`

<center>  
 <img src="pic/logo.png" style="zoom:80%;" />  
 <br>  
 <p>香 港 中 文 大 學（深 圳）</p >  
 <p>The Chinese University of Hong Kong, Shenzhen</p >  
 <p style="font-size: 15px;">School of Data Science</p >  
 <p style="font-size: 15px;" >CSC4001: Software Engineering</p >  
 <hr>  
 <b>Course Project Report, Group 18</b>  
 <br>  
 <b>CUHKSZ-Overflow: An Online Forum For CUHKSZ Programmers</b>  
 <hr>  
 <p style="font-size: 20px;">Author:</p >  
 <p style="font-size: 20px;">Huang Pengxiang 119010108</p >  
 <p style="font-size: 20px;">Li Zihan 119010167</p >  
 <p style="font-size: 20px;">Linghu Han 119010185</p >  
 <p style="font-size: 20px;">Qin Peiran 119010249</p >  
 <b>May, 2022</b>  
</center>

`

[TOC]

## Introduction

### Background & Motivation

​ Our group observe that many student-programmers from CUHKSZ may encounter many familiar programming questions in their project or assignment. Their solutions mainly include: searching online, emailing TAs or professors, uploading questions in WeChat group, or assigning an office hour. Searching online sometimes may not be an efficient way, because the blog or some guidance information may not directly answer the assignment question. Even worse, programmers need to spend much time on filtering the huge amount of information and it becomes hard for them to get answers when the homework question is not relevant to the results on website. Uploading questions on WeChat group could get the detailed guidance and answer. But the new WeChat group will be created every semester for other students who may encounter the same problems. The connection between students already taken this course with the students taking this course right now is broken in this way. Raising questions in office hour is not convenient for programmers to solve their question 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.

​ Based on the inspiration of 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 which could save their much time in finishing homework. programmers could use this platform to browse their questions on website and they will get the relevant answer immediately since there are many students who have already taken this course and they may encounter the similar problems. Also, the platform is more friendly. Programmers could use their nickname rather than true name and it is totally private for others. Therefore it would not make them shame about their questions. The platform also supports many formats of questions, which means programmers could use code block to express their specific question rather than word. It is much more efficient and clearer to use code block to express their idea. So students could find the help on our website when they encounter some difficult bugs.

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

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

### Project Overview

​ Our project, CUHKSZ-Stack-Overflow, mainly aims to provide a Q&A platform for CUHKSZ programmers who have technical questions about their Computer Science courses’ projects or assignments in CUHK(SZ). It also offers a social channel where some of students, especially senior students, could share their programming and working experience, provide the guidance in CS learning, and give the 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 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 enable them to accumulate programming experience. Eventually, those communication and connection could save much time for students learning CS and also reduce the teaching load for TA at the same time, which definitely will enhance both learning and teaching qualities in Computer Science courses.

### Objective

​ Our thoughts of 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 and the backend part. In the frontend, users can perform many operations, including registering and logging in their accounts, posting and replying 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 frontend and setting the relevant URL for the connection between 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.

### Key Features

​ Our system is mainly composed of 6 parts, including registration and login, searching relevant questions, posting new questions, replying the questions, displaying the highlight and hot questions or topic, ranking the excellent answers. Users could upload their questions online and also have the access to invite other people to answer their questions. They also could browse those hot questions that remain answered to see if they could help or join in the discussion. Moreover, the searching function is also valid for users. They are allowed to search the questions by themselves with various filter types that could narrow down their searching range. More details of these features will be included in later sections.

| Module ID | Module Name | Frontend Functions | Backend Functions | Description |
| --- | --- | --- | --- | --- |
| 1 | Login | check the username valid, password, | check if exist username, check password correct, | Login for users |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |

### Highlights

#### cloud deployment and website access by domain name

​ Distinguish from running the code in the local terminal and access the website by localhost, our project is deployed on the cloud server, so our project can be accessed directly by typing the URL [http://175.178.34.84](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 become more conspicuous, characteristic and elegant. In this case, our website can be accessed by typing the URL [http://www.cuhksz-stackoverflow.cn](http://www.cuhksz-stackoverflow.cn/). In short, we design and implement a website with an attractive domain name being easy to remember and a remote access from other devices.

