香 港 中 文 大 學（深 圳）

The Chinese University of Hong Kong, Shenzhen

School of Data Science

CSC4001: Software Engineering

Course Project Report, Group 18
CUHKSZ-Overflow: An Online Forum For CUHKSZ Programmers

Author:

Huang Pengxiang 119010108

Li Zihan 119010167

Linghu Han 119010185

Qin Peiran 119010249

May, 2022

1 Introduction

1.1 Background & Motivation

It is common for many student-programmers from CUHKSZ encounter many familiar programming questions in their projects or assignments. To solve those questions, their solutions mainly include: searching online, emailing TAs or professors for help, uploading questions in class WeChat group, or making appointments with TAs or professors during their office hours. Searching online sometimes may not be efficient because the blogs or some guidance information may not directly answer the assignment question. Even worse, programmers need to spend enormous of time filtering a vast amount of information. It becomes hard for them to get answers directly when the homework question is not relevant to the results on the website. Uploading questions on the class WeChat group could get detailed guidance and response. But the new WeChat group will be created every semester for other students who may encounter the same problems. The connection between students who have already taken this course with the students taking this course right now is broken in this way. Raising questions during office hours is not convenient for programmers to solve their questions immediately since they need to make an appointment and wait until that day comes. Therefore, our group would like to take the first step to change the current situation by providing an online Q&A platform with the support of an online forum database.

Based on the inspiration from the public Q&A platform Stack-Overflow, our group thinks it is necessary to design a similar Q&A platform for programmers in CUHKSZ. It could help them solve their technical questions in time, saving them much time in finishing homework. Programmers could use this platform to browse their questions on the website. They will get the relevant answer immediately since there are many students who have already taken this course and may encounter similar problems. Also, the platform is more friendly. Programmers could use their nickname rather than the actual name, and it is private for others. Therefore it would not make them shame about their questions. The platform also supports many question formats, which means programmers could use code blocks to express their specific questions rather than a word. It is much more efficient and clearer to use code blocks to express their idea. So students could find help on our website when they encounter some problematic bugs.

Our group also wants to build it as a social community for those programmers who want to raise some interesting topics to discuss with others. Programmers will have a platform to share their learning experience and their suggestions with those younger students. And this friendly environment will also encourage more programmers to learn skills and be more creative and productive.

In the view above, our groups borrow the ideas from the famously used app called Stack-Overflow in the CS community, and we hope to create a CUHKSZ version for programmers in our school. This website can upload the questions relevant to their course homework and support users using code blocks to express their problems and specific need. The website also will rank the high-quality answer and recommend every member in this community to read on their homepage, which raises interest for every community member to learn. By knowing the drawback of the current Q&A mode, this system can hopefully boost programmers' motivation to raise questions and solve questions immediately in CUHKSZ. Thus, empower every CUHKSZer programming skill.

1.2 Project Overview

Our project, CUHKSZ-Stack-Overflow, mainly aims to provide a Q&A platform for CUHKSZ programmers with technical questions about their Computer Science courses' projects or assignments in CUHK(SZ). It also offers a social channel where some of the students, especially senior students, could share their programming and working experience, provide guidance in CS learning, and give specific suggestions to the other younger students who may encounter the same problem or situation. Since many students may be afraid of seeking assistance from the professor and the teaching assistants through WeChat, email, or during office hour, we hope to create an environment where students could bravely post their programming problems while other students or teachers would willingly to see and reply those questions. Hopefully, this kind of communication will help students fully understand their homework, improve their programming skills, and accumulate programming experience. Eventually, that communication and connection could save much time for students learning CS and reduce the teaching load for TA simultaneously, which will definitely enhance both learning and teaching qualities in Computer Science courses.

1.3 Objective

Our thoughts on the system design come from a famous Q&A application, Stack Overflow, which is a public platform building the definitive collection of coding Q&A for professional and enthusiast programmers. In detail, our system is divided into the frontend part, the backend part, database part. In the frontend, users can perform many operations, including registering and logging in to their accounts, posting and replying to the questions, following questions or partitions, giving a like to questions or answers, and setting filters and searching questions. In the backend, developers and servers deal with the data transfer and management, including performing transactions in the database, executing corresponding functions according to the invokes from the frontend, and setting the relevant URL for the connection between the frontend and backend. Moreover, an administrator account is designed for the management of the whole application. In short, our goal is to provide a satisfied and perfect CUHKSZ-Stack-Overflow Q&A system, particularly for programmers in CUHKSZ.

1.4 Key Features

Our system is composed of 6 parts: registration, log-in and logout, searching relevant questions, posting new questions, replying to the questions, and displaying hot questions. Users could register an account by validating their email in the registration part. In the log-in and logout part, users could log in to the system with their own accounts and log out of the system. Users could search the questions with various filter types in the searching part, such as searching the questions in CSC4001 or in the CSC4001 project. In the post part, users could upload their questions with pictures or files, and they can also run their code online in the code compiler provided by our system. In the reply part, users could answer the questions with pictures or files, and they can also give a like to the answers proposed by others. In the display part, some blogs will be displayed in the 'Hot Blogs' part in the descending order of the popularity value. The value is highly related to likes, favors, views, and the creation time. The newer the questions posted, the larger the likes, favors, and views, and the higher the value of popularity. Apart from those six parts, the system provides many humanized functions, such as resetting the username or password and uploading the profile.

1.5 Highlights

1.5.1 Cloud deployment and website access by domain name

Distinguish from running the code in the local terminal and accessing the website by localhost, and our project is deployed on the cloud server. Our project can be accessed directly by typing the URL http://175.178.34.84 without compiling and running the code in the terminal, thus leading to a more efficient way. Furthermore, a specific domain name is applied to make our project more conspicuous, characteristic, and elegant. In this case, our website can be accessed by typing the URL http://www.cuhksz-stackoverflow.cn. In short, we design and implement a website with an attractive domain name being easy to remember and remote access from

other devices.

1.5.2 Search engine implementation - NLP and inverted index design in database

Initially, we used the string match method to divide the whole sentences into several words and find blogs whose title contains those words. It has some limitations. At first, the method cannot detect the words with tense inconsistency. For example, the word “make” in the search contents cannot find blogs whose title contains the word “made”. Additionally, those words that do not have semantic meaning will also be considered in string match, thus causing poor search results. For example, the search content “how to learn programming” may find blogs whose title contains the word “to”. Therefore, we improved our search engine algorithm and adopted the natural language processing algorithm.

Apart from using NLP in the process of searching content, we also use the inverted index in our database. Since the workload of a forward index is huge, which is caused by traversing all the records in a table to determine whether each record contains the related words, for example, if the user takes “kernel” as inputs, record 1, record 2 and record 3 will be retrieved and then record 1 is found to meet the requirements (Table 1), we consider to build a table with inverted index (Table 2). With the application of inverted index, if the user takes “kernel” as inputs, record 2 is found to meet the requirements, and the blog id 1 is obtained to directly retrieve the related blog, which improves the efficiency in getting data from the database.

BLOG_ID	CONTENT
1	How to build kernel
2	How to learn C++
3	How about my code below

Table 1 - Forward Index

ID	WORDS	BLOG_ID
1	build	{1}
2	kernel	{1}
3	learn	{2}
4	code	{3}
..	..	..

Table 2 - Inverted Index

To sum up, the search content will first be split into words, and then the word frequency is calculated to make up a vector (target_vec). For example, if the user takes “I want to learn build kernel” as inputs, the target_vec should be [1, 1, 1] where the first 1 means the word “learn” appears once in the content, the second 1 means the word “build” appears once in the content and the third 1 means the word “kernel” appears once in the content. Second, the inverted index table will be retrieved to get the list of word frequency according to the blogs. For example, the inverted index table is retrieved according to the words in the search content, and the blog_vec can be obtained in the form of [[1, 0, 1, 1], [2, 1, 0, 0]] where the first value in each list represents the blog id, and the rest three values in each list represent the frequency of the words in the blog content. Finally, the similarity is calculated by inner product and then list all the blogs in the descending order of the similarity. For example, [0, 1, 1] inner product with [1, 1, 1] is 2 and [1, 0, 0] inner product with [1, 1, 1] is 1. Therefore, the results contain blog 1 and blog 2, and blog 1 is listed in the former blog 2.

1.5.3 Online complier and code editor highlight

Distinguish from the traditional communication platform, our platform provides a code editor for users to write the code. It is worth noting that the editor supports the highlight of different kinds of programming language, which improves the coding experience of the users and enhances the visualization of the code, thus providing a comfortable and convenient coding environment for users. Moreover, our platform provides an online compiler for users to run the code online. There is no need for users to open the local IDE to edit their codes and run their codes to see the results. With the application of the online compiler, users can directly write the code in the code editor and then run the code to see the results online, which is a kind of convenient and efficient approach.

1.6 Project Statistics

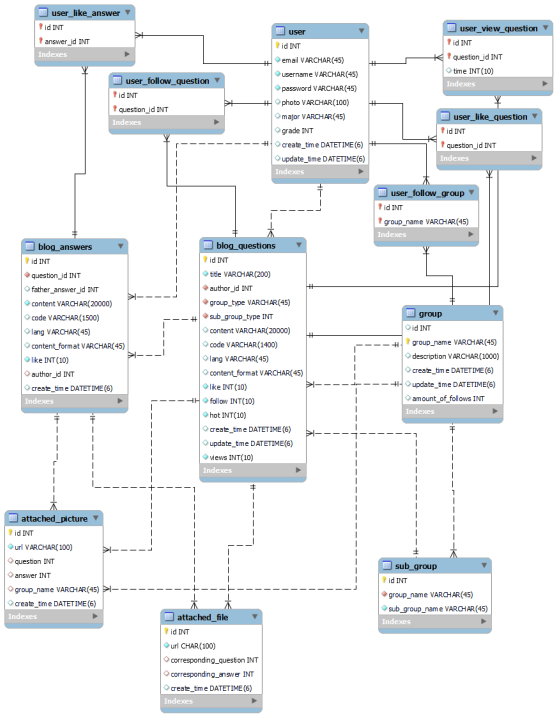
MODULE ID	MODULE NAME	FRONTEND FUNCTIONS	BACKEND FUNCTIONS	DESCRIPTION
1	Login	check the username and password	check if username exists, check if password is correct	Login for users
2	Register	check the username, password and email	check if username exists	Register for users
3	Email	validate the email	send code to the mailbox	Email validation
4	Search	show the search results	find the corresponding blogs in database according to the search consitions	Search the quesitons
5	Post	check title, content, partition and sub partition and display the post	store the related info into database	Post for users
6	Reply	check the content and display the reply	store the related info into database	Reply for users
7	Reset	check the username and password	check if username exists	Reset for users
8	Profile	upload the profile image	write the image file into the local storage	Upload profile
9	Like	update the button status and the amounts of like	update the related info in the database	Like blog
10	Follow	update the button status and the amounts of favors	update the related info in the database	Follow blog

Table 1.3.2.2 - Inverted Index

LOC (lines of code) is showing in the appendix.

2 System Arcgitectual Design by DFD

2.1 System Architecture



In our project, we have a totally 7 entity sets and 5 relationship sets. The table gives a detailed interpretation of these sets.

ENTITY / RELATIONSHIP SETS NAME	DESCRIPTION
user	Stores the username, password, email, profile and etc. of the user's account
group	Stores the name, the description and etc. of the group
sub_group	Stores the name of the group and the name of the sub group
blog_questions	Stores the title, the author, the group type, the sub group type, the content and etc. of the blog
blog_answers	Stores the question id, the father answer id, the content and etc. of the answer
attached file	Stores the URL of the file, the question id, the answer id and etc.
attached picture	Stores the URL of the picture, the question id, the answer id and etc.
user like answer	Stores the user id and the answer id, which means the answer is liked by the user
user like question	Stores the user id and the question id, which means the question is liked by the user
user follow question	Stores the user id and the question id, which means the question is followed by the user
user follow group	Stores the user id and the group name, which means the group is followed by the user
user view question	Stores the user id, the question id and the viewing times

The Relational Schemas:

SCHEMAS
user(id, email, username, password, photo, major, grade, create_time, update_time)
group(id, group_name, description, create_time, update_time, amount_of_follows)
sub_group(id, group_name, sub_group_name)
blog_questions(id, title, author_id, group_type, sub_group_type, content, code, lang, content_format, like, follow, hot, create_time, update_time, views)
blog_answers(id, question_id, father_answer_id, content, code, lang, content_format, like, author_id, create_time)
attached_file(id, url, corresponding_question, corresponding_answer, create_time)
attached_picture(id, url, question, answer, group_name, create_time)

Below list the foreign key referencing:

- "group_name" in sub_group refers to "group_name" in group: each sub group belongs to a group.
- "author_id" in blog_questions refers to "id" in user: each blog belongs to a user.
- "group_type" in blog_questions refers to "group_name" in group: each blog belongs to a group.
- "sub_group_type" in blog_questions refers to "sub_group_name" in sub_group: each blog belongs to a sub group.
- "question_id" in blog_answers refers to "id" in blog_questions: each answer belongs to a blog.
- "author_id" in blog_answers refers to "id" in user: each answer belongs to a user.
- "corresponding_question" in attached_file refers to "id" in blog_questions and "corresponding_answer" in attached_file refers to "id" in blog_answers: each file belongs to a question or an answer.
- "question" in attached_picture refers to "id" in blog_questions, "answer" in attached_picture refers to "id" in blog_answers" and group_name" in attached_picture refers to "group_name" in group: each picture belongs to a question, an answer or a group.

