar(30) / not null 7. int / not null
5	`post`	1. `id` (primary key) 2. `title` 3. `content` 4. `send_date` 5. `post_type` 6. `reply_num` 7. `view_num` 8. `like_num` 9. `zone` 10. `user_id`	1. int / not null / auto_increment 2. varchar(40) / not null 3. varchar(1024) / not null 4. datetime / not null 5. int 6. int 7. int 8. int 9. foreign key references `zone(id)` / not null 10. foreign key references `user(id)` / not null
6	`comments`	1. `id` 2. `follow_content` 3. `follow_date` 4. `like_num` 5. `post` 6. `user_id`	1. int / not null / auto_increment 2. varchar(1024) / not null 3. datetime / not null 4. int 5. foreign key references `post(id)` / not null 6. foreign key references `user(id)` / not null
7	`notice`	1. `id` 2. `title` 3. `content` 4. `notice_date` 5. `manager_id`	1. int / not null / auto_increment 2. varchar(40) / not null 3. varchar(1024) / not null 4. datetime / not null 5. foreign key references `manager(id)` / not null
8	`black_list`	1. `id` 2. `level` 3. `user_id`	1. int / not null / auto_increment 2. int / not null 3. foreign key references `user(id)` / not null

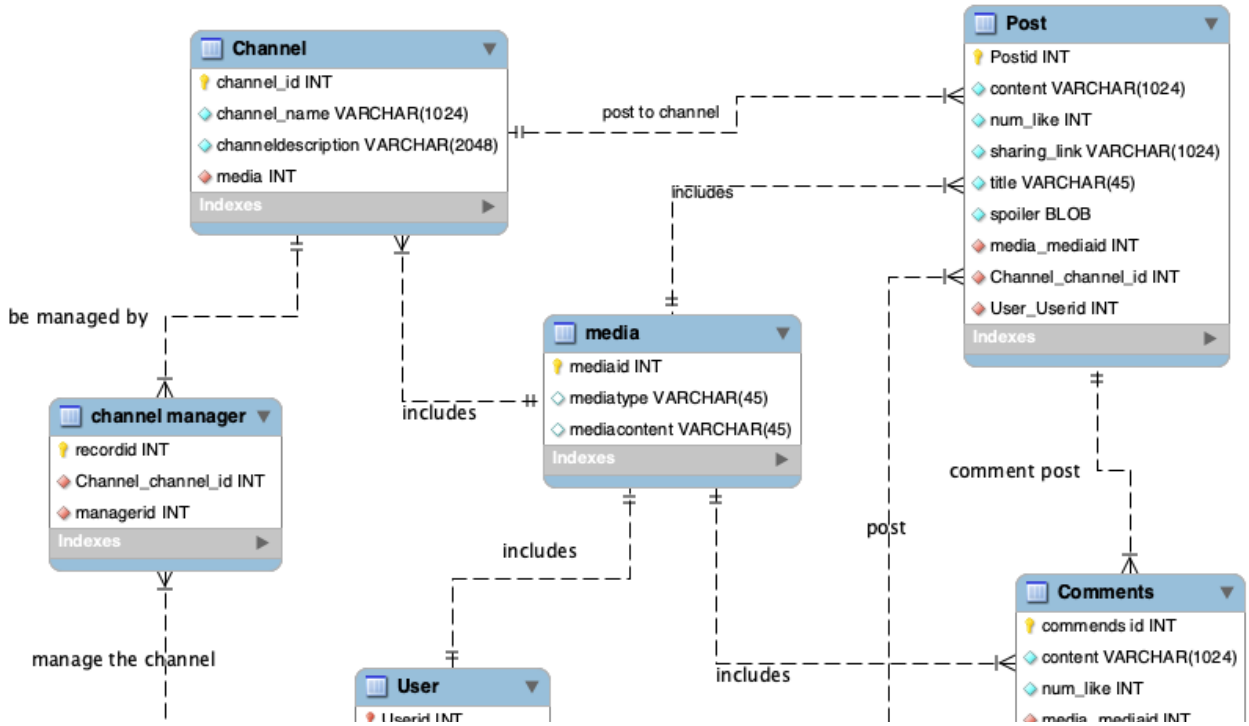
Alterations

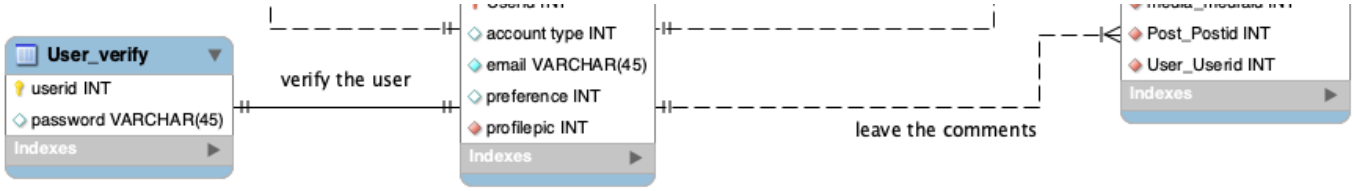
2022-09-03

2022-09-03 Meeting notes

Design of the database

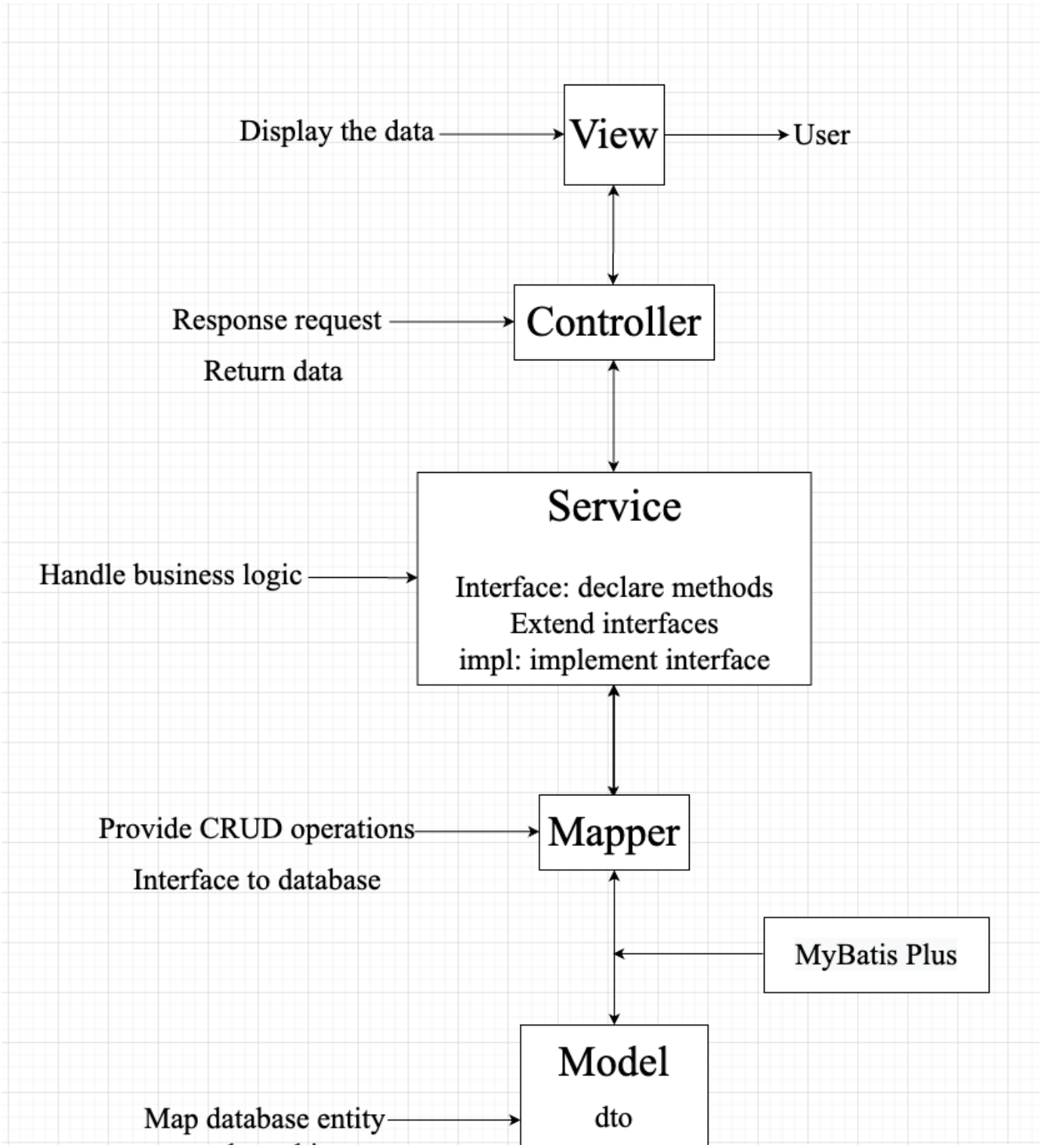
Below is our first version of the structure of our database.

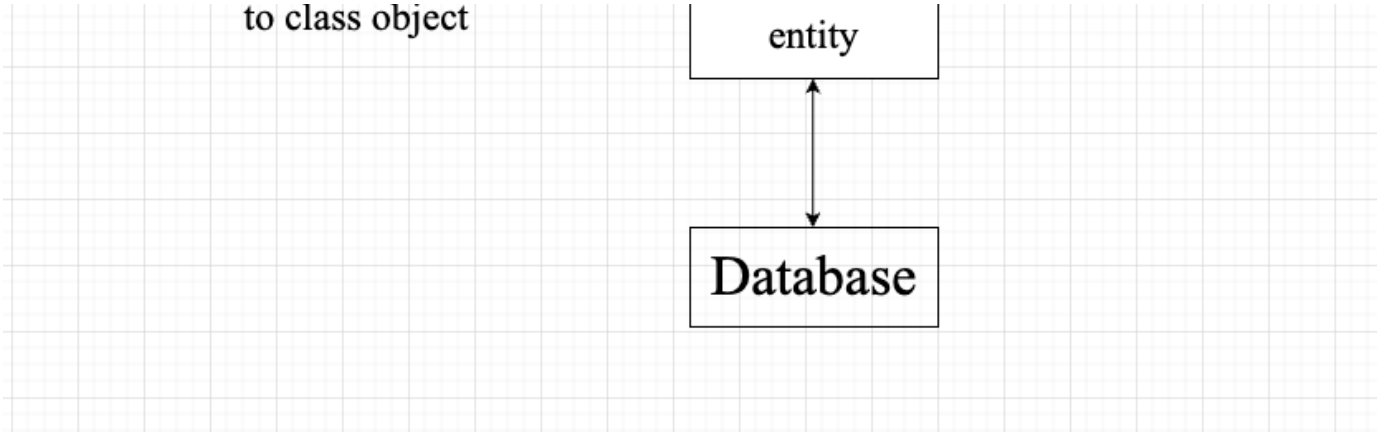




Code structure

The code structure of back end in this project.





API Documentation

API for Sprint 0 **https://team72itproject.atlassian.net/wiki/spaces/KB/pages/8192028/API+Documentation**

```
APIs recorded blow is all interfaces developed for sprint one and
future development.
```


1.1.1 Delete top

URLlocalhost:8888/posts/admin/top/delete?postId=6

Content-Typeapplication/json

Methodget

URI parameters

Name	Example	Type	Required
postId	6	String	Yes

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data	null	Null	data

Response instance: (200) Successful operation

```
{"code":200,"msg":"Operation is successful","data":null}
```

Response: (400)Fail

Name	Example	Type	Description
code	400	Integer	code
msg	the posts is not exist	String	message
data	null	Null	data

Response instance: (400)Fail

```
{"code":400,"msg":"the posts is not exist","data":null}
```

1.1.2 ArticleType

URLlocalhost:8888/article/types

Content-Typeapplication/json

Methodget

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data		Object	data
data.id	1	Integer	type id
data.name	MOVIE	String	type name
data.description	MOVIW	String	type description
data.refCount	3	Integer	type number of article
data.scope	USER	String	type scope
data.isDelete	0	Integer	type is detele
data.createTime	2022-09-27T21:00:57	String	type create time
data.updateTime	2022-09-27T21:00:57	String	type update time

Response instance: (200) Successful operation


```
{"code":200,"msg":"Operation is successful","data":[{"id":1,"name":"MOVIE","description":"MOVIW","refCount":3,"scope":"
USER","isDelete":0,"createTime":"2022-09-27T21:00:57","updateTime":"2022-09-27T21:00:57"},{"id":2,"name":"TV","description":"
TV","refCount":0,"scope":"USER","isDelete":0,"createTime":"2022-09-27T21:01:08","updateTime":"2022-09-27T21:01:08"},{"id":3,"
```

name":"ANIME","description":"ANIME","refCount":0,"scope":"USER","isDelete":0,"createTime":"2022-09-27T21:01:17","updateTime":"2022-09-27T21:01:17"},{"id":4,"name":"GAME","description":"GAME","refCount":0,"scope":"USER","isDelete":0,"createTime":"2022-09-27T21:01:28","updateTime":"2022-10-10T23:22:33"},{"id":5,"name":"OTHER","description":"OTHER","refCount":6,"scope":"USER","isDelete":0,"createTime":"2022-09-27T15:01:36","updateTime":"2022-09-27T15:01:38"},{"id":6,"name":"ANIMALS","description":"ANIMALS","refCount":0,"scope":"USER","isDelete":0,"createTime":"2022-10-30T02:13:07","updateTime":"2022-10-30T02:13:07"}]}

1.1.3 AddType

URLlocalhost:8888/article/admin/type/add?typeName=NewType

Content-Typeapplication/json

Methodget

URI parameters

Name	Example	Type	Required	Description
typeName	NewType	String	Yes	add type name

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data	add success	String	data

Response instance: (200) Successful operation

{"code":200,"msg":"Operation is successful","data":"add success"}

1.1.4 Delete type

URLlocalhost:8888/article/admin/type/delete?id=8

Content-Typeapplication/json

Methodget

URI parameters

Name	Example	Type	Required	Description
id	8	String	Yes	delete type id

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data	delete success	String	data