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

​ Initially, we use string match method which aims 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 which do not have semantic meaning will also be considered in string match, thus causing a imperfect search results to some degree. For example, the search contents “how to learn programming” may find blogs whose title contains the word “to”. Therefore, we improve our search engine algorithm and adopt the natural language processing algorithm.

​ Apart from using NLP in the process of search content, we also use inverted index in our database. Since the workload of 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 take “kernel” as inputs, record 1, record 2 and record 3 will be retrieved and then record 1 is found to meet the requirements (Table 1.3.2.1), we consider to build a table with inverted index (Table 1.3.2.2). With the application of inverted index, if the user take “kernel” as inputs, record 2 is found to meet the requirements and the blog id 1 is obtained to directly retrieve the corresponding 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 1.3.2.1 - Forward Index

| id | words | blog\_id |
| --- | --- | --- |
| 1 | build | {1} |
| 2 | kernel | {1} |
| 3 | learn | {2} |
| 4 | code | {3} |
| .. | .. | .. |

Table 1.3.2.2 - Inverted Index

​ To sum up, the search content will first be split into words and then the words frequency is calculated to make up a vector (target\_vec). For example, the user take “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 words 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 represents 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 the blog 1 is listed in the former of blog 2.

## System Arcgitectual Design by DFD

## Detailed Description of Components by UML

### 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.

### 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.

### 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.

### 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.

## User Interface Design

### Description of the User Interface

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

​ One of the characteristic in our user interface is **animation**. Our interface integrate many animations in order to make the interface more vivid and more intersting to users. For example, you would notice that there is a dynamic crystal ball waving on the login in page, as well as the animation text in login in page. Also, after you finish the register page, there is also another animation between two different webpage, it is not simply jump to another webpage directly in a rigid way. To make user more pleased when they use our forum, we also designed a loading animation between home page and post blog page. In this way, user would not watch the webpage stuck in a while, or some hacker may use this time to cilck or type something on interface to make the webpage crash, instead, we design this loading page to protect our program but also could provide an animation for user to see to improve their experience feeling.

​ Another hilighted feature for our user interface is **button design**. we took 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 user. For example, to alter user to log out, we make the button highlighed when user hover this button in case user will log out by mistake. To warn user to reset username and passoword, we make the button in a red and a warning way. In those details, we have considered them to make sure they are user friendly as much as possible.

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

### Object and Actions

* **Login**

​ The login page is designed for user directly use their user name and password to get into home page, user must input the correct username and password in our database in order to login. User could use cilck the eye icon to hide or show the password. they can aslo click remember to remember their password. The sign up and forget password button are on the top, user could click those icon to jump to another webpage. The animation text will be shown everytime one referesh 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 condition follow the rules and be responsible for their words in our forum. Notice that the username is unique, if user try to type the username that has been used by others, which is not allowed here. User could also cilck the login button to jump back to login part.

* **Forget Password**

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

* **Verification**

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

* **Home Page**

​ As for the home page, The layout of home page consist three parts. First is the menu on the top, user could click the home page icon and partition icon to go into 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, user could click the button to jump to the post blog part and click 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 catlogs, each catlogs contains different function and order of blogs. In hottest blogs, it mainly contain 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 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 server. The Third part is blogs part, blogs are ordered by some function in catlogs, When user hover each blog, the color will be gray. The tiltle of blogs is bold and the content is normal. each blogs will showing user the view times, the answer amount and its partition type. User also could click the like or follwed button, the color will also changed after user click it.

* **Reset**

​ When User click the Reset Username or Reset password, they could reset their user name or password in the home page. When user reset their password, they are required to log in again.

* **Partition**

​ When user click the partition button, they will jump to this part. This part is used for user to find the corresponding blogs in specific partions. Each partition contains different sub partitons. User 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 User are allowed to click the followed button so that they could see their followed partitions in home page, which is convienent for user to find corresponding blogs.

* **Search**

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

* **Blog**

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

* **Reply**

​ User could click the Answer button to answer each blog. Notice that the identation will be different when user answer in a different situation. When they answer author, they will be classified into first level, otherwise they are second level. The first and second level has different identations so user could identify the blog answer quickly. User are also allowed to cilck like button to supprt this answer, The answer blog will be ordered in time.