Normalization

In our project, we spare a lot of effort on normalization. At first, we intend to reach the first normal form, so we reconstruct our database. For example, “Table 1” shows our initial group table design with a tuple of sub-group names. Therefore, we split the group table into a group table (Table 2) and a subgroup table (Table 3). In this case, the first normal form is achieved.

ID	GROUP_NAME	SUB_GROUP_NAME	DESCRIPTION	CREATE_TIME	UPDATE_TIME	AMOUNT_OF_FOLLOWS
1	CSC4001	{Assignment1, Assignment2, ...}	Database System	2022-04-30	null	100

Table 1

ID	GROUP_NAME	DESCRIPTION	CREATE_TIME	UPDATE_TIME	AMOUNT_OF_FOLLOWS
1	CSC4001	Database System	2022-04-30	null	100

Table 2

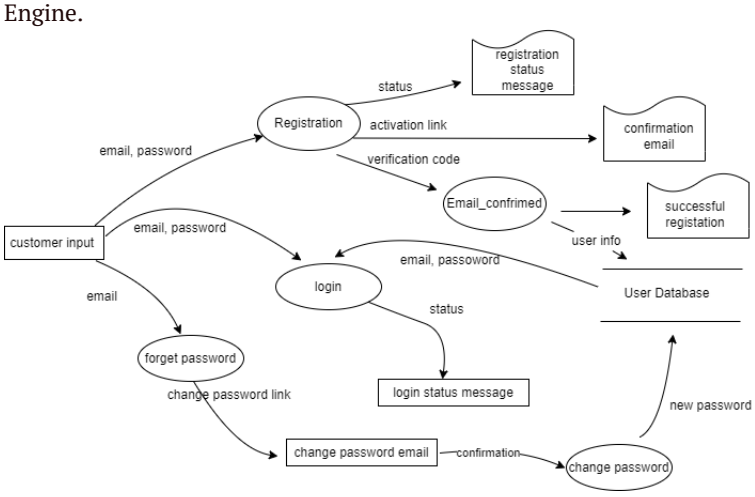
ID	GROUP_NAME	SUB_GROUP_NAME
1	CSC4001	Assignment1
2	CSC4001	Assignment2

Table 3

Based on the first normal form, we also try to reach the second and third normal form. As a result, we create an unique id for each table to enable all nonprime attributes are fully functionally dependent on the primary key (id). Apparently, there does not exist nonprime attributes in our tables transitively dependent on the primary key (id). Therefore, the second normal form and third normal form are also implemented.

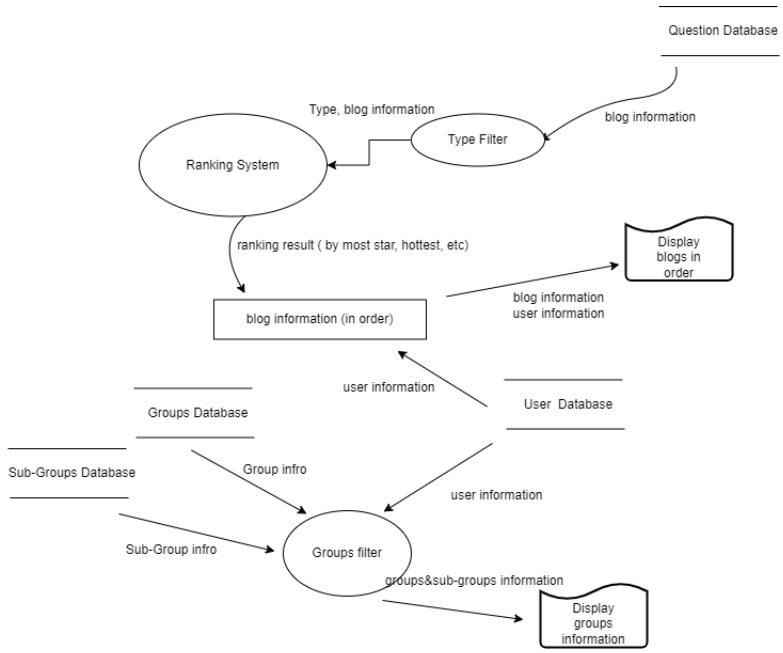
2.2 DFDs

To better illustrate how data flows on our website as well as assist in explaining the mechanism of our work, this report will provide you with some Data Flow Diagrams and several detailed explanations regarding these DFDs. The components of systems we plan to describe below are: Login and Register module, a Main page that delivers blogs, a Question Posting system, and Searching Engine.



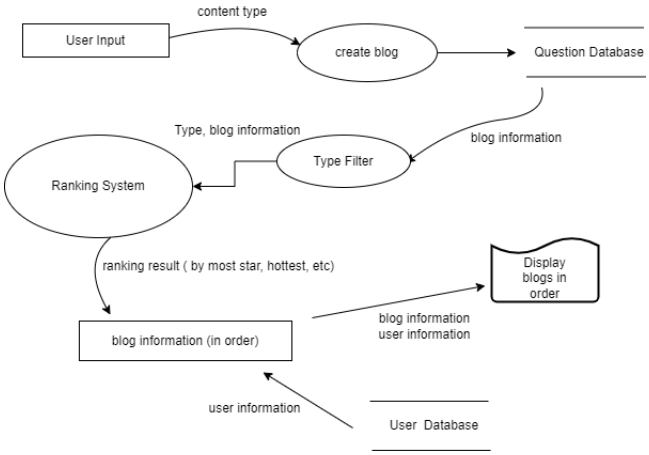
For the Login and register module, our website provides users three main functions:

1. Users can go into the registration part and register a new account by entering his/her email and password. Both backend and frontend will check the registration status and show it on the screen(if the username has already been registered, the status message will display an error.) If what users type are valid, they will be guided to the email confirmation page, and a confirmation email will be sent to the user's email address. The verification code is used to activate the user account, and after the email is confirmed, the corresponding user information will be stored in the user database.
2. If the user has already got an account, he/she can directly enter the email and password to log in. The log-in status would be shown on the screen after backend checking.
3. We empower the user to use email to reset the password if he/she forgets the password. For the safety consideration, there will also be a confirmation code sent by email. Once confirmed, the new password of users would be stored in a database.

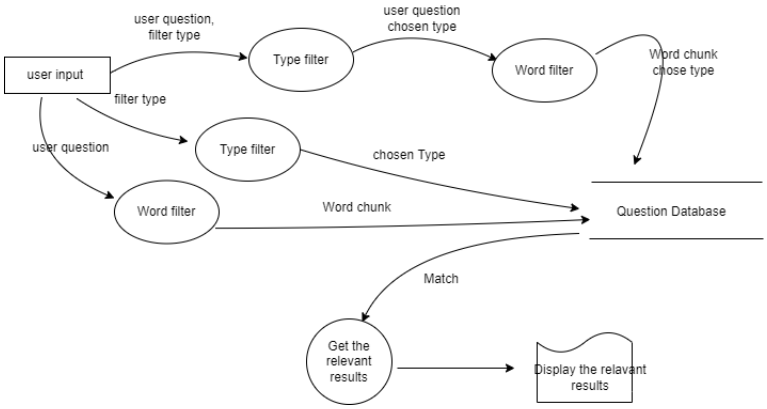


For the MainPage module, two main messages will be shown to the current user: Question(blog) information and Groups & sub-groups information. Of course, there are several other non-crucial information like the number of likes, follows, and views that will also be deliered to users on the main page while not being included in DFD.

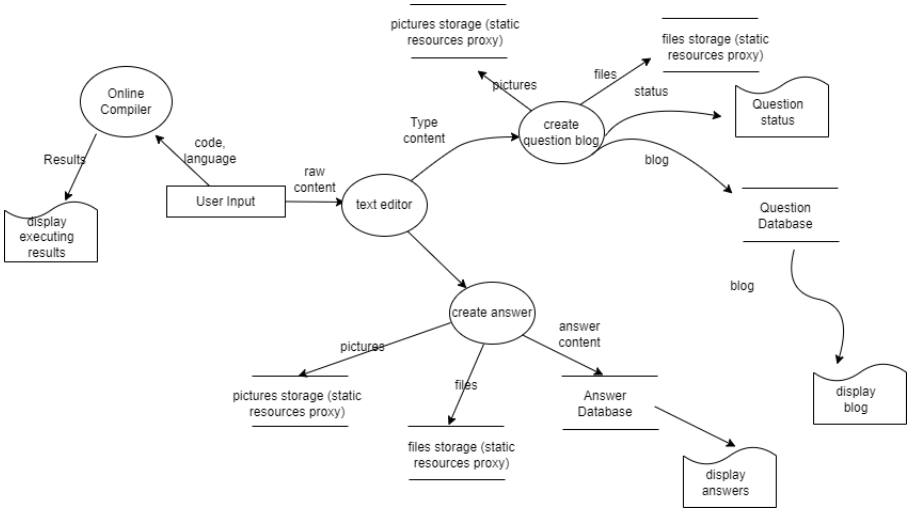
1. For the Questions(blogs), they will be retrieved from the backend and through a ranking system and to be recommended to the current user. Finally, the corresponding blogs will be shown in the frontend.
2. For the group's and sub-groups' information, they will be retrieved from the database. They will first go through a filter to distinguish the groups are followed by users or not. Finally, they will be delivered to user in together.



For the search engine module, the basic data flow is chunking the user input and get the corresponding results in the database according to the similarities. Since the detailed mechanism will be explained in the following part, we don’t introduce it too much here. The above figure is the corresponding DFD.

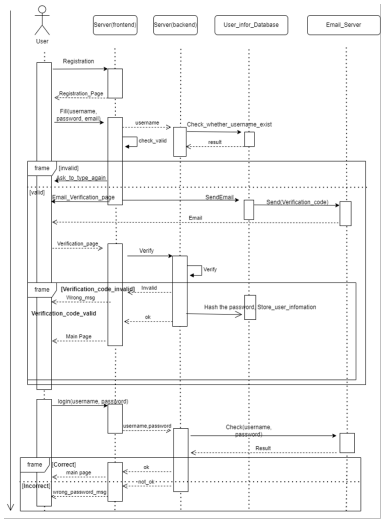


For the upload questions, the content of questions goes through a text editor and becomes pretty(in HTML format). Then, the content will be stored in the Question Database. The pictures and files will be separately stored into the storage file in the cloud server. For creating a new answer under a blog, the data flow is very similar to the one of uploading new questions. Answers will be stored in the anwer database, and files and pictures will be stored in the cloud. The third flow is regarding the online compiling part, the code and language will be sent to the online compiler, and after the execution of the code, the running result will be displayed to users.



3 Detailed Description of Components by UML

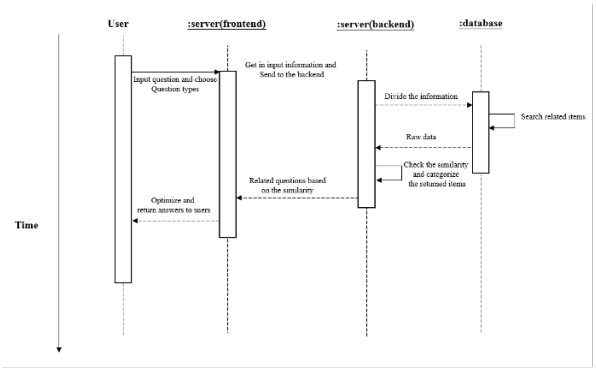
3.1 Component-1: Registration & Login



If it is the first time the user visits our website, then registration is essentially the first step. To make a new count, users must input the username, password, and email address. Based on the information, the backend will check whether the username has been taken or not. If the username is invalid, then the information will be returned to the user in order to help them choose a new valid username. If the username is valid, then an email with a randomized verification code will be sent to the input email address. The user has to type in the exact verification code to finish the verification process. If the code is invalid, the corresponding message will be sent back to the user. If the code is correct, which means the registration is legal, then all the formation about the new user will be formally recorded in the database. The most sensitive information, the password, will be hashed and encoded in the database to protect the users' privacy and keep the website safe. After storing the information, a new account is created. And the user will be redirected to his main page directly without another log-in process since the registration will promise the validation and safety of the user.

If the user already has a valid account, then the registration process can be ignored, but the log-in process is now required. The user has to type in a unique username and password to get into the website. Then the information will be sent to the backend of the server to validate. If neither the username nor the password is valid, the log-in request will be surly rejected, and faint hints will be sent back to the user. Specifically, the hints information goes like "Username or Password Fail", which will not provide detailed information about the error so that the privacy of the user is protected. If the information is valid, then the user will get permission to the main page.

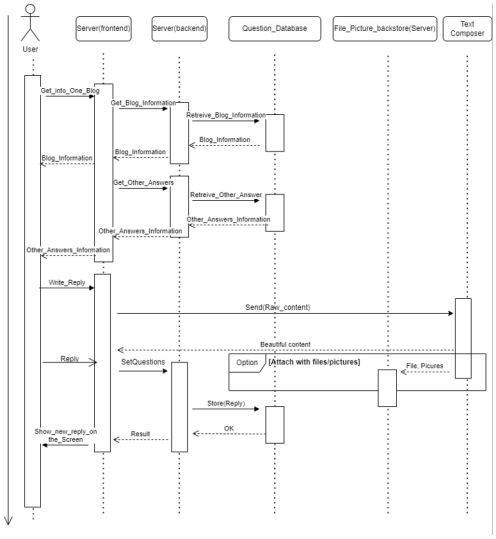
3.2 Component-2: Search Question



The search function is implemented by keyword match rather than string match. Users can search blocks related to a specific topic on the main page. Generally, the frontend will check the input and send it to the backend. The backend will split the content into different vital words based on the typed content. Then based on the derived keywords, the backend will go through the entire database and search for the blocks with the maximum similarity. Then, then related answer blocks will be returned in the order of similarity for the users to choose. For the keyword match, it is used to expand the searching scope so that the users can get more related blocks that they might be interested in.

