Group Members:

Ziwei Zhao (CCID: ziwei11)
Weixi Cheng (CCID: weixi)
Bingran Huang (CCID: bingran1)

1. Overview of System with A Small User Guide

We have two parts in this project. For phase 1, we linked and transferred all data to mongoDB and created a database named 291db. After transferring, we extracted all terms of length 3 characters or more in title and body fields of Posts, added those terms as an array named terms to Posts collection, and built an index on those terms. For phase 2, we designed a system to allow the user to log in. After logging into the system, the user can post a question or search posts by inputting the keyword. The user can also search the posts by inputting one or more keywords, the system will retrieve all posts that contain at least one keyword either in title, body, or tag fields (the match should be case-insensitive). The user can select a question post and see all fields of the question. In the interface of each question, the user also set a button to answer the post and a button to view all answers. Besides, the user can also see all fields of the selected answer. The users can also vote the post they want.

2. A Detailed Design of Software

(1) LogIn class:

The first screen of your system should provide options for users to login. After a successful login or signup, users should be able to perform the subsequent operations (chosen from a menu).

(2) SignIn class:

Main menu interface. There are three functions (Post Question, Search, Vote) By clicking the button, the user can jump to a specific interface and implement its function. There is also a logout button on the interface. If a user id is provided, the user will be shown a report that includes (1) the number of questions owned and the average score for those questions, (2) the number of answers owned and the average score for those answers, and (3) the number of votes registered for the user. Users may also use the system without providing a user id, in which case no report is displayed.

(3) PostQ class:

The user could be able to post a question by providing title, texts and tags. The post should be properly recorded in the database tables. A unique Id should be assigned by your system, the post date should be set to the current date and the owner user id should be set to the user posting it. The quantities Score, ViewCount, AnswerCount, CommentCount, and FavoriteCount are all set to zero and the content license is set to "CC BY-SA 2.5".

(4) Vote class:

The user should be able to vote on the selected question or answer if not voted already on the same post (this constraint is only applicable to users with a user id; anonymous users can vote with no constraint). The vote should be recorded in Votes with a unique vote id assigned by your system, the post id set to the question/answer id, vote type id set to 2, and the creation date set to the current date. If the current user is not anonymous, the user id is set to the current user. With each vote, the score field in Posts will also increase by one.

(5) Search class, ShowResult class:

The user could provide one or more keywords to search posts, and the system could retrieve all posts that contain at least one keyword either in title, body, or tag fields (the match should be case-insensitive). If the user input more than one keyword, the keywords will be split by comma. For each matching post, the title, the creation date, the score, and the answer count should be displayed. If there are more than 5 matching posts, at most 5 matches will be shown at a time, letting the user select a post or see more matches. The user should be able to select a post and perform a post action. The user also can go back to the SignIn class.

(6) PerformPostAction class:

If the user selected a question post, there will be a vote button, an answer button and a show all answers button. Also, after the user selects one question post, its view count should be added one. If you click the answer button, the user can post an answer for the question by providing text. The answer should be properly recorded in the database tables. A unique Id should be assigned by your system, the post date should be set to the current date and the owner user id should be set to the user posting it. The answer should be also linked to the question. If you click the vote button, the user can vote on the question (if not voted already on the same post). The vote should be recorded in the database with a vno assigned by the system, the vote date set to the current date and the owner user id should be set to the user voting it. If you click the show all answers button, the

user can see all the answers (some of their fields) and some buttons(as discussed next).

(7) PerformPostAction2 class:

The user can see all fields of the answer from Posts. After an answer is selected, the user can vote for the selected answer. The user also can go back to the SignIn class.

(8) Answer class

The user should be able to answer the question by providing a text. An answer record should be inserted into the database, with the body field set to the provided text. When an answer is posted, its parent post's answer count will be added one. A unique id should be assigned to the post by your system, the post type id should be set to 2 (to indicate that the post is an answer), the post creation date should be set to the current date and the owner user id should be set to the user posting it (if a user id is provided). The parent id should be set to the id of the question. The quantities Score and CommentCount are all set to zero and the content license is set to "CC BY-SA 2.5".

(9) ShowAnswer class:

If the user select show all answers button, the user should be able to see all answers of a selected question. If an answer is marked as the accepted answer, it must be shown as the first answer and should be marked with a star. Answers have a post type id of 2 in Posts. For each answer, display the first 80 characters of the body text (or the full text if it is of length 80 or less characters), the creation date, and the score. The user should be able to select an answer to see all fields of the answer from Posts. If there are more than 5 matching answers, at most 5 matches will be shown at a time, the user can click the searchmore button to see more answers. After an answer is selected, the user may perform an answer action (as discussed next).

3. Testing Strategy

We write 10 different classes, each class represents a different interface and function. So we can easily test our source code based on these classes.
Test cases include storing directly into the database before the system is running, and manually storing them when the system is running.
Here is our general strategy for testing, with the scenarios being tested.

For all users:

(1) Test registered user information (LogIn class, SignIn class)

(2) Test post a question (PostQ class)
(3) Test search a post (Search class, ShowResult class)
(4) Test select a question (PerformPostAction class)
(5) Test post a answer (Answer class)
(6) Test show all answers (ShowAnswer class)
(7) Test vote a question / an answer (PerformPostAction2 class)
(8) Test select an answer (PerformPostAction2 class)

4. GroupWork

Our team distributed tasks fairly evenly, we designed the system, interface and database together. It took about 5 days to complete the source code. In order to keep the project on track, we always discuss and share our own ideas. Also, no matter which team member encountered difficulties, others will help s/he to solve them together.
The following is the specific division of tasks.

Ziwei Zhao:
main(), Search class, ShowResult class, PerformPostAction class, ShowAnswer class, PerformPostAction2 class

Weixi Cheng:
main(), LogIn class, SignIn class, PostQ class, Vote class, PerformPostAction class, PerformPostAction2 class

Bingran Huang:
main(), Answer class, ShowAnswer class, PerformPostAction class, PerformPostAction2 class