1.

	Worst Time Complexity	Best Time Complexity
addPostToDatabase(User u, Post p)	O(n)	O(1)
computeMostUrgentQuestion()	O(n)	O(n)
retrievePost(User u)	O(n)	O(n)
deletePostFromDatabase(User u, Post p)	O(n)	O(1)

Best case scenario:

addPostToDatabase: The best case occurs when the user is at the first of the user list so it does not need to iterate through the user list to see whether the user is enrolled.

computeMostUrgentQuestion: The best scenario is the same as the worst because it always needs to iterate through the array to compare the priority.

retrievePost: The best scenario is the same as the worst because it always needs to iterate through the array to find whether the post was associated with the user.

deletePostFromDatabase: The best scenario is when the deleted post is at the end of the array so that the array does not need to shift the elements, and also the user is at the first of the user list so that it does not need to iterate through the user list.

2. Advantages: 1. Arraylist is not limited to the length. We do not need to specify the length of the list when we are creating an arraylist.
   2. It is easy to insert or remove the particular element in the arraylist.
   3. It is easy to get the element by index.
   4. We can easily access the elements in the arraylist.

   Disadvantages: 1. When the element is added to the arraylist or removed from the arraylist, if it is not at the end of the arraylist, the arraylist needs to shift the elements inside.
   2. Once the arraylist exceeds its capacity, it has to copy the current array and creat a new array.

3. Throughout theses questions, I changed the data structure Arraylist to Hashmap. Because compared to the Arraylist, Hashmap has a lower average runtime. So when I am adding a new post, a hashmap just need to find the hash value and insert the keyword as the key and the Post arraylist as the value which takes the average runtime of O(1). It also saves a lot of time for searching and deleting, because I just need to use the key to find the node and I will get a list of post with the same keyword. I built three hashmaps. The first is called posthashmap which takes keyword as the key, and post arraylist as the value. Second is called userhashmap which takes user as the key, and post arraylist which contains all of the post that associated with the corresponding user as the value. Third is called unansweredhashmap which takes UID of the post as the key, and the post itself as the value. Therefore, it will be quick and easy to access the target post. For the function addPostToDatabase, the new hashmap just needs to use the post's keyword to find the node and add the post to the node's value. And it also need to add the post to the userhashmap and unansweredhashmap (if it is a question) in the same way. It functions in the same way for deleting a post. For retrieving the post, hashmap works much faster than an arraylist. Because if we're using an arraylist, we need to go through every post and see whether it is associated with the user or contains the same keyword. For hashmap, however, it only needs to

go to the posthashmap and use the keyword as the key to find all of the posts that have that keyword. Same for the userhashmap, it only needs to know the user we are trying to search, and use that user as the key to search userhashmap, we can get all of the posts that are associated with that user. For computing the most urgent question, I created another hashmap and takes the priority of the question as the key to store all of the unanswered questions. Therefore, I just need to search the highest priority in the hashmap and I will get the most urgent question. Last, for answering the question, using a hashmap only requires me to take the UID of the question that user is trying to answer as the key, the hashmap will find out whether that question is in the unanswered list or not in a short time. Therefore, I believe hashmap maximizes the efficiency by reducing the time for adding, deleting and searching the post.