* **Post**

​ When user click post button, they could create their own blogs. They are required to type the blog title and content, partition and their sub partitions in order to classify. The content input will be finished in a rich editor, the rich edtior support many forms input and file uploading. User could upload their file or just insert a file link. Our sever will recieve this file and store it into server. User could use the rich editor to create the blog forms they prefer. And all the blogs in home page will be decode as the same as they post in this webpage.

* **Compile**

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

## Test

* For the explanation of our test part, it will be divided into 3 sub-sections. In the first section, the introduction of test-files arrangement will be introduced. In the second sectiosn, 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 are have a good coverage of cases. In the third section, we will report about the testing results.

**Test-Part1: Overall arrangement of test files**

To ensure the functionality, robusty, Genarality of our programs, we implemented the Unit tests, Component tests, and system test based on blackbox testing. The arrangement of test files are 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 provided to users in our website, such as click-on-like button, login system, and so on.
* 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 tests suites to form a 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 test 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 result.

1. test\_follow.py

* The function this file testing is views.follow(request) which is expected to perform the function of empower follow another user by clicking the follow-button we provided in 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 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.

1. test\_followGroup().py

* This file test the views.followGroup(), which is expected to empower user to follow one specific group by clicking button we provided. There are 4 test cases involves: 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-exsiting group’s name.
* As the same idea 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.

1. test\_getGroup.py

* This test file tests the views.getGroup()funciton, which is expected to return all the related blog 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.

1. test\_GetQuestion.py

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

1. 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 picture, number of follows, and whether the current user follow in main-page. There are 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 user try some invalid inputs.

1. test\_like.py

* This test file tests views.like(request) . Since this function is expected to add the amount of like when user who never like this blog/answers clicks the like-botton, and cancel 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.

1. test\_login.py

* This test file test the views.login(). Since this function is the function that require the most robusty, we consider some specials cases that main encounter when using this function.
* First, we test login with user name and correct password. In the second case, we test login with user name and wrong password. In the third case, we test login with sessions (because we keep sessions valid for 1 day if the already login and user can avoid login again within 1 day). In the fourth case, we test login with valid cookies.

1. 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 login page. One normal test case and two boundary(also invalid) test cases are included.

1. 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 corret number of blogs that the user is following, as well as whether the contents are valid. There are also two test cases for invlid request method and non-exsited user name.
2. test\_my\_group.py

* This file tests views.my\_group() , which is expected to return groups that the current user is following. There are one normal input for testing and two invlid and rare inputs for testing as well.

1. 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 a correct result.

1. test\_MyGroups()

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

1. 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 as Python. As well as some invalid input like introducing buggy code to the source code, or use a non-consistent language, or directly not specify the language of source code.

1. test\_Search\_Question.py

* This test file test the functionality of our search engine. We introduce 5 test cases in this file. For example, search the question with chinese, with english. Or, search the questions within spedific scope. We compare the search result with wht we expect to get to evaluate whether this functions work normally.

1. test\_sendEmail.py

* This file test the Email-sending function which is the basic function when 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.

1. 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, corresponding group. Except for one test case that the user post a complete question. We also introduce another 4 test cases to test the situation when user post a question without tile, or without specific group, or without any contetn, or without his/her name included. We check whether this functions is able to handle all of this possible situations.
2. test\_unAnswered.py

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

1. test\_update.py

* This file specially test the functions views.update() which is in charge of enabling users to change 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 register a new account. It includes email sending, user name checking, and database updating. We includes three test cases to verify its correctness.

1. test\_login\_then\_logout.py

* In this component testing file, we consider login and logout functions together to test ensure there will be noting wrong when user login first, then logout the system. Also, when designing the test cases, we specially consider the situations with cookies and sessions to ensure they functions well in this component.

1. test\_Blog.py

* This component testing file tests the correct function of delevering a blog to user when he/she click on the blog.

1. test\_Post.py

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

1. test\_mainpage.py

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

1. test\_like\_follow.py

* This component test file test the like-follow system of our website. It includes 4 test cases to cover the possible situations such as the user clicks likes on a blog/answer then follow it, or click 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 return show that our programs pass all of the testing cases except the API for searching question as it shown in the below figure. Then, we debug the API for searching questions and finally get all the test cases passed.

## Lessons Learned

## Conclusion