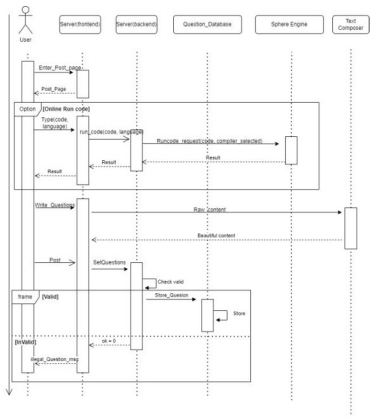
Getting lots of returned blocks is always a good idea. Sometimes, people are more interested in finding what they want in a faster way. Therefore, our search function is implemented together with the scope selection. That is, users can select the topic of the questions to narrow down the search scope. In this way, the blocks returned will be more related and concentrated.

3.3 Component-3: Reply



To reply to a block, users have to get into a block in the first place. After users get into the block, not only the block information will be shown, other answers will be returned if existing as well. All the returned information is provided for the users to better understand the blocks. If users want to offer more information and reply to the block, they should type in their answers, and the answers will be sent as raw content to the text composer. Then the text composer will process the raw content, returning and showing the content in a neat and nice way. Besides, users can also upload the files and pictures together with the answers. The uploaded answers will be stored in a unique area on the server. After the edit of the reply, the reply can be formally uploaded. Answers and their relation to different questions will also be stored in the server's database. At last, the reply will be shown to every visitor to the block.

3.4 Component-4: Online Compiler



The online compiler is on the posting page. Users can enter the posting page at first. If they want to test some codes, they should choose the programming language first. Then, they are free to program on the posting page. After they click the run button, the request will be sent to the sphere engine, and different compilers will be chosen based on which kind of program language the users are using. After that, the code will be compiled in the backend, and the result will be sent back to users.

The process of raising a question is similar to the replying procedure. In the text chat, users can type in the essential information about what they want to post, including the title, content, group type, and sub-group type. Then the information will be optimized by the text composer. Next, the backend will check the validation of the question. If there is nothing wrong, the question will be stored and posted. If not, corresponding hints will be returned to the users for them to correct.

4 User Interface Design

4. 1Description of the User Interface

The interface of our project is inspired by the public forum in China: Zhihu, especially the home page. It mainly consists of the search engine on the top and the different navigation bars below. Each navigation bar represents a unique order and function for all of the blogs in our database. For example, the hottest blog navigation bar represents the most popular blogs among all users, and the non-solved navigation bar represents the blogs that are not answered yet, and whose layout is similar to Zhihu.

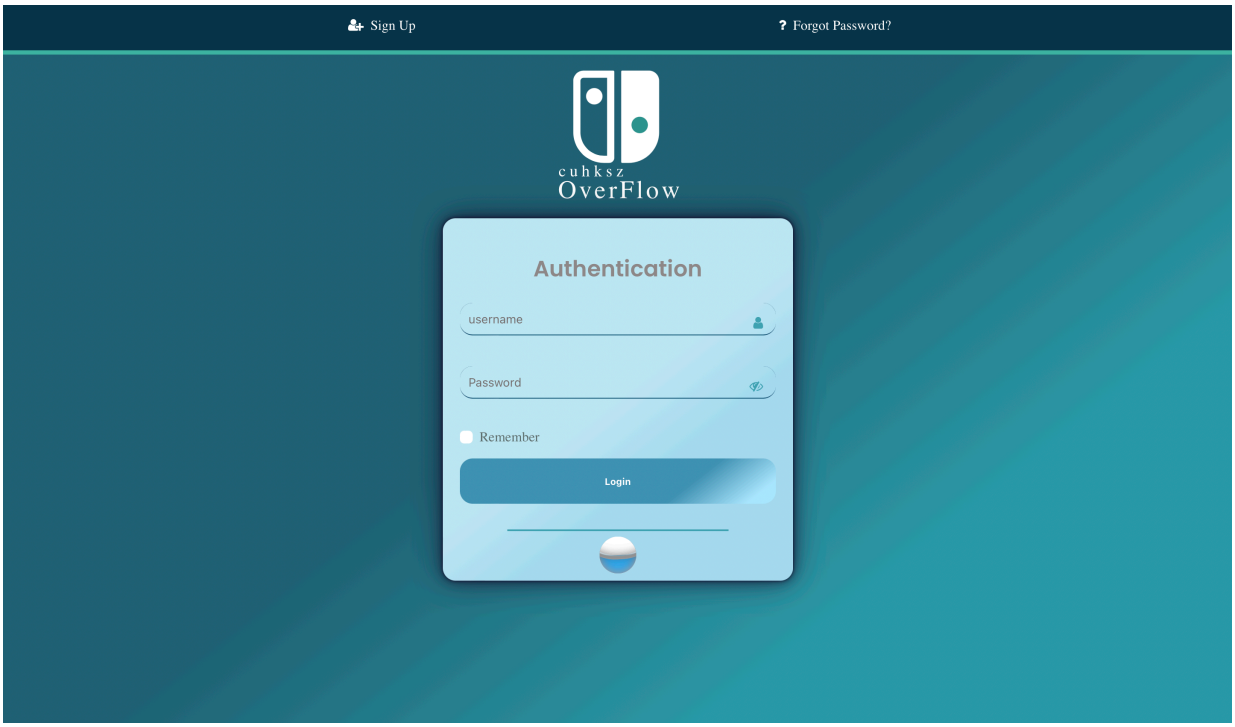
One of the characteristics of our user interface is animation. Our interface integrates many animations in order to make the interface more vivid and more interesting to users. For example, you would notice that there is a dynamic crystal ball waving on the log-in page, as well as the animation text on log-in in page. Also, after you finish the register page, there is also another animation between two different web pages. It is not simply jumping to another webpage directly in a rigid way. To make users more pleased when they use our forum, we also designed a loading animation between the home page and post blog page. In this way, the user would not watch the webpage stuck in a while, or some hacker may use this time to click or type something on the interface to make the webpage crash. Instead, we designed this loading page to protect our program and could provide an animation for users to see to improve their experience feeling.

Another highlighted feature of our user interface is button design. We took an amount of time to design our buttons. Since there are a lot of buttons, typically 15-20 buttons in our program, and there are all different to each other. Each button is well-designed for the user. For example, to alter user to log out, we make the button highlighted when the user hovers this button in case the user will log out by mistake. To warn users to reset username and passoword, we make the button red and a warning way. We have considered those details to make sure they are as user-friendly as possible.

Overall, our UI have is **detailed, well-designed and friendly to user**. The follwing part will demonstrate each function and action.

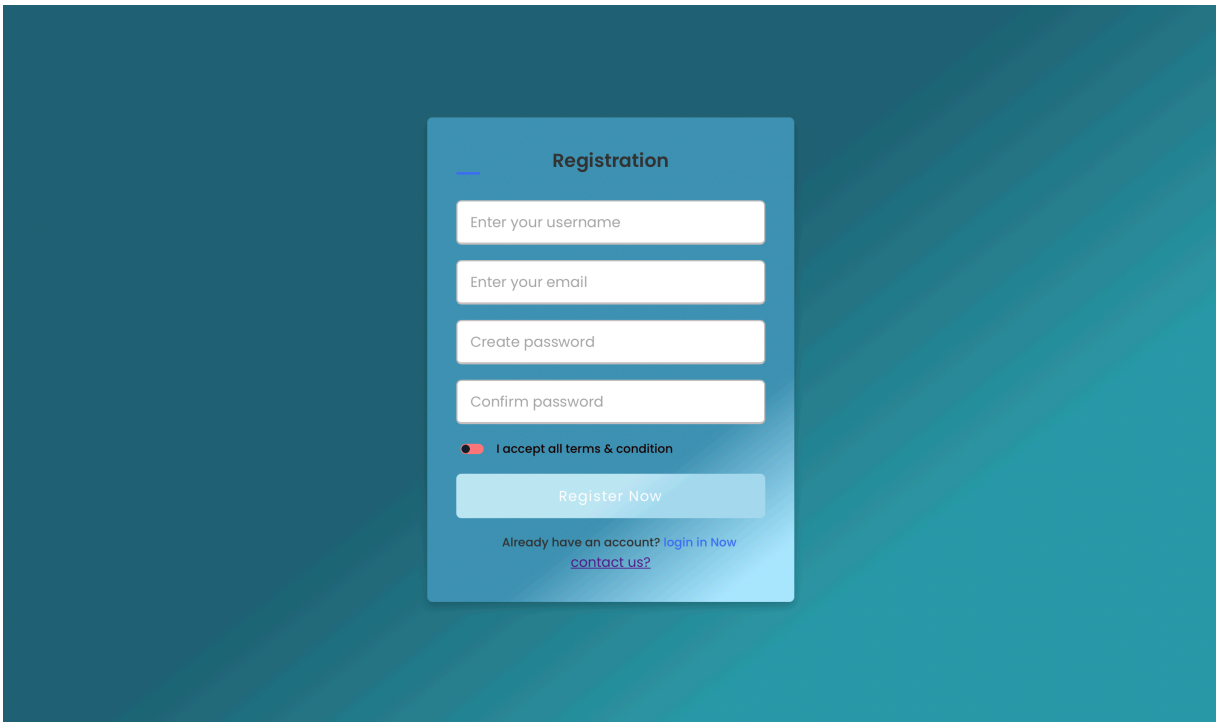
4.2 Object and Actions

- **Login**



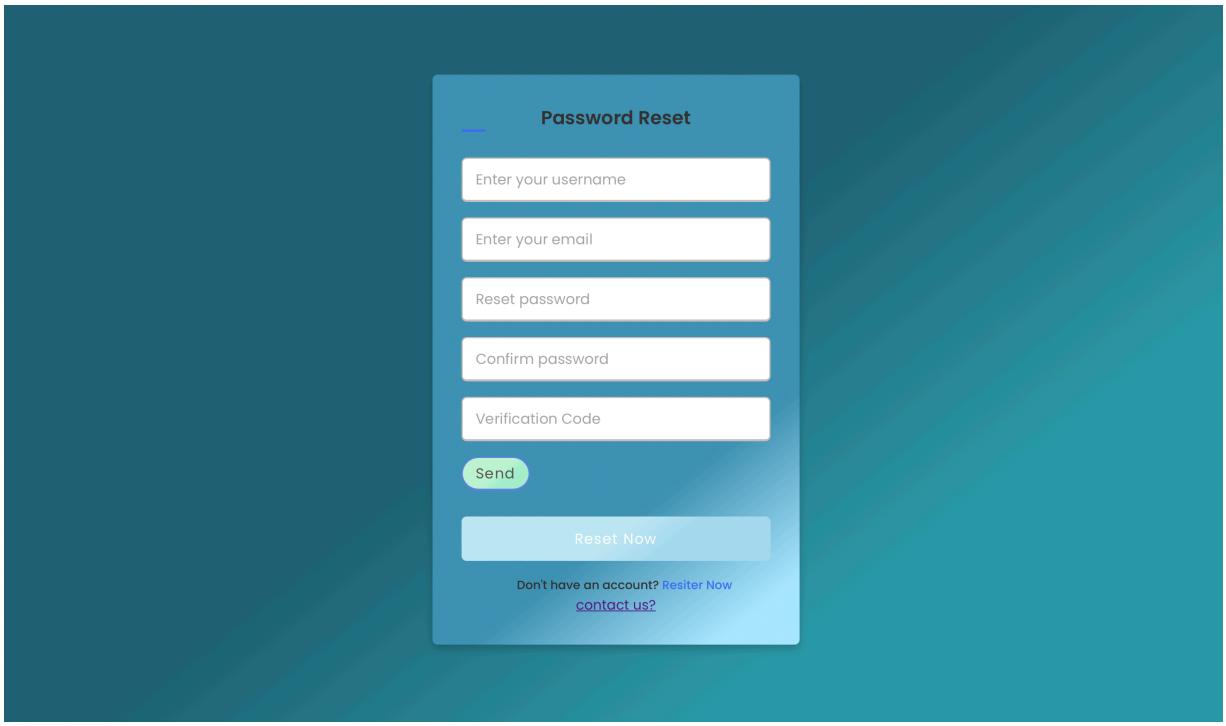
The log-in page is designed for the user directly use their user name and password to get into the home page, users must input the correct username and password in our database in order to log-in. Users could click the eye icon to hide or show the password. They can also click remember to remember their password. The sign-up and forget password buttons are on the top. Users could click those icons to jump to another webpage. The animation text will be shown every time one refreshes the page.