Response instance: (200) Successful operation

{"code":200,"msg":"Operation is successful","data":"delete success"}

Response: (400)Fail

Name	Example	Type	Description

code	400	Integer	code
msg	delete fail	String	message
data	null	Null	data

Response instance: (400)Fail

```
{"code":400,"msg":"delete fail","data":null}
```

1.1.5 Add follow

URLlocalhost:8888/follow/add?authorId=2&followerId=1

Content-Typeapplication/json

Methodget

URI parameters

Name	Example	Type	Required	Description
authorId	2	String	Yes	followed user id
followerId	1	String	Yes	follower user id

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data	null	Null	data

Response instance: (200) Successful operation

```
{"code":200,"msg":"Operation is successful","data":null}
```

Response: (400)Fail

Name	Example	Type	Description
code	400	Integer	code
msg	You're already following the user	String	message
data	null	Null	data

Response instance: (400)Fail

```
{"code":400,"msg":"You're already following the user","data":null}
```

1.1.6 Delete follow

URLlocalhost:8888/follow/delete?authorId=1&followerId=2

Content-Typeapplication/json

Methoddelete

URI parameters

Name	Example	Type	Required	Description

authorId	1	String	Yes	followed user id
followerId	2	String	Yes	follower user id

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data	null	Null	data

Response instance: (200) Successful operation


```
{"code":200,"msg":"Operation is successful","data":null}
```


Response: (400)Fail

Name	Example	Type	Description
code	400	Integer	code
msg	You're not following the user anymore	String	message
data	null	Null	data

Response instance: (400)Fail


```
{"code":400,"msg":"You're not following the user anymore","data":null}
```


1.1.7 List post

URLlocalhost:8888/posts/index/list?pageNum=1&pageSize=5&search=test&typeId=

Content-Typeapplication/json

Methodget

URI parameters

Name	Example	Type	Required	Description
pageNum	1	Integer	No	current page
pageSize	5	Integer	No	page size
search	test	String	No	search content
typeId		String	No	post category

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data		Object	data
data.records		Object	
data.records.id	7	Integer	deletype
data.records.auditState	PASS	String	
data.records.category	ARTICLE	String	
data.records.authorId	6	Integer	followed user id

data.records.title	pic test	String	
data.records.contentType	MARKDOWN	String	
data.records.markdownContent	![go](https://image.shutterstock.com /image-vector/go-vector-lettering-isolated-on-260nw-1717017016.jpg)	String	
data.records.htmlContent	test	String	
data.records.views	79	Integer	
data.records.approvals	9	Integer	
data.records.comments	2	Integer	
data.records.typeId	5	Integer	post category
data.records.headImg		String	
data.records.official	0	Integer	
data.records.top	0	Integer	
data.records.sort	1000	Integer	
data.records.marrow	1	Integer	
data.records.isDelete	0	Integer	
data.records.createTime	2022-10-10T11:15:57	String	
data.records.updateTime	2022-10-10T11:15:58	String	
data.records.authorName	gin	String	
data.records.type	OTHER	String	
data.total	3	Integer	
data.size	5	Integer	
data.current	1	Integer	
data.orders		Object	
data.searchCount	true	Boolean	
data.pages	1	Integer	

Response instance: (200) Successful operation

{"code":200,"msg":"Operation is successful","data":{"records":[{"id":7,"auditState":"PASS","category":"ARTICLE","authorId":6,"title":"pic test","contentType":"MARKDOWN","markdownContent":"![go](https://image.shutterstock.com/image-vector/go-vector-lettering-isolated-on-260nw-1717017016.jpg)","htmlContent":"test","views":79,"approvals":9,"comments":2,"typeId":5,"headImg":"","official":0,"top":0,"sort":1000,"marrow":1,"isDelete":0,"createTime":"2022-10-10T11:15:57","updateTime":"2022-10-10T11:15:58","authorName":"gin","type":"OTHER"},{"id":6,"auditState":"PASS","category":"ARTICLE","authorId":6,"title":"Gin is super handsome","contentType":"MARKDOWN","markdownContent":"HI**, this is a *++testing++* bala post\ns\n![16137992359659254.jpg](1)","htmlContent":"good-looking","views":28,"approvals":4,"comments":0,"typeId":5,"headImg":"","official":0,"top":0,"sort":1000,"marrow":0,"isDelete":0,"createTime":"2022-10-10T20:13:40","updateTime":"2022-10-10T20:13:40","authorName":"gin","type":"OTHER"},{"id":5,"auditState":"PASS","category":"ARTICLE","authorId":6,"title":"Gin is super handsome","contentType":"MARKDOWN","markdownContent":"**HI**, this is a *++testing++* bala post\ns\n![16137992359659254.jpg](1)","htmlContent":"good-looking","views":8,"approvals":7,"comments":0,"typeId":5,"headImg":"","official":0,"top":0,"sort":1000,"marrow":0,"isDelete":0,"createTime":"2022-10-10T17:13:34","updateTime":"2022-10-10T17:13:35","authorName":"gin","type":"OTHER"}],"total":3,"size":5,"current":1,"orders":[],"searchCount":true,"pages":1}}**

1.1.8 Post detail

URLlocalhost:8888/posts/index/:id

Content-Typeapplication/json

Methodget

URL parameter:

Name	Example	Description
id	6	post id

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data		Object	data
data.id	6	Integer	type id
data.auditState	PASS	String	
data.category	ARTICLE	String	
data.authorId	6	Integer	followed user id
data.title	Gin is super handsome	String	
data.contentType	MARKDOWN	String	
data.markdownContent	**HI**, this is a *++testing++* bala post s ![16137992359659254.jpg](1)	String	
data.htmlContent	good-looking	String	
data.views	28	Integer	
data.approvals	4	Integer	
data.comments	0	Integer	
data.typeId	5	Integer	post category
data.headImg		String	
data.official	0	Integer	
data.top	0	Integer	
data.sort	1000	Integer	
data.marrow	0	Integer	
data.isDelete	0	Integer	type is detele
data.createTime	2022-10-10T20:13:40	String	type create time
data.updateTime	2022-10-10T20:13:40	String	type update time
data.authorName	gin	String	
data.type	OTHER	String	

Response instance: (200) Successful operation

{"code":200,"msg":"Operation is successful","data":{"id":6,"auditState":"PASS","category":"ARTICLE","authorId":6,"title":"Gin is super handsome","contentType":"MARKDOWN","markdownContent":"HI**, this is a *++testing++* bala post\ns\n! [16137992359659254.jpg](1)","htmlContent":"good-looking","views":28,"approvals":4,"comments":0,"typeId":5,"headImg":"","official":0,"top":0,"sort":1000,"marrow":0,"isDelete":0,"createTime":"2022-10-10T20:13:40","updateTime":"2022-10-10T20:13:40","authorName":"gin","type":"OTHER"}}**

1.1.9 Post edit

URLlocalhost:8888/posts/index/edit

Content-Typeapplication/json

Methodpost

Request parameters:

Name	Example	Type	Required	Description

id	6	Integer	Yes	deletype
auditState	PASS	String	Yes	
category	ARTICLE	String	Yes	
authorId	6	Integer	Yes	followed user id
title	Gin is super handsome	String	Yes	
contentType	MARKDOWN	String	Yes	
markdownContent	**HI**, this is a *++testing++* bala post s ![16137992359659254.jpg](1)	String	Yes	
htmlContent	good-looking	String	Yes	
views	28	Integer	Yes	
approvals	4	Integer	Yes	
comments	0	Integer	Yes	
typeId	5	Integer	Yes	post category
headImg		String	Yes	
official	0	Integer	Yes	
top	0	Integer	Yes	
sort	1000	Integer	Yes	
marrow	0	Integer	Yes	
isDelete	0	Integer	Yes	
createTime	2022-10-10T20:13:40	String	Yes	
updateTime	2022-10-10T20:13:40	String	Yes	
authorName	gin	String	Yes	
type	OTHER	String	Yes	

Request instance:


```
{
    "id": 6,
"auditState": "PASS",
"category": "ARTICLE",
"authorId": 6,
"title": "Gin is super handsome",
"contentType": "MARKDOWN",
"markdownContent": "**HI**, this is a *++testing++* bala post\ns\n![16137992359659254.jpg](1)",
"htmlContent": "good-looking",
"views": 28,
"approvals": 4,
"comments": 0,
"typeId": 5,
"headImg": "",
"official": 0,
"top": 0,
"sort": 1000,
```

"marrow": 0,

"isDelete": 0,

"createTime": "2022-10-10T20:13:40",

"updateTime": "2022-10-10T20:13:40",

"authorName": "gin",

"type": "OTHER"

}

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data		Object	data
data.id	6	Integer	type id
data.auditState	PASS	String	
data.category	ARTICLE	String	
data.authorId	6	Integer	followed user id
data.title	Gin is super handsome	String	
data.contentType	MARKDOWN	String	
data.markdownContent	**HI**, this is a *++testing++* bala post s ![16137992359659254.jpg](1)	String	
data.htmlContent	good-looking	String	
data.views	28	Integer	
data.approvals	4	Integer	
data.comments	0	Integer	
data.typeId	5	Integer	post category
data.headImg		String	
data.official	0	Integer	
data.top	0	Integer	
data.sort	1000	Integer	
data.marrow	0	Integer	
data.isDelete	0	Integer	type is detele
data.createTime	2022-10-10T20:13:40	String	type create time
data.updateTime	2022-10-10T20:13:40	String	type update time
data.authorName	gin	String	
data.type	OTHER	String	

Response instance: (200) Successful operation

{"code":200,"msg":"Operation is successful","data":{"id":6,"auditState":"PASS","category":"ARTICLE","authorId":6,"title":"Gin is super handsome","contentType":"MARKDOWN","markdownContent":"HI**, this is a *++testing++* bala post\ns\n! [16137992359659254.jpg](1)","htmlContent":"good-looking","views":28,"approvals":4,"comments":0,"typeId":5,"headImg":"","official":0,"top":0,"sort":1000,"marrow":0,"isDelete":0,"createTime":"2022-10-10T20:13:40","updateTime":"2022-10-10T20:13:40","authorName":"gin","type":"OTHER"}}**

1.1.10 Approval post

URLlocalhost:8888/posts/index/approval?postId=99&userId=1&click=0

Content-Typeapplication/json

Methodget

URI parameters

Name	Example	Type	Required	Description
postId	99	Integer	Yes	post id
userId	1	Integer	Yes	user id
click	0	Integer	Yes	is first click

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data	approval	String	data

Response instance: (200) Successful operation

```
{"code":200,"msg":"Operation is successful","data":"approval"}
```

Response: (400)Fail

Name	Example	Type	Description
code	400	Integer	code
msg	you have already approved	String	message
data	null	Null	data

Response instance: (400)Fail

```
{"code":400,"msg":"you have already approved","data":null}
```

Response: (400)post not exist

Name	Example	Type	Description
code	400	Integer	code
msg	the posts is not exist	String	message
data	null	Null	data

Response instance: (400)post not exist

```
{"code":400,"msg":"the posts is not exist","data":null}
```

1.1.11 Delete approval

URLlocalhost:8888/posts/index/delete/approval?postId=6&userId=1

Content-Typeapplication/json

Methodget

URI parameters

Name	Example	Type	Required
postId	6	String	Yes
userId	1	String	Yes

Response instance: (200) Successful operation

{"code":200,"msg":"Operation is successful","data":null}

1.1.12 Set top

URLlocalhost:8888/posts/admin/top/set?postId=99

Content-Typeapplication/json

Methodget

URI parameters

Name	Example	Type	Required	Description
postId	99	String	Yes	post id

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data	null	Null	data

Response instance: (200) Successful operation

{"code":200,"msg":"Operation is successful","data":null}

Response: (400)Fail

Name	Example	Type	Description
code	400	Integer	code
msg	the posts is not exist	String	message
data	null	Null	data

Response instance: (400)Fail

{"code":400,"msg":"the posts is not exist","data":null}

1.1.13 Set marrow

URLlocalhost:8888/posts/admin/marrow/set?postId=99

Content-Typeapplication/json

Methodget

URI parameters

Name	Example	Type	Required
postId	99	String	Yes

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data	null	Null	data

Response instance: (200) Successful operation

{"code":200,"msg":"Operation is successful","data":null}

Response: (400)Fail

Name	Example	Type	Description
code	400	Integer	code
msg	the posts is not exist	String	message
data	null	Null	data

Response instance: (400)Fail

{"code":400,"msg":"the posts is not exist","data":null}

1.1.14 Delete marrow

URLlocalhost:8888/posts/admin/marrow/delete?postId=6

Content-Typeapplication/json

Methodget

URI parameters

Name	Example	Type	Required
postId	6	String	Yes

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data	null	Null	data

Response instance: (200) Successful operation

{"code":200,"msg":"Operation is successful","data":null}

Response: (400)Fail

Name	Example	Type	Description
code	400	Integer	code
msg	the posts is not exist	String	message
data	null	Null	data

Response instance: (400)Fail

```
{"code":400,"msg":"the posts is not exist","data":null}
```

1.1.15 Set official

URLlocalhost:8888/posts/admin/official/set?postId=6

Content-Typeapplication/json

Methodget

URI parameters

Name	Example	Type	Required
postId	6	String	Yes

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data	null	Null	data

Response instance: (200) Successful operation

```
{"code":200,"msg":"Operation is successful","data":null}
```

Response: (400)Fail

Name	Example	Type	Description
code	400	Integer	code
msg	the posts is not exist	String	message
data	null	Null	data

Response instance: (400)Fail

```
{"code":400,"msg":"the posts is not exist","data":null}
```

1.1.16 Delete official

URLlocalhost:8888/posts/admin/official/delete?postId=99

Content-Typeapplication/json

Methodget

URI parameters:

Name	Example	Type	Required
postId	99	String	Yes

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code

msg	Operation is successful	String	message
data	null	Null	data

Response instance: (200) Successful operation

```
{"code":200,"msg":"Operation is successful","data":null}
```

Response: (400)Fail

Name	Example	Type	Description
code	400	Integer	code
msg	the posts is not exist	String	message
data	null	Null	data

Response instance: (400)Fail

```
{"code":400,"msg":"the posts is not exist","data":null}
```

1.1.17 Set unseen

URLlocalhost:8888/posts/admin/posts/unseen?postId=99

Content-Typeapplication/json

Methodget

URI parameters:

Name	Example	Type	Required
postId	99	String	Yes

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data	null	Null	data

Response instance: (200) Successful operation

```
{"code":200,"msg":"Operation is successful","data":null}
```

Response: (400)Fail

Name	Example	Type	Description
code	400	Integer	code
msg	the posts is not exist	String	message
data	null	Null	data

Response instance: (400)Fail

```
{"code":400,"msg":"the posts is not exist","data":null}
```

1.1.18 Set seen

URLlocalhost:8888/posts/admin/posts/seen?postId=6

Content-Typeapplication/json

Methodget

URI parameters:

Name	Example	Type	Required
postId	6	String	Yes

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data	null	Null	data

Response instance: (200) Successful operation

```
{"code":200,"msg":"Operation is successful","data":null}
```

Response: (400)Fail

Name	Example	Type	Description
code	400	Integer	code
msg	the posts is not exist	String	message
data	null	Null	data

Response instance: (400)Fail

```
{"code":400,"msg":"the posts is not exist","data":null}
```

1.1.19 Comment list

URLlocalhost:8888/posts/comment/list?postId=6

Content-Typeapplication/json

Methodget

URI parameters:

Name	Example	Type	Required	Description
postId	6	String	Yes	post id

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data		Object	data
data.id	3	Integer	type id
data.userId	4	Integer	user id

data.username	admin	String	
data.avatar	http://localhost:8888/upload/9860d86b-4508-4487-bfb0-97b3f965c0d8.jpg	String	
data.replyNum	0	Integer	
data.content	1231231	String	
data.updateTime	2022-10-18T05:27:27	String	type update time
data.commentReplyList		Object	

Response instance: (200) Successful operation


```
{
"code": 200,
"msg": "Operation is successful",
"data": [
{
"id": 3,
"userId": 4,
"username": "admin",
"avatar": "http://localhost:8888/upload/9860d86b-4508-4487-bfb0-97b3f965c0d8.jpg",
"replyNum": 0,
"content": "1231231",
"updateTime": "2022-10-18T05:27:27",
"commentReplyList": []
},
{
"id": 2,
"userId": 4,
"username": "admin",
"avatar": "http://localhost:8888/upload/9860d86b-4508-4487-bfb0-97b3f965c0d8.jpg",
"replyNum": 2,
"content": "1111",
"updateTime": "2022-10-18T05:19:45",
"commentReplyList": [
{
"id": 2,
"userId": 4,
"username": "admin",
"avatar": "http://localhost:8888/upload/9860d86b-4508-4487-bfb0-97b3f965c0d8.jpg",
"commentId": 2,
"replyId": 4,
"replyName": "admin",
"replyNumber": 0,
```

```
"content": "123123123",
"createTime": "2022-10-17T21:20:34",
"updateTime": "2022-10-17T21:20:34"
},
{
"id": 1,
"userId": 4,
"username": "admin",
"avatar": "http://localhost:8888/upload/9860d86b-4508-4487-bfb0-97b3f965c0d8.jpg",
"commentId": 2,
"replyId": 4,
"replyName": "admin",
"replyNumber": 0,
"content": "@admin 1231231",
"createTime": "2022-10-17T21:19:53",
"updateTime": "2022-10-17T21:19:53"
}
]
},
{
"id": 1,
"userId": 4,
"username": "admin",
"avatar": "http://localhost:8888/upload/9860d86b-4508-4487-bfb0-97b3f965c0d8.jpg",
"replyNum": 0,
"content": "123123",
"updateTime": "2022-10-18T04:56:42",
"commentReplyList": []
}
]
}
```


1.1.20 Comment add

URLlocalhost:8888/posts/comment/add

Content-Typeapplication/json

Methodpost

Request instance:


```
{
"postId": 6,
"userId": 4,
```

"username": "admin",

"avatar": "http://localhost:8888/upload/9860d86b-4508-4487-bfb0-97b3f965c0d8.jpg",

"replyNum": 0,

"content": "1231231",

}

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data	null	Null	data

Response instance: (200) Successful operation

{"code":200,"msg":"Operation is successful","data":null}

1.1.21 Apply add

URLlocalhost:8888/posts/reply/add

Content-Typeapplication/json

Methodpost

Request instance:

{

"commentId": 1,

"userId": 4,

"username": "admin",

"avatar": "http://localhost:8888/upload/9860d86b-4508-4487-bfb0-97b3f965c0d8.jpg",

"content": "1231231",

"replyId": 4,

"replyName": "admin",

}

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data	null	Null	data

Response instance: (200) Successful operation

{"code":200,"msg":"Operation is successful","data":null}

1.1.22 Set role

URLlocalhost:8888/user/admin/role/set?userId=4&role=ADMIN

Content-Typeapplication/json

Methodget

URI parameters:

Name	Example	Type	Required	Description
userId	4	String	Yes	user id
role	ADMIN	String	Yes	

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data	null	Null	data

Response instance: (200) Successful operation

```
{"code":200,"msg":"Operation is successful","data":null}
```

1.1.23 Set disable

URLlocalhost:8888/user/admin/disable?userId=4

Content-Typeapplication/json

Methodget

URI parameters:

Name	Example	Type	Required	Description
userId	4	String	Yes	user id

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data	null	Null	data

Response instance: (200) Successful operation

```
{"code":200,"msg":"Operation is successful","data":null}
```

1.1.24 Set enable

URLlocalhost:8888/user/admin/enable?userId=4

Content-Typeapplication/json

Methodget

URI parameters:

Name	Example	Type	Required	Description
userId	4	String	Yes	user id

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data	null	Null	data

Response instance: (200) Successful operation


```
{"code":200,"msg":"Operation is successful","data":null}
```


1.1.25 LoginLog list

URLlocalhost:8888/user/admin/loginLog?pageNum=1&pageSize=5

Content-Typeapplication/json

Methodget

URI parameters

Name	Example	Type	Required	Description
pageNum	1	Integer	No	current page
pageSize	5	Integer	No	page size

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data		Object	data
data.records		Object	
data.records.id	12	Integer	deletype
data.records.userId	4	Integer	user id
data.records.userName	admin	String	
data.records.userRole	USER	String	
data.records.loginTime	2022-10-10T22:44:58	String	
data.total	45	Integer	
data.size	5	Integer	
data.current	1	Integer	
data.orders		Object	
data.searchCount	true	Boolean	
data.pages	9	Integer	

Response instance: (200) Successful operation


```
{"code":200,"msg":"Operation is successful","data":{"records":[{"id":12,"userId":4,"userName":"admin","userRole":"USER","loginTime":"2022-10-10T22:44:58"},{"id":13,"userId":4,"userName":"admin","userRole":"ADMIN","loginTime":"2022-10-10T22:45:39"},{"id":14,"userId":4,"userName":"admin","userRole":"ADMIN","loginTime":"2022-10-10T22:47:56"},{"id":15,"userId":5,"userName":"123","userRole":"USER","loginTime":"2022-10-10T22:48:25"},{"id":16,"userId":4,"userName":"admin","userRole":"ADMIN","loginTime":"2022-10-10T22:49:23"}],"total":45,"size":5,"current":1,"orders":[],"searchCount":true,"pages":9}}
```

1.1.26 User list

URLlocalhost:8888/user/admin/all?pageNum=1&pageSize=5

Content-Typeapplication/json

Methodget

URI parameters:

Name	Example	Type	Required	Description
pageNum	1	Integer	No	current page
pageSize	5	Integer	No	page size

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data		Object	data
data.records		Object	
data.records.id	1	Integer	deletype
data.records.email	yyz@163.com	String	
data.records.username	TestAistegg	String	
data.records.password	e10adc3949ba59abbe56e057f20f883e	String	
data.records.role	USER	String	
data.records.state	ENABLED	String	
data.records.sex	Male	String	
data.records.avatar	http://localhost:8888/upload /6c1e6a25-65dd-4cdb-9886-657787d9e711.jpg	String	
data.records.signature		String	
data.records.isDelete	0	Integer	
data.records.createTime	2022-09-26T21:56:47	String	
data.records.updateTime	2022-09-27T05:56:47	String	
data.total	8	Integer	
data.size	5	Integer	
data.current	1	Integer	
data.orders		Object	
data.searchCount	true	Boolean	
data.pages	2	Integer	

Response instance: (200) Successful operation

{"code":200,"msg":"Operation is successful","data":{"records":[{"id":1,"email":"yyz@163.com","username":"TestAistegg","password":"e10adc3949ba59abbe56e057f20f883e","role":"USER","state":"ENABLED","sex":"Male","avatar":"http://localhost:8888/upload/6c1e6a25-65dd-4cdb-9886-657787d9e711.jpg","signature":"","isDelete":0,"createTime":"2022-09-26T21:56:47","updateTime":"2022-09-27T05:56:47"},{"id":2,"email":"yyz@163.com","username":"xiaofu","password":"e10adc3949ba59abbe56e057f20f883e","role":"USER","state":"ENABLED","sex":"Male","avatar":"http://localhost:8888/upload/093c9242-f701-4e9b-9cff-2ab58f3a493f.jpg","signature":"","isDelete":0,"createTime":"2022-09-26T22:10:37","updateTime":"2022-09-27T06:10:37"},{"id":3,"email":"yyz@163.com","username":"test1","password":"e10adc3949ba59abbe56e057f20f883e","role":"USER","state":"ENABLED","sex":"No","avatar":"http://localhost:8888/upload/7091fdd5-fc4d-405d-8dba-55adf78d62bc.jpg","

signature":"","isDelete":0,"createTime":"2022-09-27T21:47:48","updateTime":"2022-09-28T05:47:48"},{"id":4,"email":"admin","username":"admin","password":"21232f297a57a5a743894a0e4a801fc3","role":"ADMIN","state":"ENABLED","sex":"No","avatar":"http://localhost:8888/upload/9860d86b-4508-4487-bfb0-97b3f965c0d8.jpg","signature":"test","isDelete":0,"createTime":"2022-10-10T22:44:51","updateTime":"2022-10-17T00:30:35"},{"id":5,"email":"123@123.com","username":"123","password":"202cb962ac59075b964b07152d234b70","role":"USER","state":"ENABLED","sex":"Female","avatar":"http://localhost:8888/upload/eea25789-4d6c-4ad1-ad60-b17b8c67260b.jpg","signature":"","isDelete":0,"createTime":"2022-10-10T22:48:15","updateTime":"2022-10-11T06:48:15"}],"total":8,"size":5,"current":1,"orders":[],"searchCount":true,"pages":2}}

1.1.27 Follower list

URLlocalhost:8888/user/follower/list?userId=4&pageNum=1&pageSize=5

Content-Typeapplication/json

Methodget

URI parameters:

Name	Example	Type	Required	Description
userId	4	Number	Yes	user id
pageNum	1	Integer	No	current page
pageSize	5	Integer	No	page size

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data		Object	data
data.records		Object	
data.records.id	7	Integer	deletype
data.records.followed	4	Integer	
data.records.follower	11	Integer	
data.records.isDelete	1	Integer	
data.records.createTime	2022-10-28T22:44:51	String	
data.records.updateTime	2022-10-29T00:19:37	String	
data.records.followerInfo		Object	
data.records.followerInfo.id	11	Integer	deletype
data.records.followerInfo.email	123	String	
data.records.followerInfo.username	123	String	
data.records.followerInfo.password	202cb962ac59075b964b07152d234b70	String	
data.records.followerInfo.role	USER	String	
data.records.followerInfo.state	ENABLED	String	
data.records.followerInfo.sex	Male	String	
data.records.followerInfo.avatar	http://localhost:8888/upload/3d511350-1887-4914-a88d-cbb495f80dcf.jpg	String	
data.records.followerInfo.signature		String	
data.records.followerInfo.isDelete	0	Integer	
data.records.followerInfo.createTime	2022-10-28T22:43:20	String	
data.records.followerInfo.updateTime	2022-10-28T22:43:20	String	

data.records.followedInfo	null	Null	
data.total	1	Integer	
data.size	5	Integer	
data.current	1	Integer	
data.orders		Object	
data.searchCount	true	Boolean	
data.pages	1	Integer	

Response instance: (200) Successful operation

{"code":200,"msg":"Operation is successful","data":{"records":[{"id":7,"followed":4,"follower":11,"isDelete":1,"createTime":"2022-10-28T22:44:51","updateTime":"2022-10-29T00:19:37","followerInfo":{"id":11,"email":"123","username":"123","password":"202cb962ac59075b964b07152d234b70","role":"USER","state":"ENABLED","sex":"Male","avatar":"http://localhost:8888/upload/3d511350-1887-4914-a88d-cbb495f80dcf.jpg","signature":"","isDelete":0,"createTime":"2022-10-28T22:43:20","updateTime":"2022-10-28T22:43:20"},"followedInfo":null}],"total":1,"size":5,"current":1,"orders":[],"searchCount":true,"pages":1}}

1.1.28 Comments list

URLlocalhost:8888/user/comments/list?userId=4&pageNum=1&pageSize=5

Content-Typeapplication/json

Methodget

URI parameters:

Name	Example	Type	Required	Description
userId	4	Number	Yes	user id
pageNum	1	Integer	No	current page
pageSize	5	Integer	No	page size

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data		Object	data
data.records		Object	
data.records.id	7	Integer	deletype
data.records.userId	4	Integer	user id
data.records.username	admin	String	
data.records.avatar	http://localhost:8888/upload/9860d86b-4508-4487-bfb0-97b3f965c0d8.jpg	String	
data.records.replyNum	0	Integer	
data.records.postId	6	Integer	post id
data.records.content	1231231	String	
data.records.isDelete	0	Integer	
data.records.createTime	2022-10-31T00:24:12	String	
data.records.updateTime	2022-10-31T00:24:12	String	
data.records.posts		Object	

data.records.posts.id	6	Integer	deletype
data.records.posts.auditState	PASS	String	
data.records.posts.category	ARTICLE	String	
data.records.posts.authorId	6	Integer	followed user id
data.records.posts.title	Gin is super handsome	String	
data.records.posts.contentType	MARKDOWN	String	
data.records.posts.markdownContent	**HI**, this is a *++testing++* bala post s ![16137992359659254.jpg](1)	String	
data.records.posts.htmlContent	good-looking	String	
data.records.posts.views	28	Integer	
data.records.posts.approvals	3	Integer	
data.records.posts.comments	1	Integer	
data.records.posts.typeId	5	Integer	post category
data.records.posts.headImg		String	
data.records.posts.official	0	Integer	
data.records.posts.top	0	Integer	
data.records.posts.sort	1000	Integer	
data.records.posts.marrow	0	Integer	
data.records.posts.isDelete	0	Integer	
data.records.posts.createTime	2022-10-10T20:13:40	String	
data.records.posts.updateTime	2022-10-10T20:13:40	String	
data.records.posts.authorName	null	Null	
data.records.posts.type	null	Null	
data.total	7	Integer	
data.size	5	Integer	
data.current	1	Integer	
data.orders		Object	
data.searchCount	true	Boolean	
data.pages	2	Integer	

