Design Document for Mini-Project 2

< General Overview of Our System >

Our program was designed for users to get access to the user information from the database and perform various actions. It is divided into 2 phases. In phase 1, the program reads the given three json files—Posts.json, Tags.json, and Votes.json—and builds a collection for each file in MongoDB. In phase 2, by connecting to the database created in phase 1, the program allows the users to complete various tasks.

Installation:

- Install dependencies and build the project:
  $ ./construct

Phase 1:

- Start phase 1 to insert examples data to MongoDB:
  $ python3 mini_project2 phase1

Phase 2:

1. Enter the following command in the terminal to start the program of Phase 2.
   $ python3 mini_project2 phase2
2. The user report is displayed, and the user is able to perform the following actions:

Main Menu:	***Sub Menu:***
➔ [pq] Post a Question ➔ [sq] Search Questions ➔ [q] Quit	➔ [pa] Post an Answer ➔ [la] List Answers ➔ [vp] Vote on a Post ➔ [bm] Back to Menu

< The Detailed Design of Our System >

Phase 1:

The program prompts the user to enter a port number under which the MongoDB server is running and creates a new database named *291db* if it doesn't exist. Then, it connects to the database 291db and creates three collections—posts, tags and votes—by reading Posts.json, Tags.json and Votes.json files. Cython library is used to support the performance-intensive work: extracting terms.

extractTermsFrom(*postDoc*)

- Extracts unique terms from the title, body, and tags fields—if they exist—in the given post document, and returns them in a list.

Phase 2:

The program prompts the user for a port number under which the MongoDB server is running and connects to the *291db* database on the server. Then, it asks for the user to enter their user id; the user is also able to log in as anonymous.

displayReport(*db, uid*)

- Displays a user report if a user id is provided by the user.
- The user report includes the following:
    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

postQ(*db, uid*)

- Prompts the user to post a question by entering a title, a body text, and zero or more tags.
- Inserts the new post into the *posts* collection, with a unique pid generated by the system.
- Inserts a new tag into the *tags* collection, with a unique tagId, if there is at least one tag entered by the user.

searchQ(*db*)

- Prompts the user for one or more keywords, and searches for any question posts that have terms equal to the given keywords.
- Displays a table that includes 5 matching posts at once and prompts the user to view more posts if there are more posts to show. The user can select a post from the table.
- After the user has selected a post, it displays all of its fields and increments the number of ViewCount by one.
- Prompts the user to choose one action from the following list of actions: post an answer, list answers, and vote on a post

postAns(*posts, uid, targetPid*)

- Prompts the user to make an answer post to the selected question by asking for a body text.
- Inserts the new answer document into the *posts* collection, with a unique pid generated by the system.
- Increments the number of AnswerCount of the target question by one.

listAnswers(*posts, targetQ*)

- Displays all the answers corresponding to the selected question post.
- An accepted answer is marked with a star and listed at the top.
- The user can select an answer post to see all its fields.
- It then prompts the user to vote on the selected answer post.

votePost(*votes, uid, targetPid*)

- Prompts the user to vote on the selected post.
- The user is allowed to vote on the same post only once.
    - This constraint is applied only if the user is not anonymous.
- Inserts a vote document into the *votes* collection, with a unique vid assigned by the system.
- Increments the score of the target post by one.

< Testing Strategy >

Phase 1:

After running Phase 1, we checked if 1) all the data was correctly inserted in the database, 2) each collection had all the fields required, 3) the program extracted terms correctly, and 4) the program created an index for terms, by MongoDB searches.

Phase 2:

We first set up the database in Phase 1 and examined Phase 2.
There were two main ways to test Phase 2:

- Search posts using the program's search functionality,
- Search posts using direct access to the database through MongoDB command-line UI

For the functionalities of which their updates are visible as outputs of the program, we were able to confirm the updates directly from the program. For the functionalities that do not display the updates, on the other hand, we used MongoDB search to check if the change was correctly reflected in the database. We also came up with some different scenarios to check the functionalities.
Posting a question, for example, we tested out when the user entered no tags and multiple tags. For searching questions, we used multiple keywords which consist of upper and lower cases to test its case-sensitivity using keywords such as "DaTaBase," "maCbook" and "software." For voting on a post, we signed in as anonymous and non-anonymous to check whether the constraint for votePost() was applied only to the non-anonymous users.

< Group Work Breakdown Strategy >

In the first meeting, we distributed the workload. We then set deadlines and had meetings to check on group members' progress. After completing all of the functionalities in Phase 1 and 2, we tried to optimize Phase 1 to reduce the time the program took to complete building the database in MongoDB. We also revised our code multiple times and tested it together by using the testing strategies discussed above and the rubric posted on the spec.

Work items breakdown by each member:

Jejoon Ryu (time spent: 20 hr)	Jayden Cho (time spent: 20 hr)	Moe Numasawa (time spent: 20 hr)
- Optimizing phase 1 - Phase 2 revision - List all answers	- Optimizing phase 1 - Phase 2 revision - Testing	- Post a Question - Post an Answer - Search Questions - Vote on a Post - Reports.pdf