- **Register**



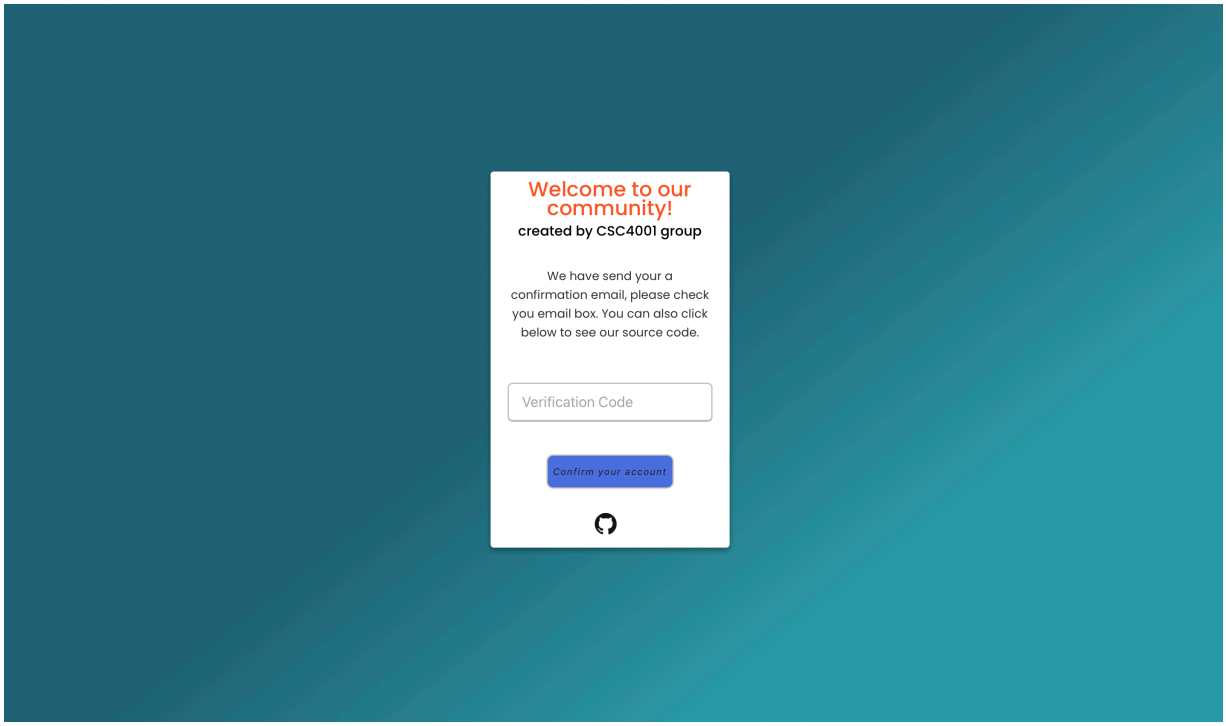
In the registration part, Users are required to input their unique username and their email. They should type their password and confirm it. They should also allow all conditions to follow the rules and be responsible for their words in our forum. Notice that the username is unique if users try to type the username that has been used by others, which is not allowed here. Users could also click the log-in button to jump back to log-in part.

- **Forget Password**



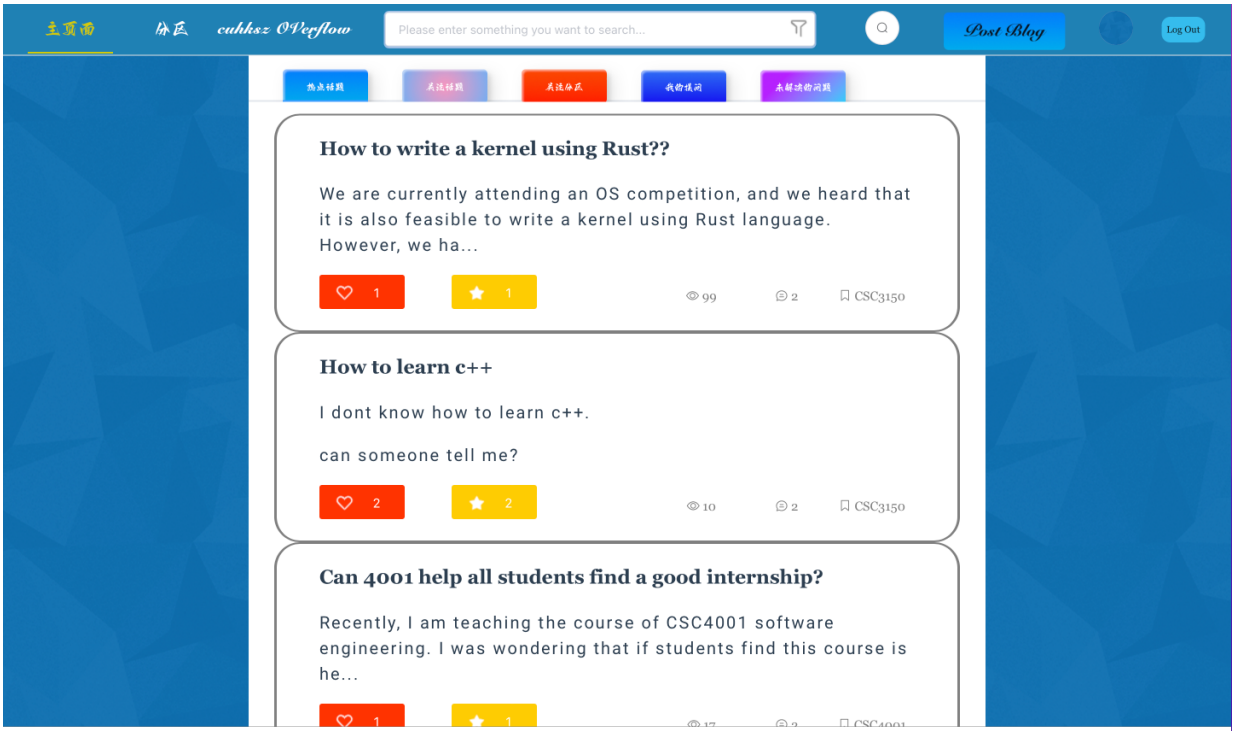
This part is used to find back their password and reset it. Users are required to type their username, email and their new password. Notice that if the username is not correct, this reset will fail. We will send you a verification email, only and if only the verification email is correct, the password will be reset, and users will directly jump to home page.

- **Verification**



After users finish the registration part, they will receive an email from us. We have an animation on this page in order to let users feel comfortable when they are waiting since the Verification email may take a while to send. During this time, Users could click the GitHub icon to see and run our source code. When the email verification code is correct, the users are allowed to go into the home page.

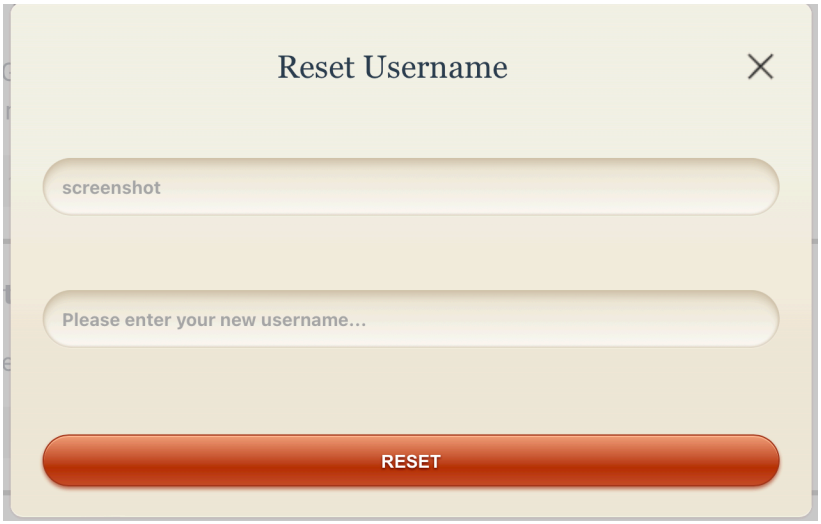
- **Home Page**



As for the home page, The layout of the home page consists of three parts. First is the menu on the top, users could click the home page icon and partition icon to go to a different page. They could also click the search bar to type some content to search. And the button on the right is the post button, users could click the button to jump to the post blog part and click the log out button to log out. They could click the me button to upload their unique profile. We will store their profile on our server. The second part is 5 catalogs, and each catalog contains a different function and order of blogs. In hottest blogs, it mainly contains the blog that is hottest and most popular on our forum. The followed blogs represent the blogs that user click followed button, and the followed partition is the partition that user clicked follow. My blogs will show users the blogs that they have a post on our server. Unsolved

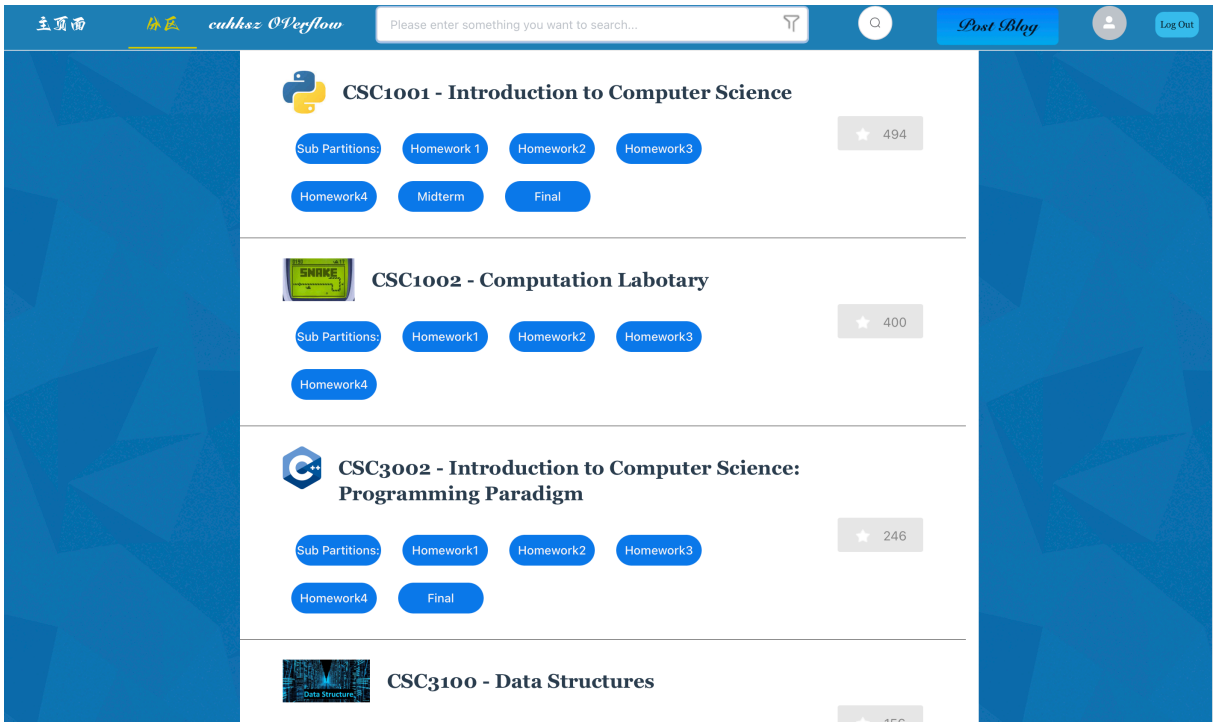
blogs is represented the blogs have no answer, it is designed to encourage user to answer and reply the blogs on the server. The Third part is blogs part, blogs are ordered by some function in catlogs. When users hover each blog, the color will be gray. The tiltle of blogs is bold and the content is normal. each blogs will show users the view times, the answer amount and its partition type. Users also could click the like or follow button. The color will also change after user click it.

- **Reset**



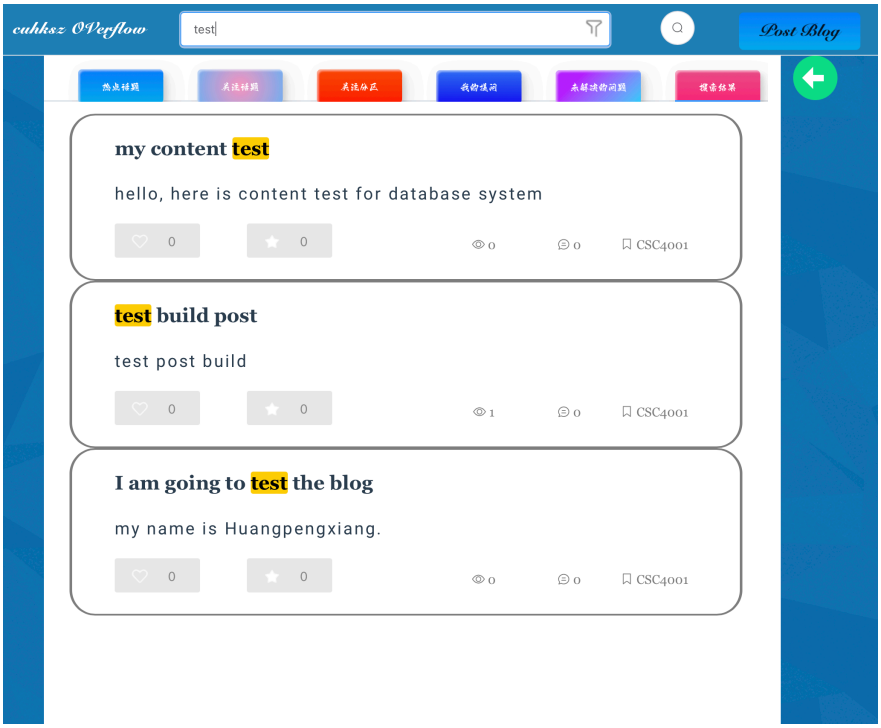
When Users click the Reset Username or Reset password, they can reset their user name or password on the home page. When users reset their password, they are required to log in again.