Response instance: (200) Successful operation

{"code":200,"msg":"Operation is successful","data":{"records":[{"id":7,"userId":4,"username":"admin","avatar":"http://localhost:8888/upload/9860d86b-4508-4487-bfb0-97b3f965c0d8.jpg","replyNum":0,"postId":6,"content":"1231231","isDelete":0,"createTime":"2022-10-31T00:24:12","updateTime":"2022-10-31T00:24:12","posts":{"id":6,"auditState":"PASS","category":"ARTICLE","authorId":6,"title":"Gin is super handsome","contentType":"MARKDOWN","markdownContent":"HI**, this is a *++testing++* bala post\ns\n![16137992359659254.jpg](1)","htmlContent":"good-looking","views":28,"approvals":3,"comments":1,"typeId":5,"headImg":"","official":0,"top":0,"sort":1000,"marrow":0,"isDelete":0,"createTime":"2022-10-10T20:13:40","updateTime":"2022-10-10T20:13:40","authorName":null,"type":null}},{"id":6,"userId":4,"username":"admin","avatar":"http://localhost:8888/upload/9860d86b-4508-4487-bfb0-97b3f965c0d8.jpg","replyNum":0,"postId":7,"content":"123","isDelete":0,"createTime":"2022-10-28T19:09:35","updateTime":"2022-10-28T19:09:35","posts":{"id":7,"auditState":"PASS","category":"ARTICLE","authorId":6,"title":"pic test","contentType":"MARKDOWN","markdownContent":"![go](https://image.shutterstock.com/image-vector/go-vector-lettering-isolated-on-260nw-1717017016.jpg)","htmlContent":"test","views":79,"approvals":9,"comments":2,"typeId":5,"headImg":"","official":0,"top":0,"sort":1000,"marrow":1,"isDelete":0,"createTime":"2022-10-10T11:15:57","updateTime":"2022-10-10T11:15:58","authorName":null,"type":null}},{"id":5,"userId":4,"username":"admin","avatar":"http://localhost:8888/upload/9860d86b-4508-4487-bfb0-97b3f965c0d8.jpg","replyNum":0,"postId":7,"content":"1111","isDelete":0,"createTime":"2022-10-17T21:41:49","updateTime":"2022-10-18T05:41:49","posts":{"id":7,"auditState":"PASS","category":"ARTICLE","authorId":6,"title":"pic test","contentType":"MARKDOWN","markdownContent":"![go](https://image.shutterstock.com/image-vector/go-vector-lettering-isolated-on-260nw-1717017016.jpg)","htmlContent":"test","views":79,"approvals":9,"comments":2,"typeId":5,"headImg":"","official":0,"top":0,"sort":1000,"marrow":1,"isDelete":0,"createTime":"2022-10-10T11:15:57","updateTime":"2022-10-10T11:15:58","authorName":null,"type":null}},{"id":4,"userId":4,"username":"admin","avatar":"http://localhost:8888/upload/9860d86b-4508-4487-bfb0-97b3f965c0d8.jpg","replyNum":1,"postId":7,"content":"12313","isDelete":0,"createTime":"2022-10-17T21:30:10","updateTime":"2022-10-18T05:30:10","posts":{"id":7,"auditState":"PASS","category":"ARTICLE","authorId":6,"title":"pic test","contentType":"MARKDOWN","markdownContent":"![go](https://image.shutterstock.com/image-vector/go-vector-lettering-isolated-on-260nw-1717017016.jpg)","htmlContent":"test","views":**

79,"approvals":9,"comments":2,"typeId":5,"headImg":"","official":0,"top":0,"sort":1000,"marrow":1,"isDelete":0,"createTime":"2022-10-10T11:15:57","updateTime":"2022-10-10T11:15:58","authorName":null,"type":null}},{"id":3,"userId":4,"username":"admin","avatar":"http://localhost:8888/upload/9860d86b-4508-4487-bfb0-97b3f965c0d8.jpg","replyNum":0,"postId":6,"content":"1231231","isDelete":0,"createTime":"2022-10-17T21:27:27","updateTime":"2022-10-18T05:27:27","posts":{"id":6,"auditState":"PASS","category":"ARTICLE","authorId":6,"title":"Gin is super handsome","contentType":"MARKDOWN","markdownContent":"HI**, this is a *++testing++* bala post\ns\n![16137992359659254.jpg](1)","htmlContent":"good-looking","views":28,"approvals":3,"comments":1,"typeId":5,"headImg":"","official":0,"top":0,"sort":1000,"marrow":0,"isDelete":0,"createTime":"2022-10-10T20:13:40","updateTime":"2022-10-10T20:13:40","authorName":null,"type":null}}],"total":7,"size":5,"current":1,"orders":[],"searchCount":true,"pages":2}}**

1.1.29 Followed list

URLlocalhost:8888/user/followed/list?userId=4&pageNum=1&pageSize=5

Content-Typeapplication/json

Methodget

URI parameters:

Name	Example	Type	Required	Description
userId	4	Number	Yes	user id
pageNum	1	Integer	No	current page
pageSize	5	Integer	No	page size

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data		Object	data
data.records		Object	
data.records.id	6	Integer	deletype
data.records.followed	6	Integer	
data.records.follower	4	Integer	
data.records.isDelete	0	Integer	
data.records.createTime	2022-10-28T01:06:35	String	
data.records.updateTime	2022-10-28T01:06:35	String	
data.records.followerInfo	null	Null	
data.records.followedInfo		Object	
data.records.followedInfo.id	6	Integer	deletype
data.records.followedInfo.email	yyz@163.com	String	
data.records.followedInfo.username	gin	String	
data.records.followedInfo.password	4297f44b13955235245b2497399d7a93	String	
data.records.followedInfo.role	USER	String	
data.records.followedInfo.state	ENABLED	String	
data.records.followedInfo.sex	Male	String	
data.records.followedInfo.avatar		String	
data.records.followedInfo.signature		String	
data.records.followedInfo.isDelete	0	Integer	
data.records.followedInfo.createTime	2022-10-10T23:14:18	String	

data.records.followedInfo.updateTime	2022-10-11T07:14:18	String	
data.total	2	Integer	
data.size	5	Integer	
data.current	1	Integer	
data.orders		Object	
data.searchCount	true	Boolean	
data.pages	1	Integer	

Response instance: (200) Successful operation

{"code":200,"msg":"Operation is successful","data":{"records":[{"id":6,"followed":6,"follower":4,"isDelete":0,"createTime":"2022-10-28T01:06:35","updateTime":"2022-10-28T01:06:35","followerInfo":null,"followedInfo":{"id":6,"email":"yyz@163.com","username":"gin","password":"4297f44b13955235245b2497399d7a93","role":"USER","state":"ENABLED","sex":"Male","avatar":"","signature":"","isDelete":0,"createTime":"2022-10-10T23:14:18","updateTime":"2022-10-11T07:14:18"}},{"id":11,"followed":11,"follower":4,"isDelete":1,"createTime":"2022-10-30T00:22:37","updateTime":"2022-10-30T00:22:36","followerInfo":null,"followedInfo":{"id":11,"email":"123","username":"123","password":"202cb962ac59075b964b07152d234b70","role":"USER","state":"ENABLED","sex":"Male","avatar":"http://localhost:8888/upload/3d511350-1887-4914-a88d-cbb495f80dcf.jpg","signature":"","isDelete":0,"createTime":"2022-10-28T22:43:20","updateTime":"2022-10-28T22:43:20"}}],"total":2,"size":5,"current":1,"orders":[],"searchCount":true,"pages":1}}

1.1.30 Approval posts list

URLlocalhost:8888/user/approval/posts?userId=4&pageNum=1&pageSize=5

Content-Typeapplication/json

Methodget

URI parameters:

Name	Example	Type	Required	Description
userId	4	Number	Yes	user id
pageNum	1	Integer	No	current page
pageSize	5	Integer	No	page size

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	code
msg	Operation is successful	String	message
data		Object	data
data.records		Object	
data.records.id	14	Integer	deletype
data.records.postId	5	Integer	post id
data.records.userId	4	Integer	user id
data.records.posts		Object	
data.records.posts.id	5	Integer	deletype
data.records.posts.auditState	PASS	String	
data.records.posts.category	ARTICLE	String	
data.records.posts.authorId	6	Integer	followed user id
data.records.posts.title	Gin is super handsome	String	
data.records.posts.contentType	MARKDOWN	String	

data.records.posts.markdownContent	**HI**, this is a *++testing++* bala post s ![16137992359659254.jpg](1)	String	
data.records.posts.htmlContent	good-looking	String	
data.records.posts.views	8	Integer	
data.records.posts.approvals	7	Integer	
data.records.posts.comments	0	Integer	
data.records.posts.typeId	5	Integer	post category
data.records.posts.headImg		String	
data.records.posts.official	0	Integer	
data.records.posts.top	0	Integer	
data.records.posts.sort	1000	Integer	
data.records.posts.marrow	0	Integer	
data.records.posts.isDelete	0	Integer	
data.records.posts.createTime	2022-10-10T17:13:34	String	
data.records.posts.updateTime	2022-10-10T17:13:35	String	
data.records.posts.authorName	null	Null	
data.records.posts.type	null	Null	
data.total	2	Integer	
data.size	5	Integer	
data.current	1	Integer	
data.orders		Object	
data.searchCount	true	Boolean	
data.pages	1	Integer	

Response instance: (200) Successful operation

{"code":200,"msg":"Operation is successful","data":{"records":[{"id":14,"postId":5,"userId":4,"posts":{"id":5,"auditState":"PASS","category":"ARTICLE","authorId":6,"title":"Gin is super handsome","contentType":"MARKDOWN","markdownContent":"HI**, this is a *++testing++* bala post\ns\n![16137992359659254.jpg](1)","htmlContent":"good-looking","views":8,"approvals":7,"comments":0,"typeId":5,"headImg":"","official":0,"top":0,"sort":1000,"marrow":0,"isDelete":0,"createTime":"2022-10-10T17:13:34","updateTime":"2022-10-10T17:13:35","authorName":null,"type":null}},{"id":15,"postId":7,"userId":4,"posts":{"id":7,"auditState":"PASS","category":"ARTICLE","authorId":6,"title":"pic test","contentType":"MARKDOWN","markdownContent":"![go](https://image.shutterstock.com/image-vector/go-vector-lettering-isolated-on-260nw-1717017016.jpg)","htmlContent":"test","views":79,"approvals":9,"comments":2,"typeId":5,"headImg":"","official":0,"top":0,"sort":1000,"marrow":1,"isDelete":0,"createTime":"2022-10-10T11:15:57","updateTime":"2022-10-10T11:15:58","authorName":null,"type":null}}],"total":2,"size":5,"current":1,"orders":[],"searchCount":true,"pages":1}}**

API Documentation for Sprint 0

1.1.1 Post/edit post

URL: http://localhost:8888/posts/index/edit

Content-Type: application/json

Method: POST

Description: Posting and edit post [if request param contains id(edit) else (add new)]

Request header:

Name	Type	Required	Description
Authorization	String	Yes	Jwt token: used for authentication

Request parameter:

Name	Type	Required	Description
title	String	Yes	Title
category	String	Yes	Category
markdownContent	String	Yes	Content
id	Integer	No	ID

Request instance:


```
{
    "title":"Title",
    "category":"MOVIE",
    "markdownContent":"This is a post"
}
```


Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	Satus code
msg	Operation is successful	String	Return message
data.id	7	Integer	ID
data.auditState	PASS	String	Audit state, default pass
data.category	MOVIE	String	Category
data.authorId	7	Integer	Author ID
data.title	Title	String	Title
data.contentType	ARTICLE	String	Content type
data.markdownContent	This is a post	String	Markdown content
data.htmlContent	html	String	Html content
data.createTime	2022-09-24T21:57:32.917	String	Create time

Response instance: (200) Successful operation


```
{
"code": 200,
"msg": "Operation is successful",
"data": {
"id": 7,
"auditState": "PASS",
"category": "MOVIE",
"authorId": 7,
"title": "Title",
"contentType": "ARTICLE",
"markdownContent": "This is a post",
"htmlContent": "html",
"createTime": "2022-09-24T21:57:32.917"
}
```

}

Response: (401) NotAuth

Name	Example	Type	Description
code	401	Integer	Status code
msg	The current Subject is not authenticated. Access denied.	String	Return message
data	null	Null	Return data

Response instance: (401) NotAuth

{

"code": 401,

"msg": "The current Subject is not authenticated. Access denied.",

"data": null

}

1.1.2 Access a post

URL: http://localhost:8888/posts/index/7

Content-Type: multipart/form-data

Method: GET

Description: Access a post by using its ID

Request: Append the ID of post to URL http://localhost:8888/posts/index/

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	Status code
msg	Operation is successful	String	Return message
data.id	7	Integer	ID
data.auditState	PASS	String	Audit state
data.category	ANIME	String	Category
data.authorId	7	Integer	Author ID
data.title	Title	String	Title
data.contentType	ARTICLE	String	Content type
data.markdownContent	This is a post	String	Markdown Content
data.htmlContent	html	String	Html content
data.views	0	Integer	Viewed number
data.approvals	0	Integer	Approval number
data.comments	0	Integer	Comment number
data.typeId	0	Integer	Type ID
data.official	0	Integer	Official post
data.top	0	Integer	Top
data.sort	1000	Integer	Sort
data.marrow	0	Integer	Marrow

data.isDelete	0	Integer	Deleted 1, not 0
data.createTime	2022-09-22T23:28:01	String	Create time
data.updateTime	2022-09-22T15:28:01	String	Update time