- **Partition**



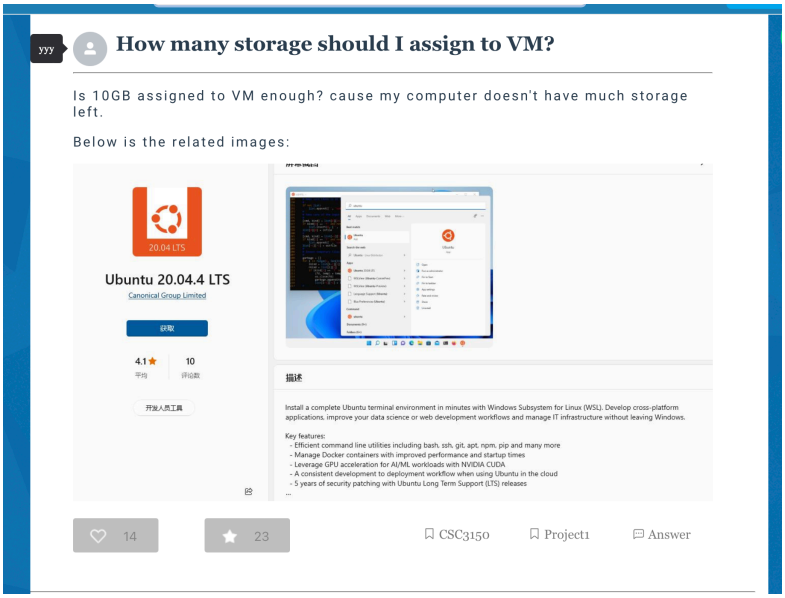
When users click the partition button, they will jump to this part. This part is used for users to find the corresponding blogs in specific partitions. Each partition contains different sub-partitons. Users are allowed to click the button to see all the blogs in this subpartition. So they could find what they want in a very short time. And users are allowed to click the followed button to see their followed partitions then home page, which is convienent for users to find corresponding blogs.

- **Search**



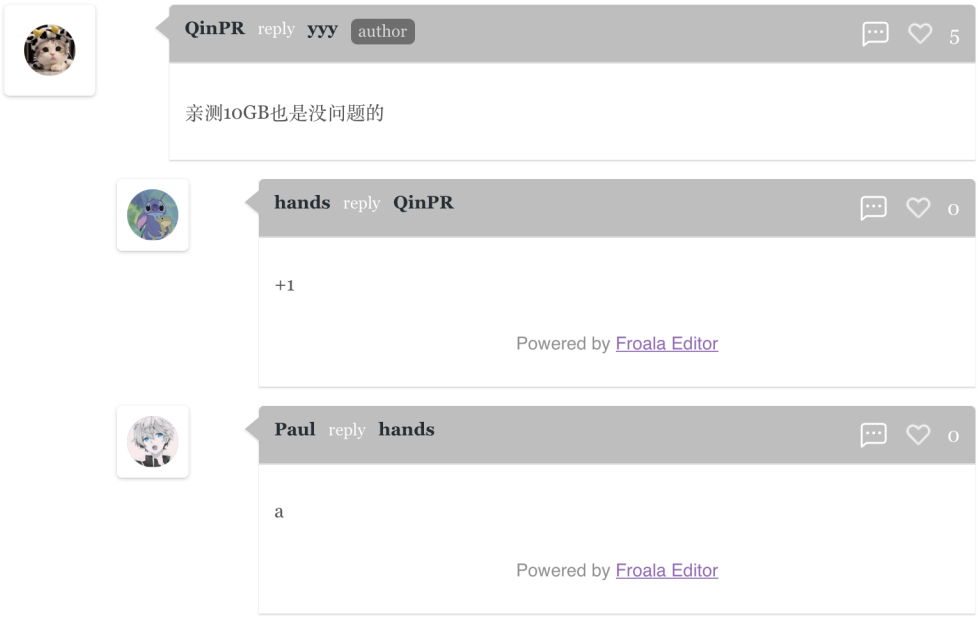
Users could type the keyword in the search bar to search the relevant blogs. They could type anything they want, and we will return the most relevant blogs on the home page in order. The keyword will be highlighted so that user could find the answer very quickly. And users are also allowed to use the filters to narrow down the search scope. They could click the button to choose the search scope. The search scope is based on the partition and subpartition. When User choose the filter, the search scope wont be all blogs, it will return all the content in this scope.

- **Blog**



When users go into the blog page, they could see the blog's partition and its subpartition and click the like or follwed button to support this blog. The title is in bold form, and the content is normal. The file will be shown as a link in the blog, and the picture will be shown directly. Users could click the file link to download the file if the blog contains any. They are also allowed to enlarge the picture or download the picture. When they hover over the user icon, the username who created this blog will be shown. All the text form in the post part will be the same here.

- **Reply**

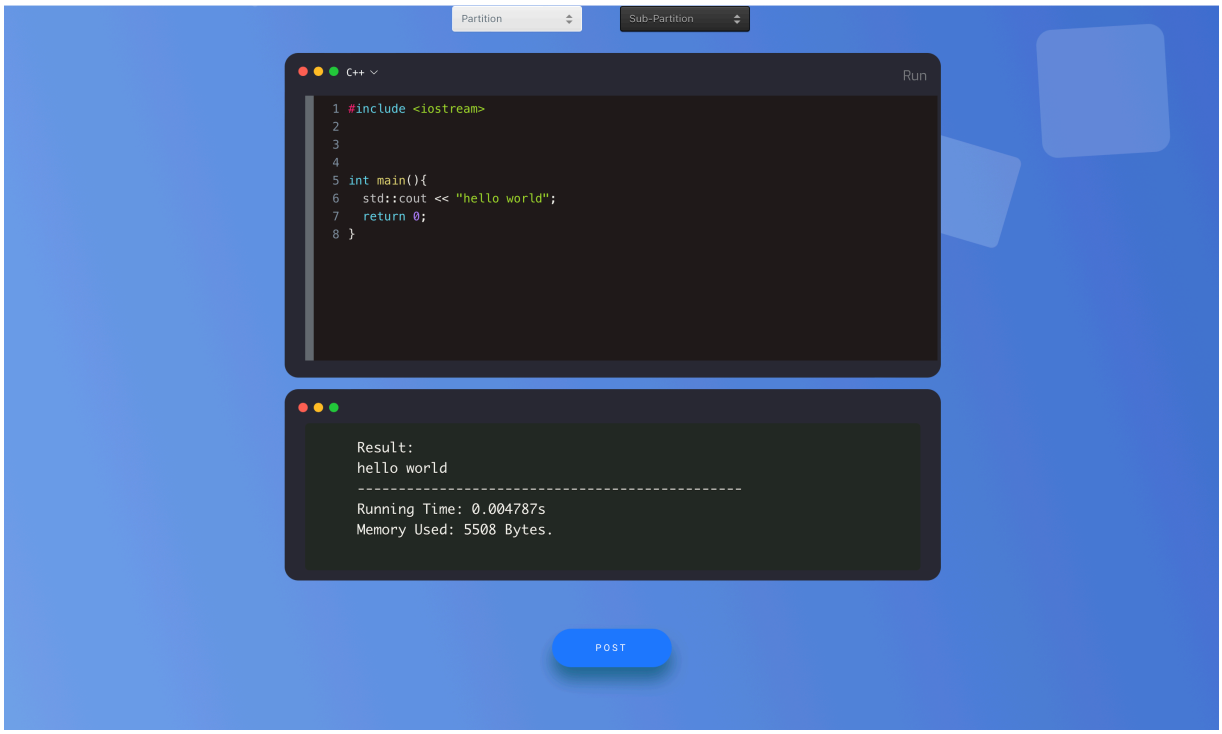


Users could click the Answer button to answer each blog. Notice that the indentation will be different when users answer in a different situation. When they answer the author, they will be classified into the first level. Otherwise, they are at the second level. The first and second levels have different identations to identify the blog answer quickly. Users are also allowed to click the like button to support this answer. The answer blog will be ordered in time.

- **Post**

When users click the post button, they can create their own blogs. They are required to type the blog title and content, partition and their subpartitions in order to classify. The content input will be finished in a rich editor. The rich editor support many forms of input and file uploading. Users could upload their file or just insert a file link. Our server will receive this file and store it on the server. Users could use the rich editor to create the blog forms they prefer. And all the blogs on the home page will be decoded as the same as they posted on this webpage.

- **Compile**



In the post part, users are also allowed to write their code as a supplement material. they could also run the code to see the output. Typically, Users are allowed to write the code without environmental configuration. They could also write code on an iPad or a phone. The online compiler support many languages, including C, C++, Python, Rust... The running time and memory used will be shown on the webpage just as an open Jude system. The code will be highlighted due to different language, different language has a different highlighted method, and they will also be used here to improve user coding feeling.

5 Test

- For the explanation of our test part, it will be divided into 3 sub-sections. In the first section, the introduction of the test-files arrangement will be introduced. In the second section, we will explain the design idea of test files, including what test suites we are leveraging, what functions we are testing, and how we make sure the test suites have good coverage of cases. In the third section, we will report on the testing results.

Test-Part1: Overall arrangement of test files

To ensure our programs' functionality, robustness, and generality, we implemented the Unit tests, component tests, and system tests based on BlackBox testing. The arrangement of test files is showing below:

- For Unit Testing, it includes 19 test files based on blackbox testing:

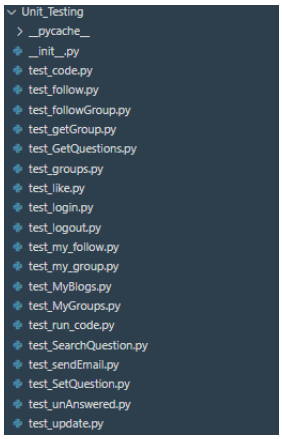


Figure 1: the files conduct Unit Tests.

  - As it is shown above, we leverage these 19 unit-test files to ensure the correctness of the basic APIs and

functions we defined in the project. The brief explanations of each of these files will be shown in the later part.

- For Component Testing, it includes 6 test files based on blackbox testing:

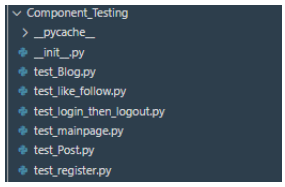


Figure 2: the files conduct Component Tests.

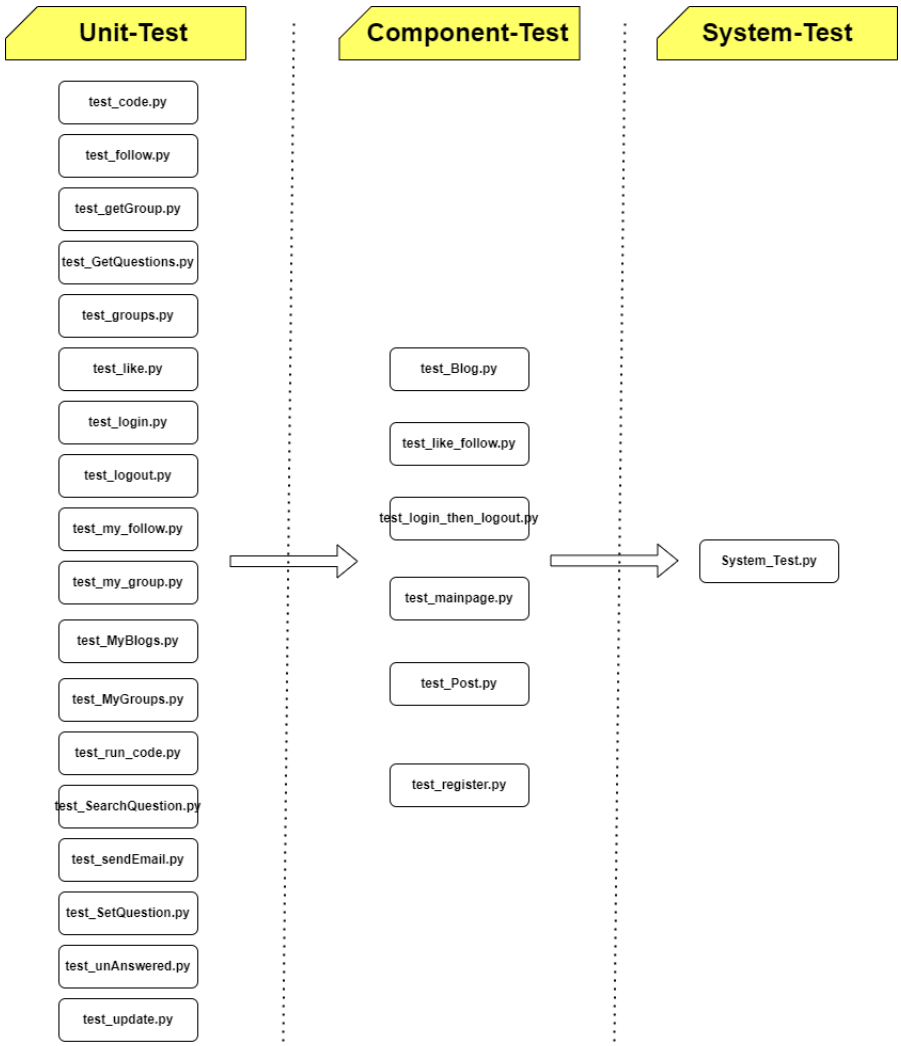
  - We leverage 6 component test files to conduct the component testing. In the component testing, we mainly test the significant functions we provide to users on our website, such as click-on-like buttons, a log-in system, etc.

- For the System Testing, it includes one overall file merged by the component tests:

Figure 3: the file conducts System Test.

  - We merge all the component tests and add some links to these component test suites to form overall testing over the system. The details will be explained in the next part.

- Based on the structure of Unit-Test, Component-Test, and System-Test as shown above, we get the whole structure of our test part:



Test-Part2: Design ideas of test files

In this part, we will explain the corresponding function that each file tests, as well as the design idea of each test file briefly.

For the unit-test:

1. `test_code.py`

   This file tests the `views.encode()` function.

   `views.encode()` is expected to randomly generate a code with length of 6. This test file would call this function 100 times and check whether the generated code gets repeated. If there are repeated generated codes in 100 times, it means the randomness is not strong enough. Thus, this file would through "FAILED" as a result.

2. `test_follow.py`

   The function of this file testing is `views.follow(request)`, which is expected to perform the function of empowering follow another user by clicking the follow-button we provided in the frontend. There are 3 test cases involved in this file: 1. the case with normal valid input. 2. the case with an invalid input: input with GET request, while the function is designed to only accept POST request. 3. request with a non-exist username.

   For the case tests, the normal valid input, this test file will check the status of the response and the return value. For the case tests the invalid input, this test file will check whether the error message is accordant with what we expected it to throw.

3. `test_followGroup().py`

   This file tests the views.followGroup(), which is expected to empower the user to follow one specific group by clicking the button we provided. There are 4 test cases involved: 1. normal and valid input of following a group. 2. request with GET request. 3. request with an invalid user's name 4. follow a non-existing group's name.

   As the same idea that we illustate in `test_follow.py`, we check the status and return value of the normal valid input. And compare the error message to what we expect to receive for the 3 invalid inputs.

4. `test_getGroup.py`

   This test file tests the `views.getGroup()` funciton, which is expected to return all the related blogs of one group (such as all the blogs related to CSC4001-Project). There are one test case for the valid and normal input and two test cases for the invalid cases. Specially, for the test for normal valid input, it should also check the contents that return to ensure it doesn't return an empty value for one of the return values.

5. `test_GetQuestion.py`

   This test file tests the views.GetQuestions(request) function, which is designed to return the contents, titles, amount of likes, follows, etc., when users enter into a specific blog. There is one test case for the normal input and two test cases for the boundary testing.

6. `test_groups.py`

   This test file tests the `views.groups(request)` function, which is expected to return all the groups, as well as each's description, URL of the picture, number of follows, and whether the current user follows in main-page. There is one normal case to check whether it returns the groups' information correctly and two invalid cases to check whether this function is robust enough when the user tries some invalid inputs.

7. `test_like.py`

   This test file tests `views.like(request)` . Since this function is expected to add the amount of like when the user who never liked this blog/answers clicks the like-botton, and cancels the like when the user already likes it, there are two cases for normal inputs: 1. like one blog/answer, 2. cancel like. There are two boundary test cases included as well.

8. `test_login.py`

   This test file test the `views.login()`. Since this function is the function that requires the most robusty, we consider some special cases that main encounter when using this function.

   First, we test log-in with a username and correct password. In the second case, we test log-in with a username and the wrong password. In the third case, we test log-in with sessions (because we keep sessions valid for 1 day if they already log-in and the user can avoid log-in again within 1 day). In the fourth case, we test log-in with valid cookies.

9. `test_logout.py`

   This file tests `views.logout()`, which is expected to enable the user to quit the system and direct him/her to the log-in page. One normal test case and two boundaries (also invalid) test cases are included.

10. `test_my_follow.py`
    This file tests `views.my_follow()` , which functions returning all the blog that the current user is following. We would check whether it would return the correct number of blogs that the user is following, as well as whether the contents are valid. There are also two test cases for invalid request method and non-existed user name.
11. `test_my_group.py`

    This file tests `views.my_group()` , which is expected to return groups that the current user is following. There is one normal input for testing and two invalid and rare inputs for testing as well.
12. `test_MyBlogs()`

    This file tests `views.MyBlogs()` , which functions by returning all the blogs posted by the current user. We check whether the returned blogs are actually posted by the current user to ensure that the current user get the correct result.
13. `test_MyGroups()`

    This file tests `views.MyGroups()`, which is a function inherits from `views.groups()`. it includes totally 3 cases in this file.
14. `test_run_code.py`

    This file tests the function of execution of Online compiler we leverage. For the normal and requent input, we test source code as `print("hello world")` and language like Python. As well as some invalid input like introducing buggy code to the source code, or use a non-consistent language, or directly not specifying the language of the source code.
15. `test_Search_Question.py`

    This test file tests the functionality of our search engine. We introduce 5 test cases in this file. For example, search the question in Chinese with English. Or search the questions within a specific scope. We compare the search result with what we expect to get to evaluate whether these functions work normally.
16. `test_sendEmail.py`

    This file test the Email-sending function, which is the basic function when the user registers. It includes 3 test cases to check whether the email has been successfully sent out and also the ability to handle the invalid email address.
17. `test_SetQuestion.py`
    This file test the basic function `views.setQuestion()` which empower the user to post a new question with title, content, source code, language, and corresponding group. Except for one test case where the user posts a complete question. We also introduce another 4 test cases to test the situation when user posts a question without tile, or without a specific group, or without any content, or without his/her name included. We check whether this function is able to handle all of these possible situations.
18. `test_unAnswered.py`

    This file test the function `views.unAnswered()` , which is expected to return all the Blogs with nobody answering it. We specifically check whether the returning blogs actually got no one answer. And also, two boundary tests are included.
19. `test_update.py`

    This file specially test the functions `views.update()`, which is in charge of enabling users to change the password and username. We check whether the user can successfully update the password or user name by checking into the database after calling this function. And there are 3 test cases included in this unit-test file.

After explaining the design of 19 unit-test files, we continue to briefly introduce the idea that we utilize to design the Component testing. There are 6 files to perform the component testing and together test the correctness of blog posting, like-follow component, login-logout component, main page component, blog delivering component, and register component:

1. `test_register.py`

   In this component testing file, we test all the functions during the process of registering a new account. It includes email sending, user name-checking, and database updating. We include three test cases to verify its correctness.
2. `test_login_then_logout.py`

   In this component testing file, we consider log-in and logout functions to ensure there will be nothing wrong when the user logs in first, then log out of the system. Also, when designing the test cases, we specially consider the situations with cookies and sessions to ensure they function well in this component.
3. `test_Blog.py`

   This component testing file tests the correct function of delivering a blog to user when he/she clicks on the blog.

4. `test_Post.py`

   We introduce 5 test cases here to test all the functions exposed to users when posting a question, such as running the code and storing the whole question(title, contents, corresponding groups, source code...) to the database. Among the 5 test cases, there is 1 case with normal and valid inputs and 4 cases covering the special situations.

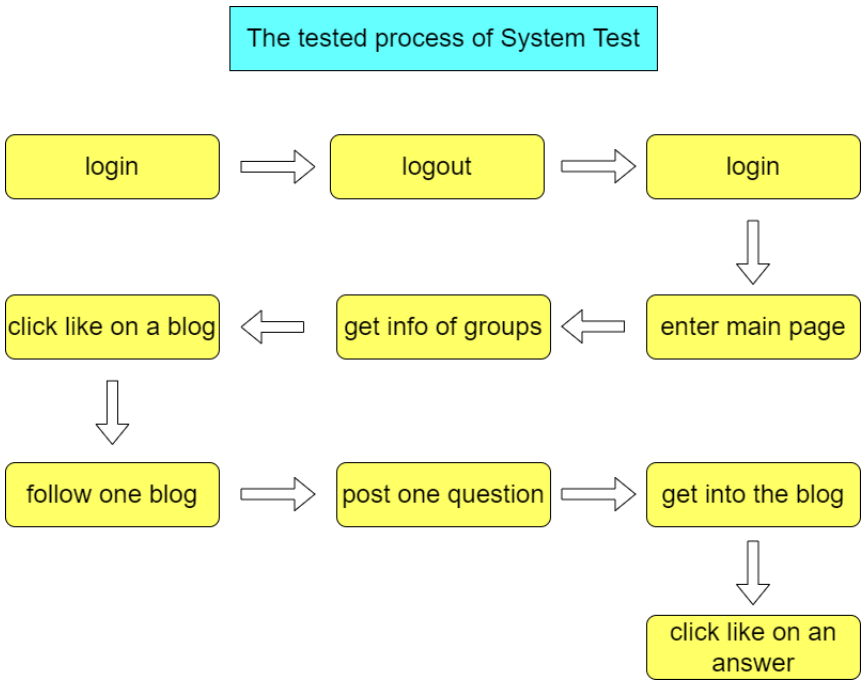
5. `test_mainpage.py`

   This test file tests the basic functions we expose on the main page. Except for checking the status of the response (expected to be 200), it also checks the values that are returned to make sure they are not empty.

6. `test_like_follow.py`

   This component test file tests the like-follow system of our website. It includes 4 test cases to cover the possible situations such as the user clicking likes on a blog/answer then following it, or clicking on like/follow on a blog/question twice.

After introducing the component testing, we link all the components together to form a whole journal of using our website in `System_Test.py` . The whole process of using the website as shown below is tested:



Test-Part3: Testing results

We test our programs using the Unit testing, Component testing, and System testing files we explain above. The results returned show that our programs pass all of the testing cases except the API for the searching question, as shown in the figure below. Then, we debug the API for searching questions and finally get all the test cases passed.

6 Lessons Learned

- Team work

During the whole process, our group enjoys great teamwork, and the teamwork gives us a valuable experience of working and developing the software within a group. Our group started thinking of the main idea of our website in February and started coding at the beginning at the start of March. When thinking of the main idea of our software, our groups discuss the project specification and main functions together for times. During the process, we learn the lesson about the values of teamwork - combine the wisdom of all

the members together and give out a more rational, innovative, and feasible idea that all the members have the will to work on it. We learn the second lesson from teamwork during the coding process: a clear and responsible division of work for each member. Before our group starts working, we discuss each member's schedule to complete each week. The responsible division of workload and clear schedule of work smooth the process of developing software together and let all of us focus more on the work. To conclude, we learned a lot from the great teamwork and got an experience about how a crucial role that teamwork plays in the software development.

- Knowledge about the cooperative development of backend and frontend

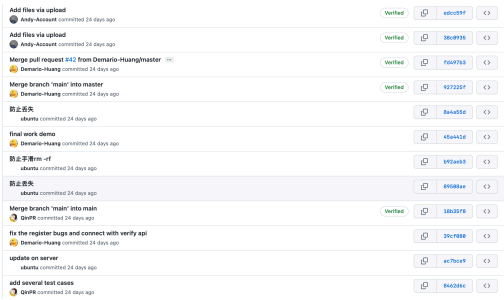
At first, it is not easy for our group's frontend engineers and backend engineers to effectively develop the software in a parallel way because there are many dependencies between frontend and backend. We realized this problem and decided to pursue a way the backend and frontend could be developed together more efficiently. Therefore, we choose to deploy the development environment to the cloud server at the very beginning. In this way, frontend engineers can access the APIs provided by the backend deployed on the cloud and, therefore, avoid the time-comsuming process of merging code of frontend and backend during the development process. On the other hand, backend engineers can also access the static resources of the frontend from the cloud server during development. In addition, every member can access the cloud server and see the program logs when bugs emerge. This development model brings several conveniences to us and provides a lesson of cooperative development of backend and frontend.

- Technical Skill learning

From this project, we have learned how to use Vue, a frontend frame, to build our webpage and connect to the backend to transfer the data. For the backend, we have come to master the Django framework, the cloud server application, and the database manipulation through the project development. Also, we have implemented many web actions to make our webpage fancier and more applied to attract users. In fact, there are amount of webpage actions in our frontend like log-in, search, post, and delete. Those actions require many buttons and data transportation, which may cause many bugs. We took enormous time to design our page and also debug those web actions. It is tedious work when the web actions increase. Most Importantly, both the members in the frontend and backend follow the strict rule of an API function, which maintains a development path for us. We also made the development timeline and divided those tasks into different pieces, and we followed our schedule and had a meeting to cover the progress each week. It guarantees to finish the vast project in time. The cohesive teamwork and cooperation ensured we could finish our project successfully step by step.

Below is the exmple of our api function and github commit log during the developing progress, which demonstrate the team work.

- SetQuestion/
- 将blog存入数据库
- 申请时需要附带：
  - title
  - author_name
  - group_type
  - sub_group_type
  - content
  - code
  - lang



7 Conclusion

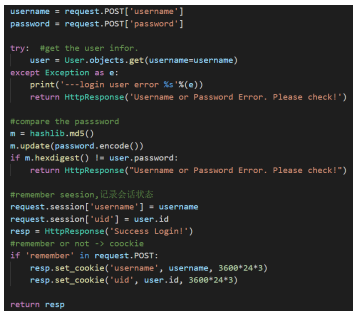
In Conclusion, to help our programmers in CUHKSZ improve their learning and problem-solving efficiency, we created the CUHKSZ-Overflow. It also offers a social media for students to share their idea, exchange their thoughts and solve their learning issues. Meanwhile, We also provide the online compiler for all of the users, which provide users to write, compile, and run their code on our web server so that they will no longer configure the environment on their localhost, and they can write their code on their iPad. More importantly, we also provide a powerful search engine, enabling users to find the relevant answers relatively quickly.