Response instance: (200) Successful operation


```
{
"code": 200,
"msg": "Operation is successful",
"data": {
"id": 7,
"auditState": "PASS",
"category": "ANIME",
"authorId": 7,
"title": "Title",
"contentType": "ARTICLE",
"markdownContent": "This is a post",
"htmlContent": "html",
"views": 0,
"approvals": 0,
"comments": 0,
"typeId": 0,
"headImg": "",
"official": 0,
"top": 0,
"sort": 1000,
"marrow": 0,
"isDelete": 0,
"createTime": "2022-09-22T23:28:01",
"updateTime": "2022-09-22T15:28:01"
}
}
```


Response: (400) NoPostData

Name	Example	Type	Description
code	400	Integer	Status code
msg	The post has been deleted	String	Return message
data	null	Null	Return data

Response instance: (400) NoPostData


```
{
"code": 400,
"msg": "The post has been deleted",
```

"data": null

}

1.1.3 Post List

URL: http://localhost:8888/posts/index/list?pageNum=1&pageSize=2

Content-Type: multipart/form-data

Method: GET

Description: List of posts used to display in home page

URI parameters:

Name	Type	Required	Description
pageNum	Integer	No	Page number
pageSize	Integer	No	Page size

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	Status code
msg	Operation is successful	String	Return message
data.records	[{},{}]	Object	List of post
data.records.id	7	Integer	ID
data.records.auditState	PASS	String	Audit state
data.records.category	ANIME	String	Category
data.records.authorId	7	Integer	Author id
data.records.title	Title	String	Title
data.records.contentType	ARTICLE	String	Content type
data.records.markdownContent	This is a post	String	Markdown content
data.records.htmlContent	html	String	Html content
data.records.views	0	Integer	Viewed number
data.records.approvals	0	Integer	Approval number
data.records.comments	0	Integer	Comment number
data.records.typeId	0	Integer	Type ID
data.records.headImg		String	Head image
data.records.official	0	Integer	Official post
data.records.top	0	Integer	Top
data.records.sort	1000	Integer	Sort
data.records.marrow	0	Integer	Marrow
data.records.isDelete	0	Integer	Deleted 1, not 0
data.records.createTime	2022-09-22T23:28:01	String	Create time
data.records.updateTime	2022-09-22T15:28:01	String	Update time
data.total	7	Integer	Total record
data.size	2	Integer	Page size
data.current	1	Integer	Current page

data.pages	4	Integer	Total page

Response instance: (200) Successful operation


```
{
"code": 200,
"msg": "Operation is successful",
"data": {
"records": [
{
"id": 7,
"auditState": "PASS",
"category": "ANIME",
"authorId": 1,
"title": "Title",
"contentType": "ARTICLE",
"markdownContent": "This is a post",
"htmlContent": "html",
"views": 0,
"approvals": 0,
"comments": 0,
"typeId": 0,
"headImg": "",
"official": 0,
"top": 0,
"sort": 1000,
"marrow": 0,
"isDelete": 0,
"createTime": "2022-09-22T23:28:01",
"updateTime": "2022-09-22T15:28:01"
},
{
"id": 6,
"auditState": "PASS",
"category": "PLAY",
"authorId": 1,
"title": "Title",
"contentType": "ARTICLE",
"markdownContent": "This is a post",
"htmlContent": "html",
"views": 0,
"approvals": 0,
```

"comments": 0,

"typeId": 0,

"headImg": "",

"official": 0,

"top": 0,

"sort": 1000,

"marrow": 0,

"isDelete": 0,

"createTime": "2022-09-22T23:00:59",

"updateTime": "2022-09-22T15:27:10"

}

],

"total": 7,

"size": 2,

"current": 1,

"orders": [],

"searchCount": true,

"pages": 4

}

}

Response: (200) NoData

Name	Example	Type	Description
code	200	Integer	Status code
msg	Operation is successful	String	Return message
data.total	0	Integer	Total record
data.size	2	Integer	Page size
data.current	1	Integer	Current page
data.pages	1	Integer	Total page

Response instance: (200) NoData

{

"code": 200,

"msg": "Operation is successful",

"data": {

"records": [],

"total": 7,

"size": 10,

"current": 2,

"orders": [],

"searchCount": true,

"pages": 1

}

}

1.1.4 Update password

URL: http://localhost:8888/user/editPassword

Content-Type: application/json

Method: POST

Description: Update password of a account

Request header:

Name	Type	Required	Description
Authorization	String	Yes	Jwt token: used for authentication

Request parameters:

Name	Type	Required	Description
id	Integer	Yes	User ID
password	String	Yes	Password

Request instance:

{

"id": 1,

"password":"12345678"

}

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	Status code
msg	Operation is successful	String	Return message
data	vin@yan.com	String	User email

Response instance: (200) Successful operation

{

"code": 200,

"msg": "Operation is successful",

"data": "vin@yan.com"

}

Response: (400) UserNotExist

Name	Example	Type	Description
code	400	Integer	Status code
msg	user is not exist	String	Return message

data	null	Null	Return data

Response instance: (400) UserNotExist


```
{
"code": 400,
"msg": "user is not exist",
"data": null
}
```


Response: (401) NotAuth

Name	Example	Type	Description
code	401	Integer	Status code
msg	The current Subject is not authenticated. Access denied.	String	Return message
data	null	Null	Return data

Response instance: (401) NotAuth


```
{
"code": 401,
"msg": "The current Subject is not authenticated.  Access denied.",
"data": null
}
```


1.1.5 Edit user profile

URL: http://localhost:8888/user/edit

Content-Type: application/json

Method: POST

Description: Edit user profile

Request header:

Name	Type	Required	Description
Authorization	String	Yes	Jwt token: used for authentication

Request parameters:

Name	Type	Required	Description
id	Integer	Yes	User id
email	String	Yes	Email
password	String	Yes	Password
gender	String	Yes	Gender
username	String	Yes	Username
signature	String	Yes	Signature

Request instance:

```
{
    "id": 1,
    "email":"vin@yan.com",
    "password":"123456",
    "gender":"male",
    "username":"Vin",
    "signature":"Vin"
}
```


Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	Status code
msg	Operation is successful	String	Return message
data.id	1	Integer	ID
data.email	vin@yan.com	String	Email
data.username	Vin	String	Username
data.password	123456	String	Password
data.role	USER	Null	Role
data.state	ENABLED	Null	state: DISABLED / ENABLED
data.gender	male	String	Gender
data.avatar	null	Null	Avatar
data.signature	Vin	String	Signature
data.isDelete	0	Null	Deleted 1, not 0
data.createTime	2022-09-24T21:32:51.751	Null	Create time
data.updateTime	2022-09-24T21:32:51.751	String	Update time

Response instance: (200) Successful operation


```
{
"code": 200,
"msg": "Operation is successful",
"data": {
"id": 1,
"email": "vin@yan.com",
"username": "Vin",
"password": "123456",
"role": USER,
"state": ENABLED,
"gender": "male",
"avatar": null,
"signature": "Vin",
"isDelete": 0,
"createTime": "2022-09-24T21:32:51.751",
```

"updateTime": "2022-09-24T21:32:51.751"

}

}

Response: (400) UserNotExist

Name	Example	Type	Description
code	400	Integer	Status code
msg	user is not exist	String	Return message
data	null	Null	Return data

Response instance: (400) UserNotExist

{

"code": 400,

"msg": "user is not exist",

"data": null

}

Response: (401) NotAuth

Name	Example	Type	Description
code	401	Integer	Status code
msg	The current Subject is not authenticated. Access denied.	String	Return message
data	null	Null	Return data

Response instance: (401) NotAuth

{

"code": 401,

"msg": "The current Subject is not authenticated. Access denied.",

"data": null

}

Response: (400) EmailRegistered

Name	Example	Type	Description
code	400	Integer	Status code
msg	The email has been registered	String	Return message
data	null	Null	Return data

Response instance: (400) EmailRegistered

{

"code": 400,

"msg": "The email has been registered",

"data": null

}

1.1.6 User registration

URL: http://localhost:8888/user/register

Content-Type: application/json

Method: POST

Description: Register a new account

Request parameters:

Name	Type	Required	Description
email	String	Yes	Email
password	String	Yes	Password
gender	String	Yes	Gender
username	String	Yes	Username
signature	String	Yes	Signature

Request instance:

```
{
    "email":"vin@yan.com",
    "password":"123456",
    "gender":"male",
    "username":"Vin",
    "signature":"Vin"
}
```

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	Status code
msg	Operation is successful	String	Return message
data.id	1	Integer	ID
data.email	vin@yan.com	String	Email
data.username	Vin	String	username
data.password	123456	String	password
data.role	USER	String	Role
data.state	ENABLED	String	State: DISABLED / ENABLED
data.gender	male	String	Gender
data.avatar	null	Null	Avatar
data.signature	Vin	String	Signature
data.isDelete	0	Null	Deleted 1, not 0
data.createTime	2022-09-24T21:22:10.727	String	Create time
data.updateTime	2022-09-24T21:22:10.727	String	Update time

Response instance: (200) Successful operation

```
{
"code": 200,
"msg": "Operation is successful",
"data": {
"id": 1,
"email": "vin@yan.com",
"username": "Vin",
"password": "123456",
"role": "USER",
"state": "ENABLED",
"gender": "male",
"avatar": null,
"signature": "Vin",
"isDelete": 0,
"createTime": "2022-09-24T21:22:10.727",
"updateTime": "2022-09-24T21:22:10.727"
}
}
```


Response: (400) EmailExisted

Name	Example	Type	Description
code	400	Integer	Status code
msg	The email has been registered	String	Return message
data	null	Null	Return data

Response instance: (400) EmailExisted


```
{
"code": 400,
"msg": "The email has been registered",
"data": null
}
```


Response: (400) EmptyEmail

Name	Example	Type	Description
code	400	Integer	Status code
msg	the email can't be empty	String	Return message
data	null	Null	Return data

Response instance: (400) EmptyEmail


```
{
"code": 400,
"msg": "the email can't be empty",
```

"data": null

}

Response: (400) EmptyPassword

Name	Example	Type	Description
code	400	Integer	Status code
msg	the password can't be empty	String	Return message
data	null	Null	Return data

Response instance: (400) EmptyPassword

{

"code": 400,

"msg": "the password can't be empty",

"data": null

}

Response: (400) EmptyUsername

Name	Example	Type	Description
code	400	Integer	Status code
msg	the username can't be empty	String	Return message
data	null	Null	Return data

Response instance: (400) EmptyUsername

{

"code": 400,

"msg": "the username can't be empty",

"data": null

}

1.1.7 Get user profile

URL: http://localhost:8888/user/7

Content-Type: multipart/form-data

Method: GET

Description: obtain user information based on user ID

Request header:

Name	Type	Required	Description
Authorization	String	Yes	Jwt token: used for authentication

Request: Append the ID of user to URL http://localhost:8888/user/7

Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	Status code
msg	Operation is successful	String	Return message
data.id	1	Integer	ID
data.email	vin@yan.com	String	Email
data.username	Vin	String	Username
data.password	123456	String	Password
data.role	USER	String	Role
data.state	ENABLED	String	State: DISABLED / ENABLED
data.gender	male	String	Gender
data.avatar	https://image.com/avatar	String	Avatar
data.signature	Vin	String	Signature
data.isDelete	0	Integer	Deleted 1, not 0
data.createTime	2022-09-22T22:09:08	String	Create time
data.updateTime	2022-09-22T15:47:55	String	Update time

Response instance: (200) Successful operation

{

"code": 200,

"msg": "Operation is successful",

"data": {

"id": 1,

"email": "vin@yan.com",

"username": "Vin",

"password": "123456",

"role": "USER",

"state": "ENABLED",

"gender": "male",

"avatar": "https://image.com/avatar",

"signature": "Vin",

"isDelete": 0,

"createTime": "2022-09-22T22:09:08",

"updateTime": "2022-09-22T15:47:55"

}

}

Response: (400) UserNotExist

Name	Example	Type	Description
code	400	Integer	Status code
msg	user is not exist	String	Return message
data	null	Null	Return data

Response instance: (400) UserNotExist

```
{
"code": 400,
"msg": "user is not exist",
"data": null
}
```


Response: (401) NotAuth

Name	Example	Type	Description
code	401	Integer	Status code
msg	The current Subject is not authenticated. Access denied.	String	Return message
data	null	Null	Return data

Response instance: (401) NotAuth


```
{
"code": 401,
"msg": "The current Subject is not authenticated.  Access denied.",
"data": null
}
```


1.1.8 User login

URL: http://localhost:8888/login

Content-Type: application/json

Method: POST

Description: Used for user login

Request parameters:

Name	Type	Required	Description
email	String	Yes	User login email
password	String	Yes	Password

Request instance:


```
{
    "email":"vin@yan.com",
    "password":"123456"
}
```


Response: (200) Successful operation

Name	Example	Type	Description
code	200	Integer	Status code
msg	Operation is successful	String	Return message
data.id	1	Integer	User id

data.avatar	https://image.com/avatar	String	Avatar
data.email	vin@yan.com	String	Email
data.username	Vin	String	Username

Response instance: (200) Successful operation

{

"code": 200,

"msg": "Operation is successful",

"data": {

"id": 1,

"avatar": "https://image.com/avatar",

"email": "vin@yan.com",

"username": "Vin"

}

}

Response: (400) IncorrectPassword

Name	Example	Type	Description
code	400	Integer	Status code
msg	Incorrect password	String	Return message
data	null	Null	Return data

Response instance: (400) IncorrectPassword

{

"code": 400,

"msg": "Incorrect password",

"data": null

}

Response: (400) UserNotExist

Name	Example	Type	Description
code	400	Integer	Status code
msg	User does not exist	String	Return message
data	null	Null	Return data