For the E-R diagram and database design, we maintain all tables in our database in the 3NF form to avoid redundancy. We also design the B tree searhing method to increase our query speed to provide users with a faster and more fluence searching experience. For the User Interface design, we implemented enormous fancy actions like a rich text editor and code highlighted editor for users. Moreover, users are allowed to upload their pictures or file on our server by posting blogs. Those files will be permanently saved on the website unless the user deletes those blogs. We also designed the blog catalog page for users, users could use those catalogs to find the corresponding answers. For the backend and server part, we use nigix as an agency for our server, and we also apply the domain named http://www.cuhksz-stackoverflow.cn so that users can directly access our website without building the environment. For the test part, we generate completed and automatic test code, which includes unit test and component test, and all the tests have been successfully passed, which means the website is well-rounded and completed without bugs.

Overall, The project, CUHKSZ-Overflow, is an efficient tool for CUHKSZ programmers to solve and exchange their technical problems and has been proved to play a role in an online web forum, which could be applied in real life to help many programmers handle their course project or other technical issues. It also has a promising future and could be improved as a more functional and attractive online forum.

8 Appendix

LOC in Project Statistic:



```
username = request.POST['username']
password = request.POST['password']

try:  #get the user infor.
    user = User.objects.get(username=username)
except Exception as e:
    print('---login user error %s'%(e))
    return HttpResponse('Username or Password Error. Please check!')

#compare the passsword
m = hashlib.md5()
m.update(password.encode())
if m.hexdigest() != user.password:
    return HttpResponse("Username or Password Error. Please check!")

#remember seesion,记录会话状态
request.session['username'] = username
request.session['uid'] = user.id
resp = HttpResponse('Success Login!')
#remember or not -> coockie
if 'remember' in request.POST:
    resp.set_cookie('username', username, 3600*24*3)
    resp.set_cookie('uid', user.id, 3600*24*3)

return resp
```

Module_1 backend

```
// used for user submit the login form to server
submit () {
  if (this.username === '' || this.password === '') {
    this.$message.error('please input your username and password!')
  } else {
    let sendData = {
      username: this.username,
      password: this.password
    }
    axios({
      method: 'POST',
      url: 'http://175.178.34.84/login/',
      data: Qs.stringify(sendData)
    }).then((response) => {
      if (response.data === 'Success Login!') {
        sessionStorage.setItem('isLogin', true)
        this.$message.success('Welcome!')
        this.$router.push({
          path: '/home/:username',
          name: 'home',
          params: {
            username: this.username
          }
        })
      } else {
        this.$message.error(response.data)
      }
    })
  }
}
```

Module_1 frontend

```
def register(request):
    #POST 返回数据
        #1.当前用户名是否可用
        #1.插入数据[暂时明文处理]
    data = {
        'isRegister': 1
    }# control flag

    if request.method == 'POST':
        username = request.POST['username']
        email = request.POST['email']
        password = request.POST['password']

        #check user name. Multi-thread considerration.
        old_users = User.objects.filter(username=username)
        if (old_users):# if exists
            data['isRegister'] = 0
            return HttpResponse(json.dumps(data), content_type='application/json')

        #return successful information
        return HttpResponse(json.dumps(data), content_type='application/json')
    else:
        data['isRegister'] = 0
        return HttpResponse(json.dumps(data), content_type='application/json')
```

Module_2 backend

```
// submit the registration form to backend
submit () {
  var regExp = /[0-9]{0}@link\.cuhk\.edu\.cn/
  if (this.username === '') {
    this.$message.error('please input username')
  } else if (this.email === '' || !regExp.test(this.email)) {
    this.$message.error('please input the correct email')
  } else if (this.password1 === '') {
    this.$message.error('please set your password')
  } else if (this.password2 === '') {
    this.$message.error('please confirm your password')
  } else if (this.password1 !== this.password2) {
    this.$message.error('The passwords are not the same, please check it!')
  } else if (this.checkbox !== true) {
    this.$message.error('Please allow the condition to continue!')
  } else {
    let sendData = {
      username: this.username,
      email: this.email,
      password: this.password1
    }
    console.log(Qs.stringify(sendData))
    var url = 'http://175.178.34.84' + '/register/'
    // pre registration to check whether the username has been taken, do not create user account
    axios({
      method: 'post',
      url: url,
      data: Qs.stringify(sendData)
    }).then((response) => {
      if (response.data.isRegister) {
        // send the email to user account
        axios({
          method: 'post',
          url: 'http://175.178.34.84' + '/sendemail/',
          data: Qs.stringify(sendData)
        }).then((response) => {
          this.veri_code = response.data.code
          // send the verification code to the next component using router
          router.push({
            path: '/design',
            query: {
              username: this.username,
              email: this.email,
              password: this.password1,
              code: this.veri_code
            }
          })
        })
      } else {
        this.$message.error('The username has been registered, please change another one!')
      }
    }).catch(error => {
      this.$message.error('It seems have some errors, please try it again')
      console.log(error)
```

Module_2 frontend

```
def verify(request):
    data = {
        'isRegister': 1
    }# control flag
    if request.method == 'POST':
        username = request.POST['username']
        email = request.POST['email']
        password = request.POST['password']

        ##hash the code
        m = hashlib.md5()
        m.update(password.encode())
        password_m = m.hexdigest()

        try: #Multi-thread consideration:唯一索引，并发写入问题.
        #insert data
            user = User.objects.create(username=username, password=password_m, email=email)
        except Exception as e:
            print('--create user error%s'%(e))
            data['isRegister'] = 0
            return HttpResponse(json.dumps(data), content_type='application/json')

        #免登录一天 session
        request.session['username'] = username
        request.session['uid'] = user.id
        #TODO 修改session储存时间为一天

        return HttpResponse(json.dumps(data), content_type='application/json')
```

Module_3 backend

```
def sendEmail(request):
    #send email heres
    #is href a POST or a GET request?
    if request.method == 'POST':
        try:
            email = request.POST['email']
        except:         # if it is a WSGIRequest, it don't have .POST[""] method, so we have to retrieve it by ourselves
            email = request.META["QUERY_STRING"]
            for i in range(0, len(email)):
                if (email[i] == "="):
                    email = email[i+1:]
                    break
        code_send = encode()
        if (email[9:] != "@link.cuhk.edu.cn"):
            return HttpResponse('Invalid email address!')


        R_list = []
        R_list.append(email)
        mail.send_mail(subject='Register code', message=code_send, from_email='1092298689@qq.com', recipient_list=R_list)

        data = {}
        data["code"] = code_send

        return HttpResponse(json.dumps(data , cls=ComplexEncoder), content_type='application/json')#需要把信息验证码一起用字
```

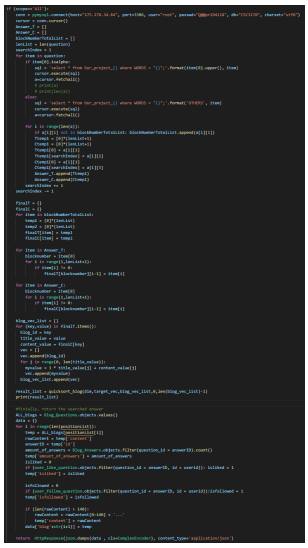
Module_3 backend

```
// send the email to user account
axios({
  method: 'post',
  url: 'http://175.178.34.84' + '/sendEmail/',
  data: Qs.stringify(sendData)
}).then((response) => {
  this.veri_code = response.data.code
  // send the verification code to the next component using router
  router.push({
    path: '/design',
    query: {
      username: this.username,
      email: this.email,
      password: this.password1,
      code: this.veri_code
    }
  })
})
```

Module_3 frontend

```
// confirm the verification code
confirm () {
  let senddata = {
    username: this.username,
    email: this.email,
    password: this.password
  }
  if (this.code !== this.correct_code) {
    this.$message.error('The Verification Code is wrong, please try again')
  } else {
    // send the information to backend, create a user account here
    axios({
      method: 'post',
      url: 'http://175.178.34.84/api/verify',
      data: Qs.stringify(senddata)
    }).then(response => {
      sessionStorage.setItem('isLogin', true)
      this.$message.success(' Welcome to our community !')
      router.push({
        path: '/home/:username',
        name: 'home',
        params: {
          username: this.username
        }
      })
    }).catch(error => {
      this.$message.error('It seems like Something wrong here, please register again.')
      router.push('/register')
      console.log(error)
    })
  }
}
```

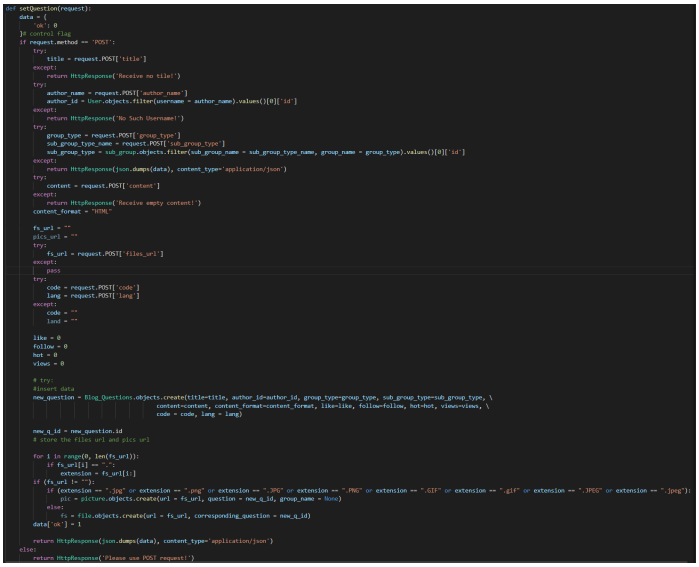
Module_3 frontend



Module_4 backend

```
// User click to get the search results
search () {
  if (this.searchContent === '') {
    this.$message.error('Please type something you want to search!')
    return
  }
  if (this.srPage.length) {
    this.srPage.pop()
    this.activeTab = 'first'
  }
  this.srPage.push({
    label: '搜索结果',
    name: 'sixth'
  })
  let sendData = {
    scope: this.searchCondition,
    content: this.searchContent,
    username: this.username
  }
  axios({
    method: 'POST',
    url: 'http://175.178.34.84/search',
    data: Qs.stringify(sendData)
  }).then((response) => {
    this.srBlogs = response.data
    this.activeTab = 'sixth'
    this.inSearch = true
  })
  if (this.index === 'Partitions') {
    this.$message('Please go to the Main page to see the results!')
  }
},
```

Module_4 frontend



```python
def setQuestion(request):
    data = {
        'ok': 0
    }# control flag
    if request.method == 'POST':
        try:
            title = request.POST['title']
        except:
            return HttpResponse('Receive no title!')
        try:
            author_name = request.POST['author_name']
            author_id = User.objects.filter(username = author_name).values()[0]['id']
        except:
            return HttpResponse('No Such Username!')
        try:
            group_type = request.POST['group_type']
            sub_group_type_name = request.POST['sub_group_type']
            sub_group_type = sub_group.objects.filter(sub_group_name = sub_group_type_name, group_name = group_type).values()[0]['id']
        except:
            return HttpResponse(json.dumps(data), content_type='application/json')
        try:
            content = request.POST['content']
        except:
            return HttpResponse('Receive empty content!')
        content_format = "HTML"

        fs_url = ""
        pics_url = ""
        try:
            fs_url = request.POST['files_url']
        except:
            pass
        try:
            code = request.POST['code']
            lang = request.POST['lang']
        except:
            code = ""
            land = ""

        like = 0
        follow = 0
        hot = 0
        views = 0

        # try:
        #insert data
        new_question = Blog_Questions.objects.create(title=title, author_id=author_id, group_type=group_type, sub_group_type=sub_group_type, \
                                                     content=content, content_format=content_format, like=like, follow=follow, hot=hot, views=views, \
                                                     code = code, lang = lang)


        new_q_id = new_question.id
        # store the files url and pics url

        for i in range(0, len(fs_url)):
            if fs_url[i] == ".":
                extension = fs_url[i:]
        if (fs_url != ""):
            if (extension == ".jpg" or extension == ".png" or extension == ".JPG" or extension == ".PNG" or extension == ".GIF" or extension == ".gif" or extension == ".JPEG" or extension == ".jpeg"):
                pic = picture.objects.create(url = fs_url, question = new_q_id, group_name = None)
            else:
                fs = file.objects.create(url = fs_url, corresponding_question = new_q_id)
        data['ok'] = 1

        return HttpResponse(json.dumps(data), content_type='application/json')
    else:
        return HttpResponse('Please use POST request!')
```

Module_5 backend

```javascript
// submit all the blog content to server
submit () {
  // console.log(sessionStorage.getItem('link'))
  this.link = sessionStorage.getItem('link')
  this.isposting = true
  console.log(this.blogtext)
  var el = document.createElement("div")
  el.innerHTML = this.blogtext
  if (this.title === '') {
    this.$message.error('Please conclude your title!')
  } else if (this.partition === '') {
    this.$message.error('Please choose your partition!')
  } else if (this.subpartition === '') {
    this.$message.error('Please chose your sub-partition!')
  } else if (this.blogtext === '') {
    this.$message.error('Please write your blog!')
  } else {
    var mysendcode = this.code
    if (mysendcode === '// include any code you want to here') {
      mysendcode = ''
    }
    let senddata = {
      title: this.title,
      group_type: this.partition,
      sub_group_type: this.subpartition,
      code: mysendcode,
      content: this.blogtext,
      author_name: this.username,
      lang: this.language,
      files_url: this.link,
      stringText: el.innerText
    }
    axios({
      method: 'post',
      url: 'http://175.178.34.84/SetQuestion',
      data: Qs.stringify(senddata)
    }).then((response) => {
      this.isposting = false
      if (response.data.ok === 0) {
        this.$message.error('can not found the corresponding subpartition, please try again!')
      } else {
        this.$message.success('Post Successfully!')
        this.$router.go(-1)
        console.log(response)
      }
    }).catch((error) => {
      this.$message.error('Post Failed, try again!')
      console.log(error)
    })
  }
}
```

Module_5 frontend

```python
def Reply(request):
    data = {}
    data["ok"] = 0
    if request.method == "POST":
        username = request.POST["username"]

        # store the user_id instead of the username
        userid = User.objects.filter(username = username).values()[0]["id"]

        question_id = request.POST["question_id"]
        if (question_id == ""):
            question_id = None
        else:
            question_id = int(question_id)

        father_answer_id = request.POST["father_answer_id"]
        if (father_answer_id == ""):
            father_answer_id = None
        else:
            father_answer_id = int(father_answer_id)

        try:
            fs_url = request.POST['files_url']
            pics_url = request.POST['pics_url']
        except:
            fs_url = ""
            pics_url = ""

        content = request.POST["content"]
        content_format = "HTML"

        Answer = Blog_Answers.objects.create(question_id=question_id, father_answer_id=father_answer_id, content=content, content_format = content_format, like
        answer_id = Answer.id

        if (fs_url != ""):
            fs = file.objects.create(url = fs_url, corresponding_question = question_id, corresponding_answer = answer_id)
        if (pics_url != ""):
            pic = picture.objects.create(url = pics_url, question_id = question_id, answer_id = answer_id)

        # If the reply has pictures and files, upload them
        data["ok"] = 1
    else:
        data['ok'] = 0

    return HttpResponse(json.dumps(data , cls=ComplexEncoder), content_type='application/json')
```

Module_6 backend

```
// user submit the reply
submitPost () {
  if (this.answerText === '') {
    this.$message.error('Please write your answer!')
  } else {
    let sendData = {
      username: this.username,
      question_id: this.blog.id,
      father_answer_id: this.father_answer_id,
      content: this.answerText
    }
    axios({
      method: 'POST',
      url: 'http://175.178.34.84/api/Reply',
      data: Qs.stringify(sendData)
    }).then((response) => {
      if (response.data.ok) {
        this.$router.replace({
          path: '/blank',
          query: {
            question_id: this.blog.id,
            username: this.username,
            searchCondition: this.searchCondition,
            searchContent: this.searchContent,
            inSearch: this.inSearch
          }
        })
      }
    })
  }
},
```

Module_6 frontend

```
def update(request):

    data = {    #api detail.
        'isRegister': 1
    }# control flag

    if request.method == 'POST':
        _type = request.POST['type']
        newVal = request.POST['newVal']
        oldName = request.POST['username']

        if _type == 'Reset Username':
            #更新用户名
            old_users = User.objects.filter(username=newVal)
            if (old_users):# if exists
                data['isRegister'] = 0  #need to adjust
                return HttpResponse('UserName has been taken')
            else:   #valid username
                user = User.objects.get(username=oldName)
                user.username = newVal
                user.save()
                return HttpResponse('UserName Reset successfully!')

        elif _type == 'Reset Password':
            #更新密码
            ##hash the code
            m = hashlib.md5()
            m.update(newVal.encode())
            password_m = m.hexdigest()

            user = User.objects.get(username=oldName)
            user.password = password_m
            user.save()
            return HttpResponse('Password Reset successfully!')
    return HttpResponse('Password Reset failed!')
```

Module_7 backend

```
// Reset username or password
reset () {
  var type = document.getElementById('reset-title').innerHTML
  if (type === 'Reset Password' && document.getElementById('inputBox1').value !== this.newVal) {
    this.$message.error('The passwords are not the same, please check it!')
  } else if (this.newVal === '') {
    this.$message.error('You have not set your new ' + type.substring(6))
  } else {
    let sendData = {
      type: type,
      username: this.username,
      newVal: this.newVal
    }
    axios({
      method: 'POST',
      url: 'http://175.178.34.84/updateInformation/',
      data: Qs.stringify(sendData)
    }).then((response) => {
      if (type === 'Reset Username' && response.data === 'UserName has been taken') {
        this.$message.error('The username has already been used, please change it again!')
      } else {
        this.$message.success((response.data))
        this.close()
        if (type === 'Reset Username') {
          this.username = this.newVal
          this.$router.replace({
            path: '/blank',
            name: 'blank',
            params: {
              username: this.username
            }
          })
        } else {
          this.$message('You need to log in again!')
          this.$router.push({
            path: '/login',
            name: 'login'
          })
        }
      }
    })
  }
},
```

Module_7 frontend

```
def uploadProfile(request):
    if request.method == 'POST':
        username = request.POST['username']
        profile = request.FILES.get('profile')
        profile_dir1 = '../../profiles/' + request.POST['id'] + '.jpg'
        f = open(profile_dir1, 'wb')
        for line in profile.chunks():
            f.write(line)
        f.close()
        data = {
            'profile_name': request.POST['id'] + '.jpg'
        }

        # 判断是否改用户已经有头像了
        target_user = User.objects.filter(username = username).values()[0]
        photo_url = target_user['photo']
        if (photo_url):    # 用户有旧的头像，则在服务器上删掉
            prefix_string = ''
            for i in range(0, len(photo_url)):
                prefix_string += photo_url[i]
                if (prefix_string == 'http://175.178.34.84/'):
                    break
            delete_url = '../../' + photo_url[i+1:]
            os.remove(delete_url)

        # 将用户头像的url更新到数据库中
        update_user = User.objects.get(username = username)
        update_user.photo = 'http://175.178.34.84/profiles/'+ request.POST['id'] + '.jpg'
        update_user.save()
        return HttpResponse(json.dumps(data , cls=ComplexEncoder), content_type='application/json')
    else:
        return HttpResponse("Only POST-request is accepted!")
```

Module_8 backend

```
// Callback when the user successfully upload a profile
handleAvatarSuccess (res, file) {
  this.profileURL = 'http://175.178.34.84/profiles/' + res.data.profile_name
},
uploadProfile (option) {
  let fd = new FormData()
  fd.append('profile', option.file)
  fd.append('id', fd.get('profile').uid)
  fd.append('username', this.username)
  return axios({
    method: 'POST',
    url: '/api/uploadProfile/',
    data: fd,
    headers: {
      'Content-Type': 'multipart/form-data'
    }
  })
},
// Execute this before the user upload a profile
beforeAvatarUpload (file) {
  const isJPG = file.type === 'image/jpeg'
  const isLt2M = file.size / 1024 / 1024 < 2
  if (!isJPG) {
    this.$message.error('上传头像图片只能是 JPG 格式!')
  }
  if (!isLt2M) {
    this.$message.error('上传头像图片大小不能超过 2MB!')
  }
  return isJPG && isLt2M
},
```

Module_8 frontend

```
# allow user to like a question/answer
def like(request):
    data = {
        'ok': 0
    }
    try:
        if request.method == 'POST':

            username = request.POST['username']
            Question_or_Answer = request.POST['type']
            target_id = request.POST['id']
            user_id = User.objects.filter(username=username).values()[0]['id']
            if (int(Question_or_Answer)):    # means the id belong to an answer

                already_like = user_like_answer.objects.filter(id= user_id, answer_id = target_id).count()
                if (already_like):     # if already like, then dislike
                    user_like_answer.objects.filter(id= user_id, answer_id = target_id).delete()

                    change_amount_of_like_or_follow(target_id, Question_or_Answer = 0, follow_or_like = 0, add_or_reduce = 0)
                else:
                    user_like_answer.objects.create(id= user_id, answer_id = target_id)

                    change_amount_of_like_or_follow(target_id, Question_or_Answer = 0, follow_or_like = 0, add_or_reduce = 1)
            else:                   # means the id belong to a question

                already_like = user_like_question.objects.filter(id= user_id, question_id = target_id).count()
                if (already_like):     # if already like, then dislike
                    user_like_question.objects.filter(id= user_id, question_id = target_id).delete()

                    change_amount_of_like_or_follow(target_id, Question_or_Answer = 1, follow_or_like = 0, add_or_reduce = 0)
                else:
                    user_like_question.objects.create(id= user_id, question_id = target_id)

                    change_amount_of_like_or_follow(target_id, Question_or_Answer = 1, follow_or_like = 0, add_or_reduce = 1)

            data['ok'] = 1
        else:
            return HttpResponse("Only POST-request is accepted!")
    except:
        return HttpResponse("Invalid input!")

    return HttpResponse(json.dumps(data , cls=ComplexEncoder), content_type='application/json')
```

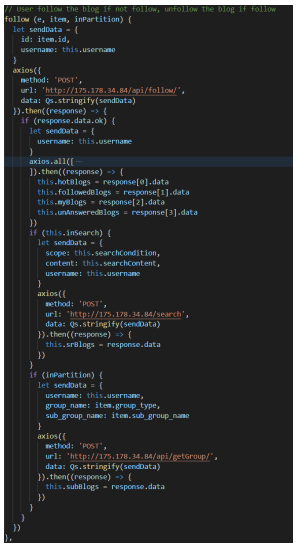
Module_9 backend

```
// User like the blog if no like, dislike the blog if
like (e, item, t, inPartition) {
  let sendData = {
    id: item.id,
    username: this.username,
    type: t // 0 is question, 1 is answer
  }
  axios({
    method: 'POST',
    url: 'http://175.178.34.84/api/like/',
    data: Qs.stringify(sendData)
  }).then((response) => {
    if (response.data.ok) {
      let sendData = {
        username: this.username
      }
      axios.all([
      ]).then((response) => {
        this.hotBlogs = response[0].data
        this.followedBlogs = response[1].data
        this.myBlogs = response[2].data
        this.unAnsweredBlogs = response[3].data
      })
      if (this.inSearch) {
        let sendData = {
          scope: this.searchCondition,
          content: this.searchContent,
          username: this.username
        }
        axios({
          method: 'POST',
          url: 'http://175.178.34.84/search',
          data: Qs.stringify(sendData)
        }).then((response) => {
          this.srBlogs = response.data
        })
      }
      if (inPartition) {
        let sendData = {
          username: this.username,
          group_name: item.group_type,
          sub_group_name: item.sub_group_name
        }
        axios({
          method: 'POST',
          url: 'http://175.178.34.84/api/getGroup/',
          data: Qs.stringify(sendData)
        }).then((response) => {
          this.subBlogs = response.data
        })
      }
    }
  })
},
```

Module_9 frontend

```python
# allow user to follow a question
def follow(request):
    data = {
        'ok': 0
    }
    try:
        if request.method == 'POST':

            username = request.POST['username']
            target_id = request.POST['id']
            user_id = User.objects.filter(username=username).values()[0]['id']

            already_like = user_follow_question.objects.filter(id= user_id, question_id = target_id).count()
            if (already_like):      # if already like, then dislike
                user_follow_question.objects.filter(id= user_id, question_id = target_id).delete()
                change_amount_of_like_or_follow(target_id, Question_or_Answer = 1, follow_or_like = 1, add_or_reduce = 0)
            else:
                user_follow_question.objects.create(id= user_id, question_id = target_id)
                change_amount_of_like_or_follow(target_id, Question_or_Answer = 1, follow_or_like = 1, add_or_reduce = 1)

            data['ok'] = 1
        else:
            return HttpResponse("Only POST-request is accepted!")
    except:
        return HttpResponse("Invalid input!")

    return HttpResponse(json.dumps(data , cls=ComplexEncoder), content_type='application/json')
```

Module_10 backend



```javascript
// User follow the blog if not follow, unfollow the blog if follow
follow (e, item, inPartition) {
  let sendData = {
    id: item.id,
    username: this.username
  }
  axios({
    method: 'POST',
    url: 'http://175.178.34.84/api/follow/',
    data: Qs.stringify(sendData)
  }).then((response) => {
    if (response.data.ok) {
      let sendData = {
        username: this.username
      }
      axios.all([
      ]).then((response) => {
        this.hotBlogs = response[0].data
        this.followedBlogs = response[1].data
        this.myBlogs = response[2].data
        this.unAnsweredBlogs = response[3].data
      })
      if (this.inSearch) {
        let sendData = {
          scope: this.searchCondition,
          content: this.searchContent,
          username: this.username
        }
        axios({
          method: 'POST',
          url: 'http://175.178.34.84/search',
          data: Qs.stringify(sendData)
        }).then((response) => {
          this.srBlogs = response.data
        })
      }
      if (inPartition) {
        let sendData = {
          username: this.username,
          group_name: item.group_type,
          sub_group_name: item.sub_group_name
        }
        axios({
          method: 'POST',
          url: 'http://175.178.34.84/api/getGroup/',
          data: Qs.stringify(sendData)
        }).then((response) => {
          this.subBlogs = response.data
        })
      }
    }
  })
},
```

Module_10 frontend