Response instance: (400) UserNotExist

{

"code": 400,

"msg": "User does not exist",

"data": null

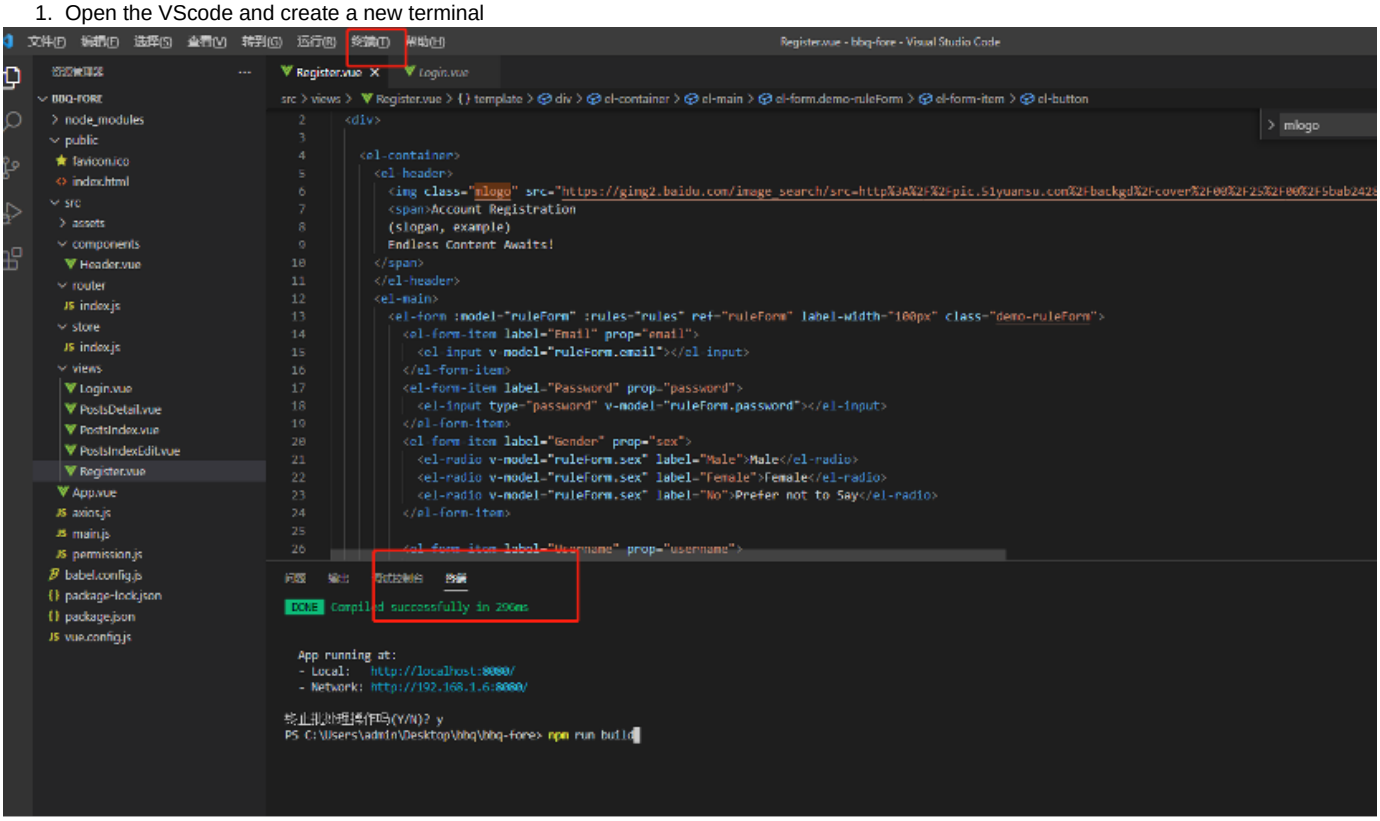
}

Deployment instruction

This article will briefly introduce how to deploy the current version of the software.

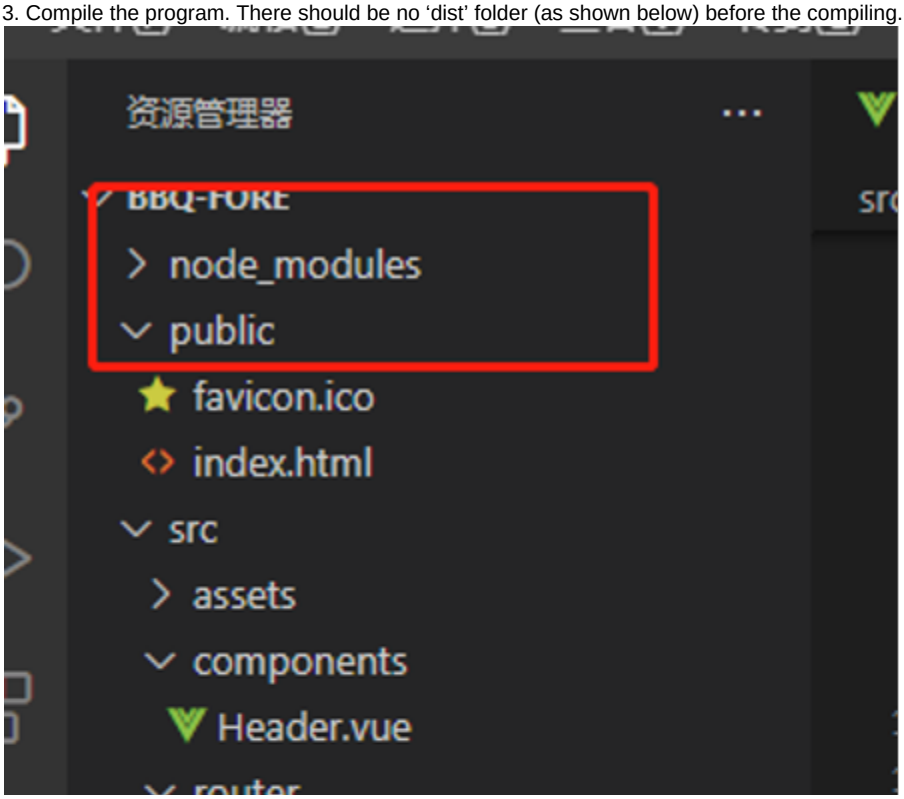
Front end part

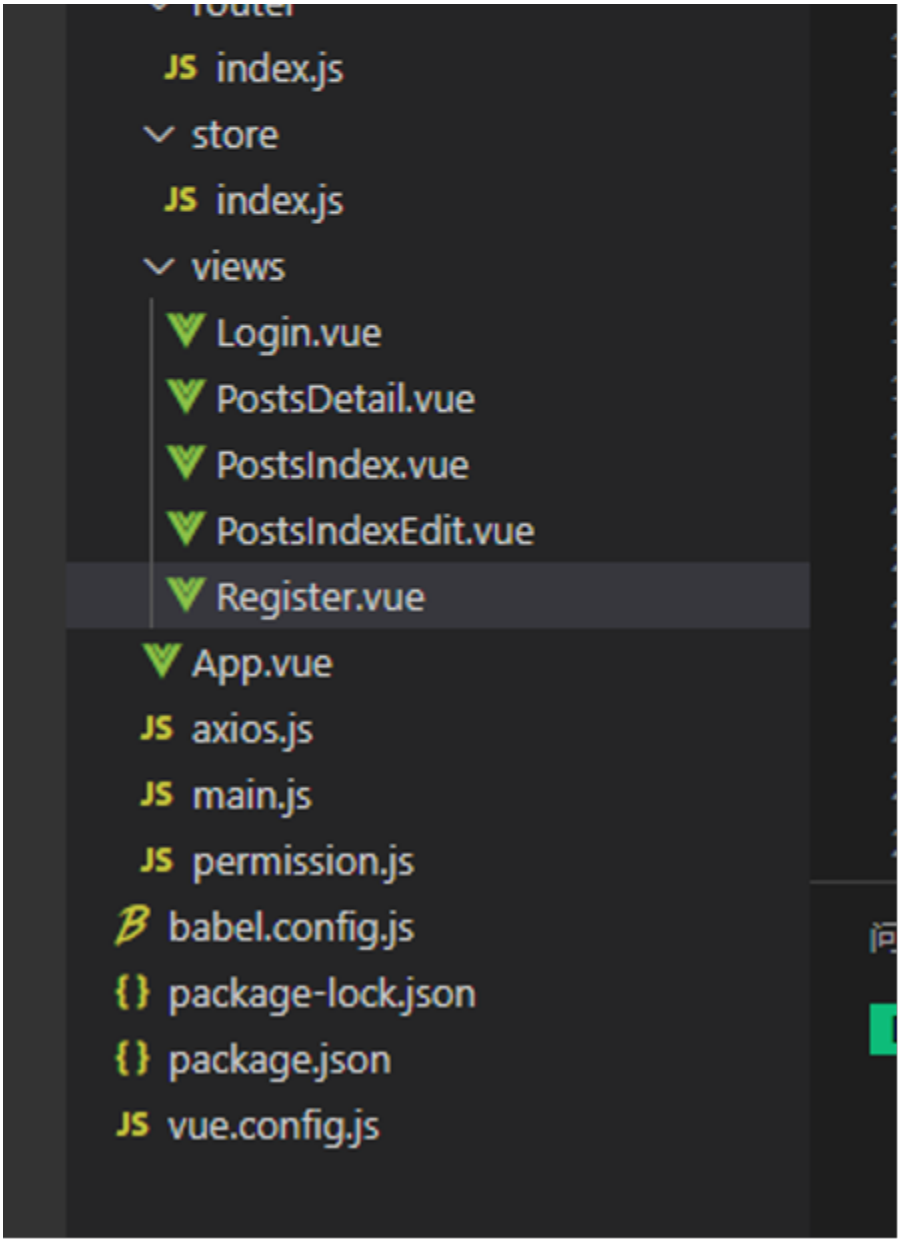
1. Open the VScode and create a new terminal

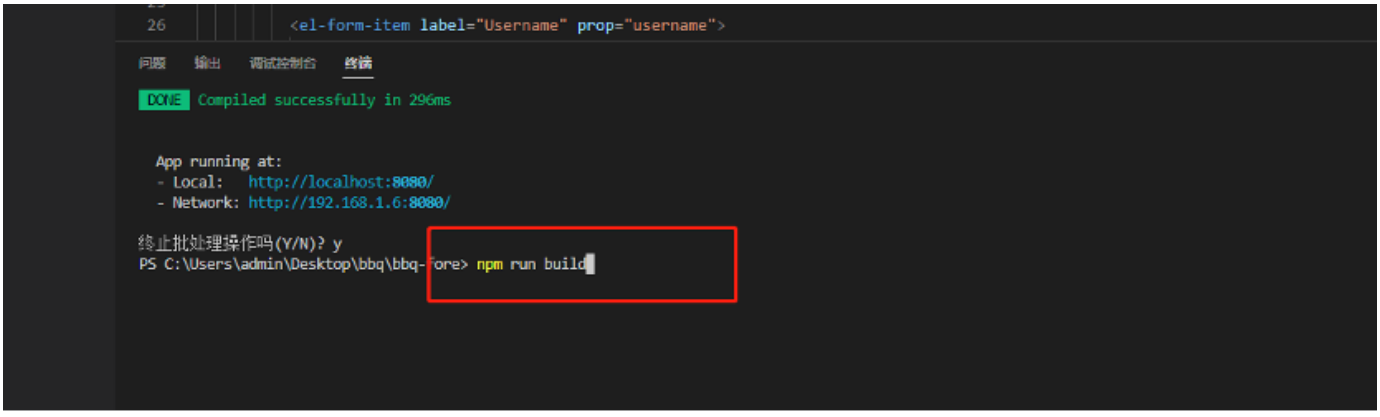


2. Run “npm install” on terminal and wait for it to finish. If errors happened here, follow the instruction on the screen.

3. Compile the program. There should be no ‘dist’ folder (as shown below) before the compiling. Use “npm run build” to compile the program.







4. When compilation is finished, there should be a ‘dist’ folder. Zip that folder.

5. Create a new directory under /root folder on server using the following commands:

mkdir nginx

cd nginx

mkdir html

upload the zip package of ‘dist.zip’ from previous step under ‘html’ folder using following command:

mv dist/ .*

rm –rf dist*

then run:

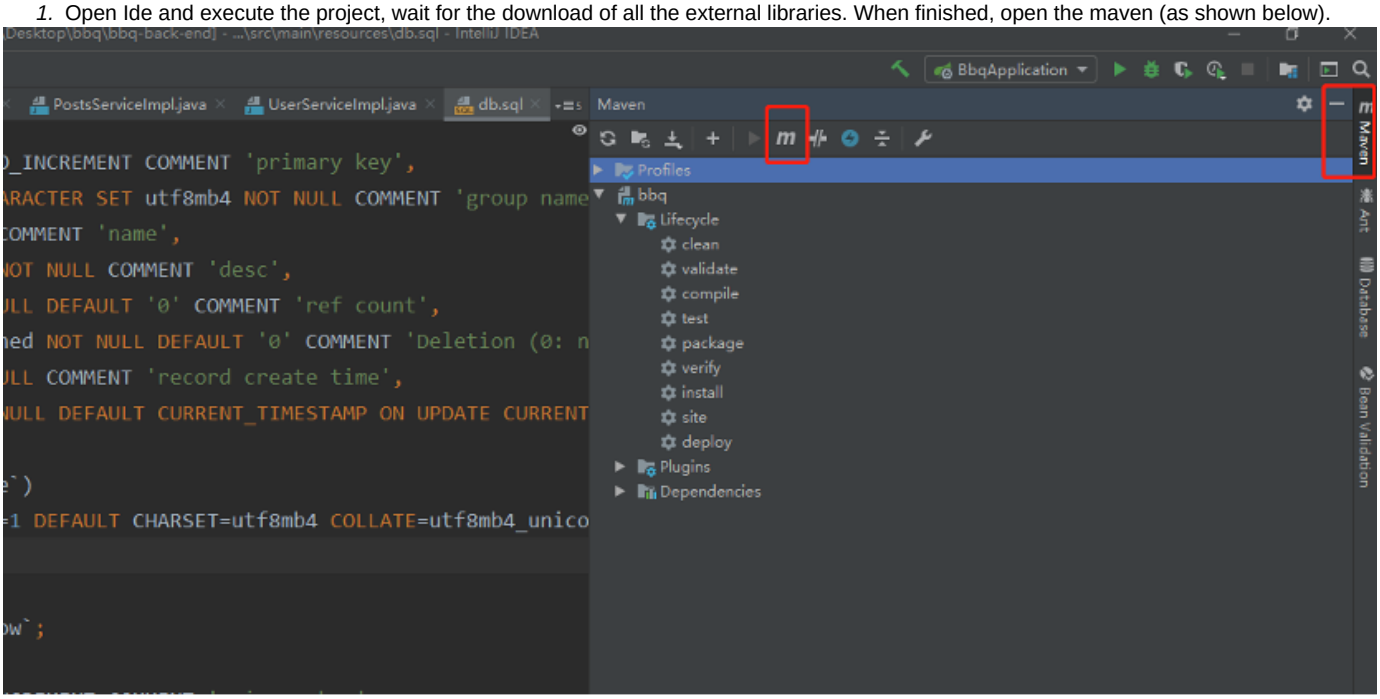
unzip dist.zip

to unzip the folder

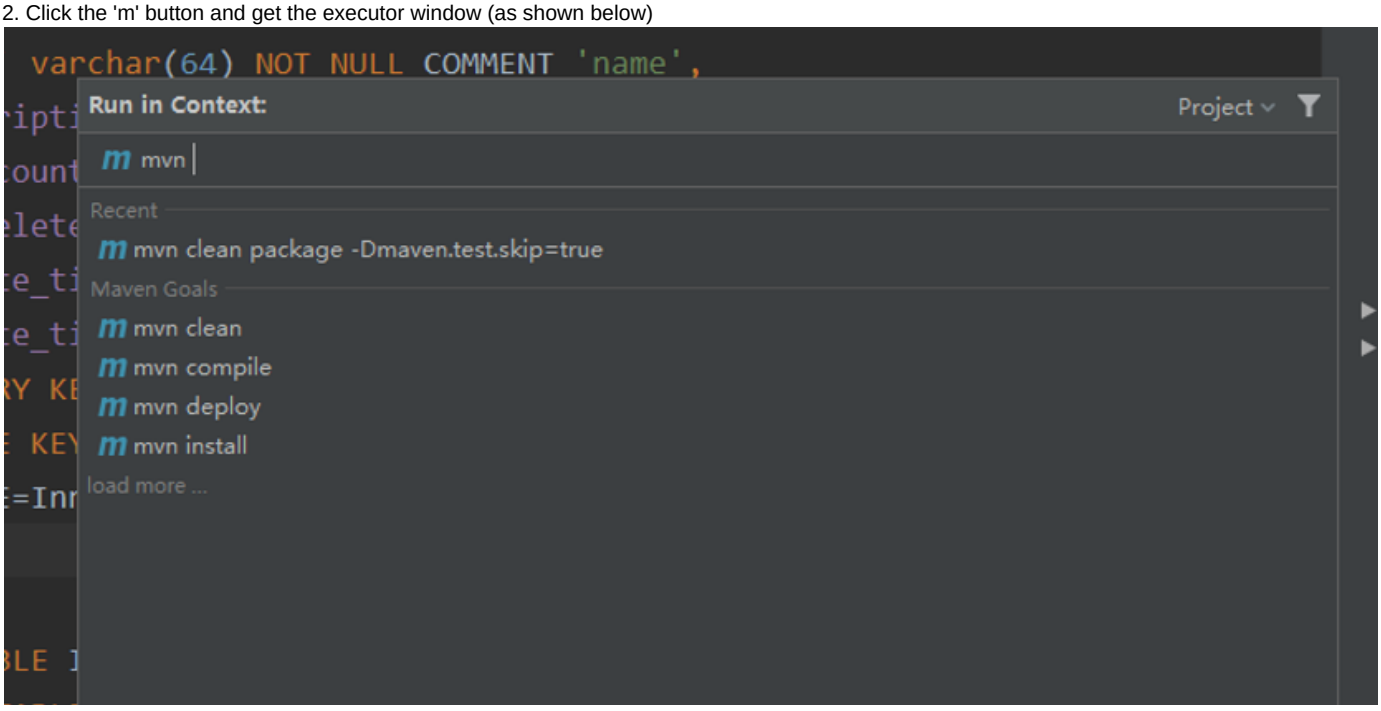
6. upload the front end document ‘nginx.conf’ to /root/nginx

Backend Part

1. Open Ide and execute the project, wait for the download of all the external libraries. When finished, open the maven (as shown below).



2. Click the 'm' button and get the executor window (as shown below)



3. execute the command below:

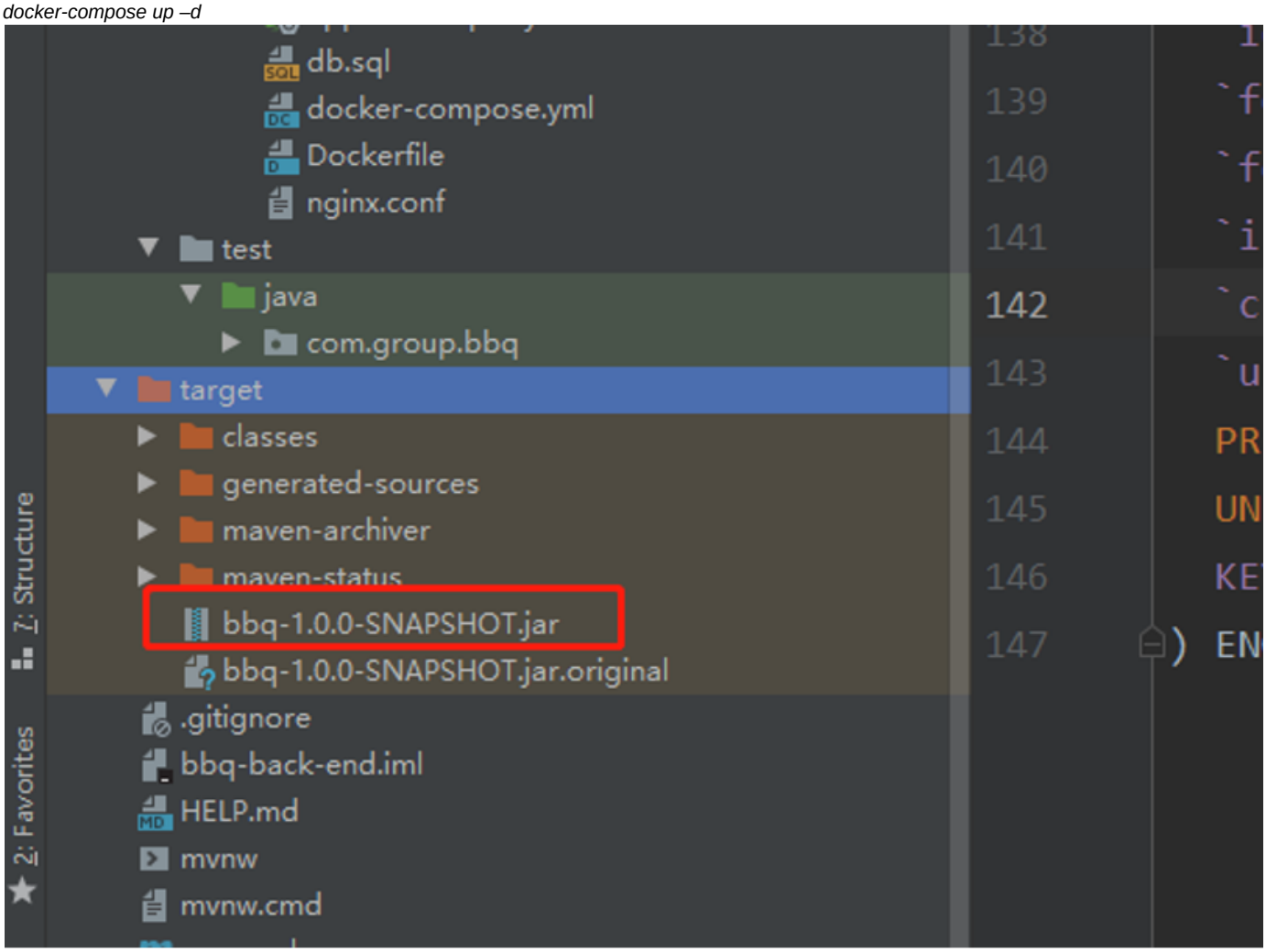
mvn clean package -Dmaven.test.skip=true

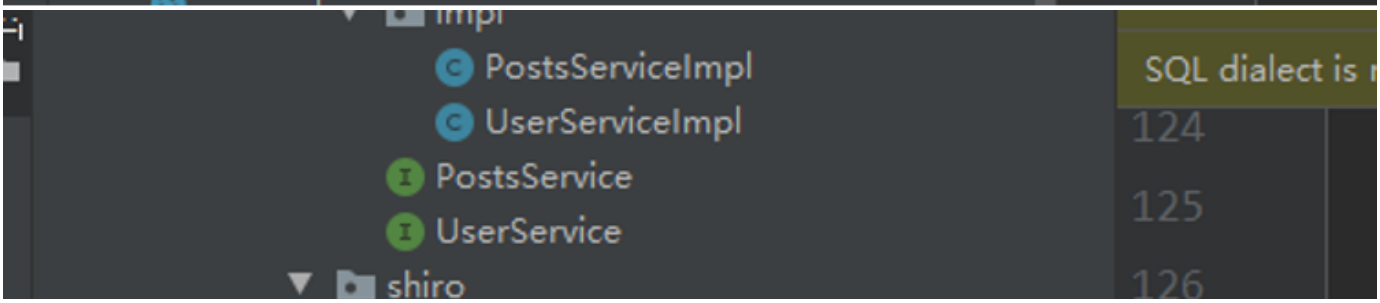
to compile the back-end into a jar package

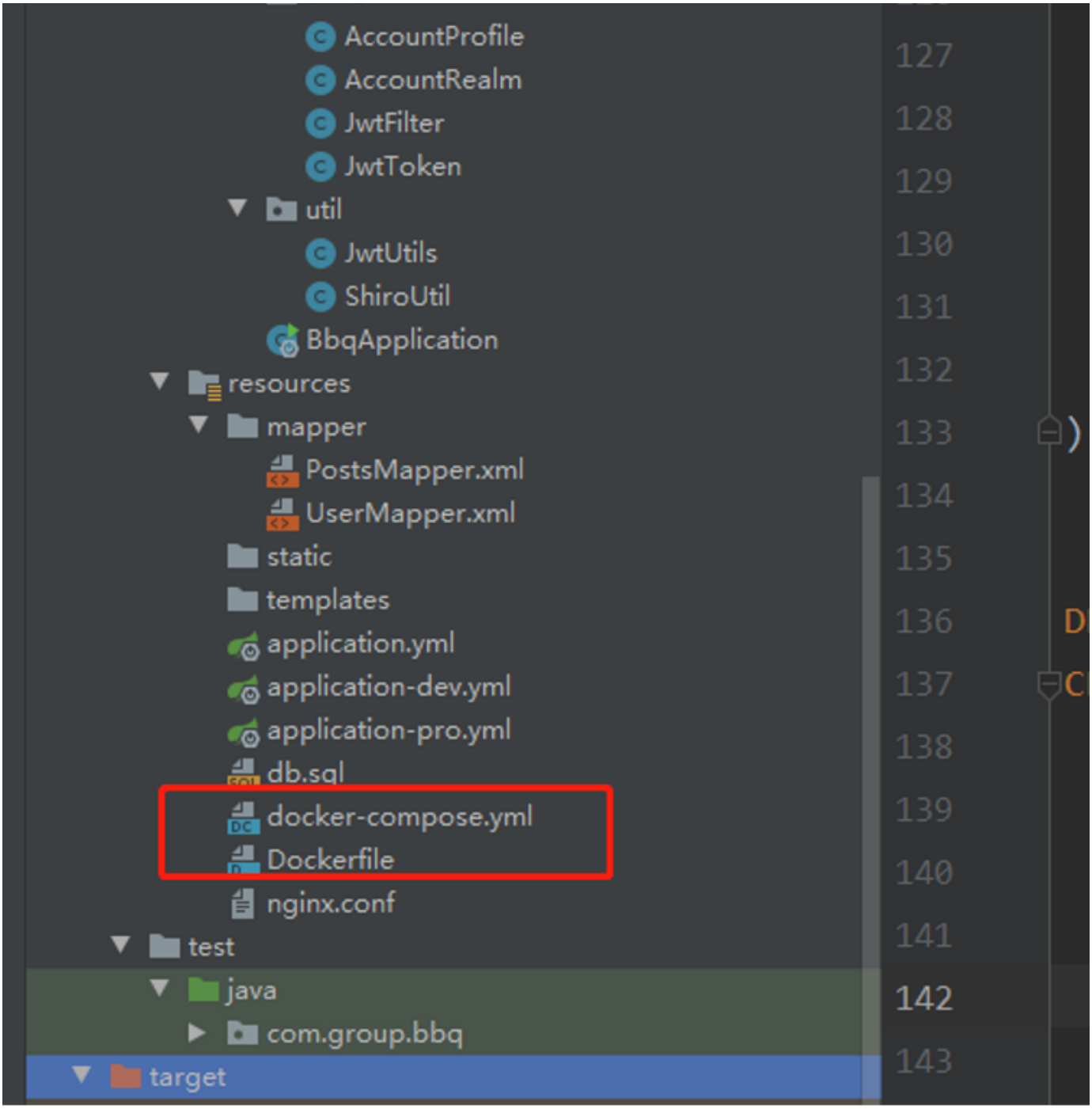
once compiling is finished, a ‘target’ folder will be shown.

4. upload the “bbq-1.0.0-SPAPSHOT.jar“ and “Dockfile docker-compose.yml“ in taget folder to ‘/root’ directory of server. Run the following command in ‘/root’ folder to run server:

docker-compose up –d







Integration Testing

Below is the table which briefly introduced the process of the integration testing plan.

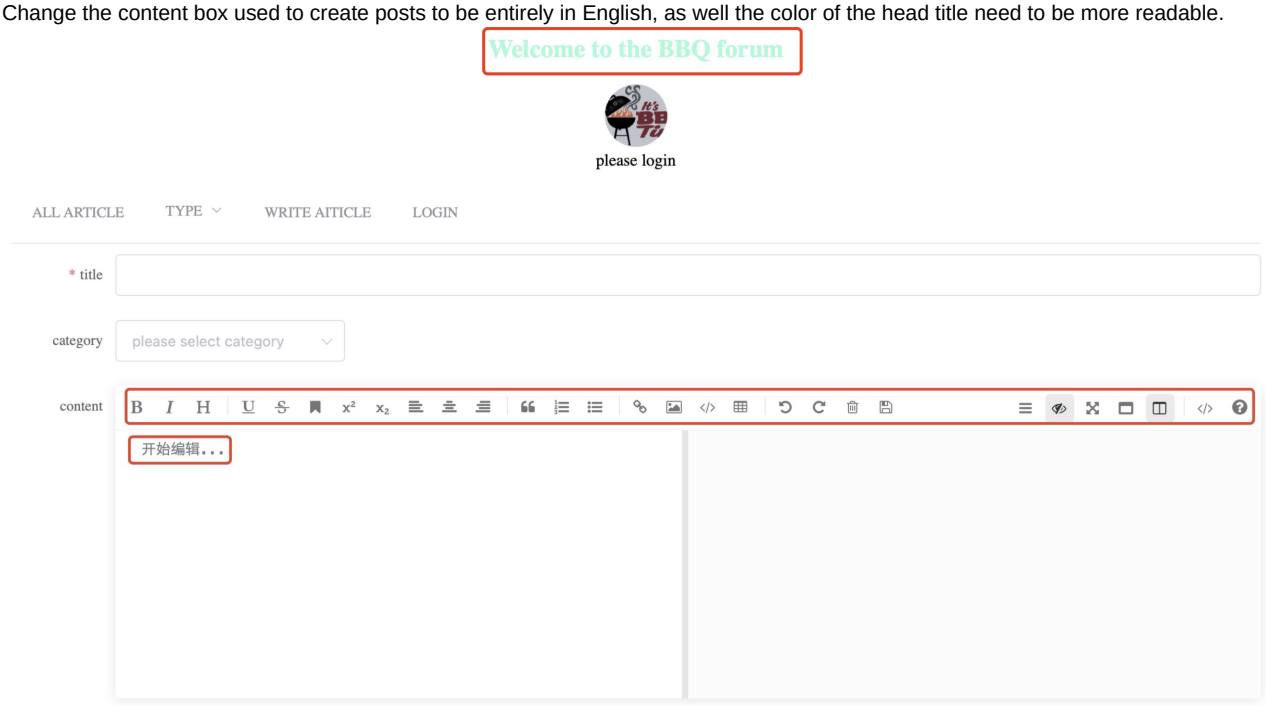
function	testcase	expected result	meets expected result?
Signup process: 1. Input and submit details on the front-end 1. Verify these details in back-end	submitForm(validUserData): creates user using supplied user data and calls router **register** , using POST which then calls back-end controller.register() function for registration	Logs "register successful" and redirects user to login page	Yes: Popup with Logs "register successful" and redirects user to login page

function 1. Successfully store these details in database			
Signup process: Invalid details test	1. submitForm(incompletedInfo) 2. send a registered email to backend	Logs “the email/password can’t be empty” logs “The email has been registered”	Yes, it does meet the expected result. Yes: Logs “The email has been registered”
Login process 1. Input and submit login details on the front-end 2. Verify these details with records in database 3. Allow verified user through to their account, or deny access if details are incorrect	submitForm(realUserData) Front-end function, calls router **Login**, using POST then calls back-end controller.login() function for authentication, then store response : Jwt token and user info	Redirects user to post index page.	Yes: redirects user to post index page.
Login process Incorrect credentials test	submitForm(fakeUserData)	Logs "Incorrect password” Logs “User does not exist“	Yes, it provides the correct output for incorrect passwords Yes, it provides the correct output for invalid users
Post process: 1. Input and submit details on the front- end 2. Send to back end and check authorization 3. Successfully store these details in database	submitForm(postsDetail)	Popup with “Operation is successful” Redirect to index once click the bottom	Yes, as expected.
Comment process: 1. toEdit() from front end 2. send to back end and check authorization 3. return 'follow successful' to front end and store comments in database	Comment on random post with “test“	Popup with info “comment successful”, the post will show a new comment after clicking the bottom.	Yes, click the bottom then back to the post.
Approval process: 1. approvals() from front end 2. send to back end by calling approval() and check authorization 3. store changes in database	Click the approval bottom on a post.	Popup with info “comment successful”, the number of the thumb up +1.	Yes.
Delete post process: 1. toDelete() from front end 2. send to back end and check authorization 3. return ‘delete successful’	First post one post and then find it in the index page, then click delete bottom.	Popup “delete successful“ and can not find it anymore in the index page after deleting.	Yes.
Search post process:			Both yes.

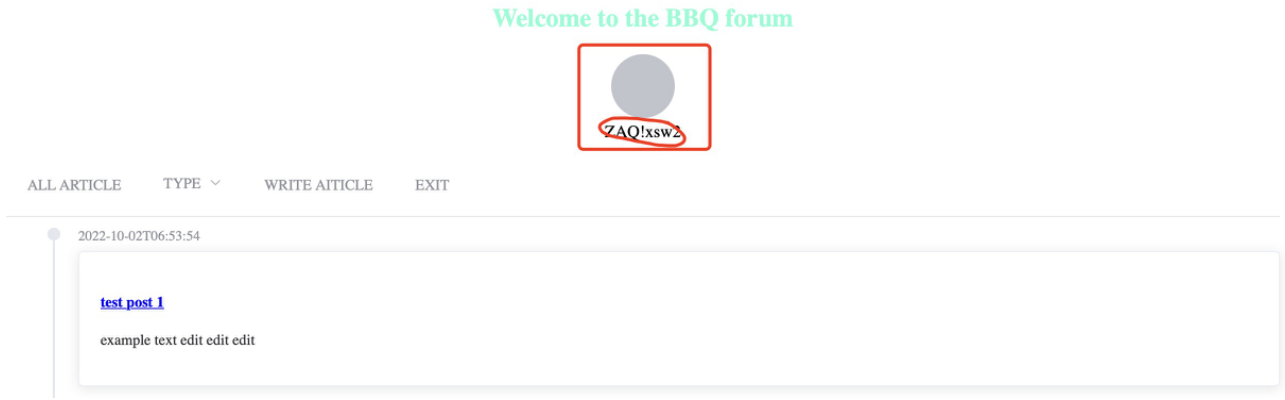
1. page(currentPage) from front 2. call indexList() in back end 3. return a page class for represent	1. Select one existed post, search with one of its keyword 2. Search with a not existed keyword	1. Show the list of all post within the key word. 2. Return empty page	

Improvements that need to be made

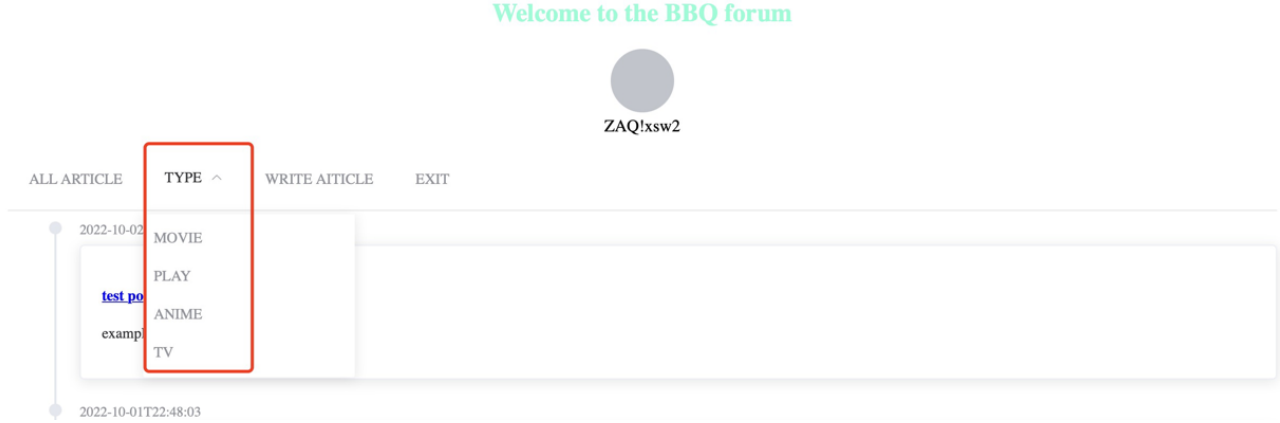
1. Change the content box used to create posts to be entirely in English, as well the color of the head title need to be more readable.

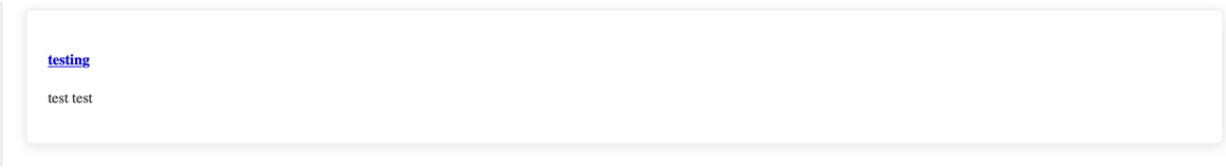


2. Password is shown instead of username at the top of the page when logged in

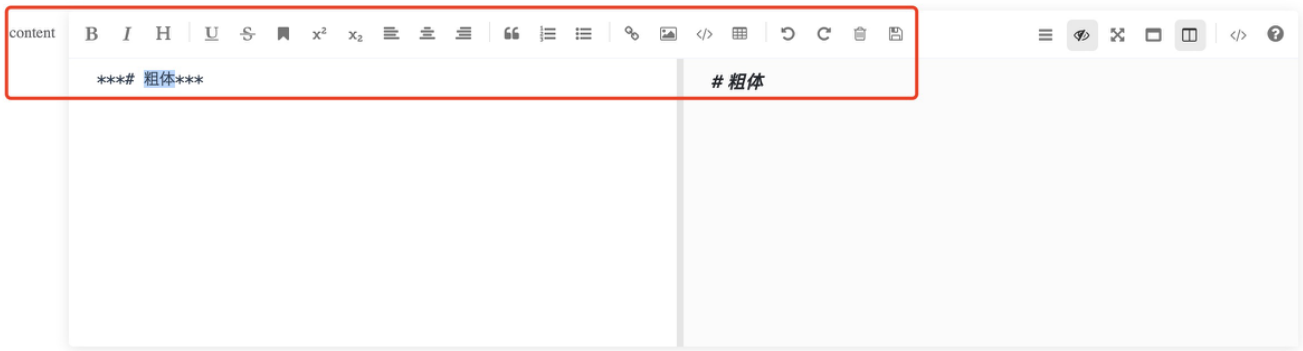


3. Sorting by a category does not change the order of posts, additionally posts do not show anywhere which category they belong to



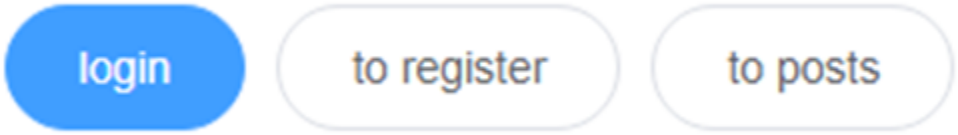


4. Examples of text functions such as **bold**, *italic,* underlined display their example in Mandarin



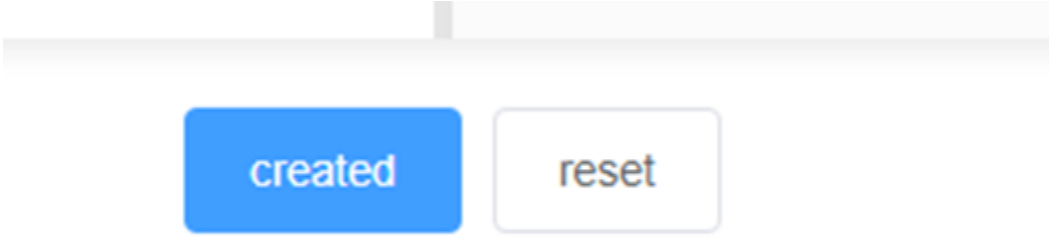
5. **Readability Changes**

a. ALL ARTICLE -> ALL POSTS
b. TYPE -> CATEGORY
c. WRITE AITICLE -> CREATE POST



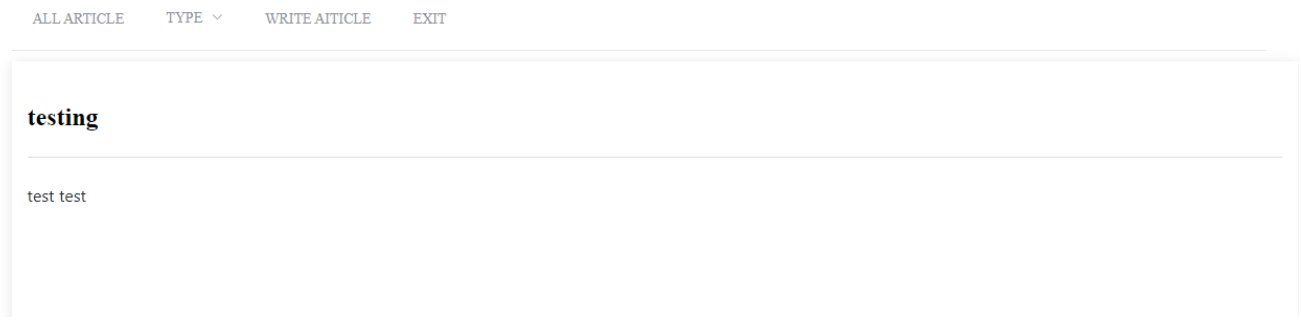
d. To register -> register
e. To posts -> home

f. Register successful -> Registration successful



g. Created -> Post

6. The writer of the post is not mentioned anywhere on the post, as well as the post's category. It should be contained somewhere near the header for readers.



7. The category in the homepage(fig1) is unmatched to the new post page(fig2)