# IMPROVED AND REVISED!

**What have we changed?**

- Added the Post dislike feature
- Modified the friend suggestion feature

M. Ahmad Raza -> SP21-BCS-003

Deepak Kumar -> SP21-BCS-017

Malik Rashid Ali -> SP21-BCS-041

Date: 3rd July 2022

## Explanation of Friend Suggestion Feature:

Our old feature had some ambiguities in its working like it wasn't that way efficient in terms of suggesting the number of users, let's see the problem we have solved so far.

We have taken extra two data structures which are "set" and "HashMap". Set works on the principle that it can insert only the unique value and only once if it gets the value of same type, it will not insert that one again and the HashMap works on the principle that the hash Code is used to check if there is already a key with the same hash Code.

In our own defined structure of user, we have assigned it in a matrix in way that the 0 index of matrix represents the user who is currently logged up in their accounts and the rest of index in that row represents the friend list of that user. Now our main motive is to iterate the friend list of that user to get some assumptions about their friends.

Now take a view here that I am Deepak, and I am logged up in my account and I have a friend name Rashid now we will iterate Rashid friend list to give some suggestions of Rashid's friends to Deepak, to do this let's iterate through Rashid friend list. Suppose we are now iterating through Rashid friend list and by iterating we have found that Rashid friend list contains Ahmad, Faisal, Deepak. We know this that Rashid is already a friend of Deepak, but Deepak is not friend of Rashid rest of friends that are Ahmad and Faisal, so now we must do the operations in a way that Deepak can get the suggestion of Ahmad and Faisal in his suggestion list.

The work we have done to achieve this functionality is first we have iterated through Rashid friend list and in that we have applied some validations so that program does not get stuck in between its runtime duration, after that we have checked that if that friend of Rashid is already a friend of Deepak or not if it wasn't the friend of Deepak then do the next action. The next action is to put the Ahmad in hash Map and in Set, this process goes on until the Rashid friend list becomes empty.

**Now what is happing in HashMap and Set right?**

Well, the answer is that we have defined our hash map structure which have data members like value (asci value of string username) and frequency which is checking how many of Deepak's friend that user is connected to.

The set here is working on the same usual algorithm of its own that is not allowing duplicate values and only allowing the unique values.

Till now we know that the values are in their specific containers **what next, we must do to get some fruitful insights from it?**

Now the next thing we are doing is we are iterating through our set which is containing the unique user values in it, creating a new hash map object in that we are getting the asci value from our inserted username indexes, also we are setting the map value by pointing it to the pointer of set.

Now all the values are stored, the next savior is here is our own defined priority queue in which are inserting the map itself and the queue working the principle that its choices the map with higher frequency number and set it according to its priority.

Now we know all stuff are done so we are now simply calling the display method to show the queue stored values to the current logged In User.

## Code of Above Explanation:

```
bool myFriendExists(string userName, int index)
{
    for (int i = 0; i < SIZE; i++)
    {
        if (matrix[index][i] == NULL)
        {
            return false;
        }
        if (matrix[index][i]->user->username == userName)
        {
            return true;
        }
    }
    return false;
}


void friendSuggestions(User *user)
{
    unordered_set<string> names;
    int userIndex = search(user);
    if (matrix[userIndex][1] == NULL)
    {
        cout << "NO FRIEND SUGGESTIONS LIST IS AVAILABLE\n";
        return;
    }
    for (int i = 1; i < SIZE; i++)
    {
        if (matrix[userIndex][i] == NULL)
        {
            break;
        }
        else
        {
            int friendIndex = search(matrix[userIndex][i]->user);
            if (friendIndex == -1)
            {
                cout << "NO FRIEND SUGGESTIONS LIST IS AVAILABLE\n";
                return;
            }
            for (int j = 1; j < SIZE; j++)
            {
                if (matrix[friendIndex][j] == NULL)
                {
                    break;
```

```
                }
                else if (!myFriendExists(matrix[friendIndex][j]->user-
>username, userIndex))
                {
                    // cout << "SUGGESTED FRIEND IS: " <<
matrix[friendIndex][j]->user->username << endl;
                    pushHashMap(matrix[friendIndex][j]->user->username);
                    names.insert(matrix[friendIndex][j]->user->username);
                }
            }
        }
    }
    Que q(100);
    unordered_set<string>::iterator it;
    for (it = names.begin(); it != names.end(); ++it)
    {
        hashMap *map = new hashMap;
        map->frequency = hash_map[getAsciValues(*it)];
        map->value = *it;
        q.enqueue(map);
    }
    q.display();
    cout << "ENTER USER NAME TO SEND REQUEST(0 to return): ";
    string recevierName;
    cin >> recevierName;
    if (recevierName == "0")
    {
        return;
    }
    User *recevier = search(recevierName);
    if (recevier != NULL)
    {
        send_request(user, recevier);
        friendSuggestions(user);
    }
    else
    {
        cout << "RECIEVER NOT FOUND";
        friendSuggestions(user);
    }
}
```

## Explanation of “Dislike the Post” Feature:

It has the same working as like feature which we have explained in class presentation that how the weight between the two user’s was decreasing based on liking the posts of one another. The similar algorithm works here only the changed here is that we have changed some conditions if a user dislikes the post of another user the weight between these two users increases and it results in a way that the now the friend suggestion between them declines.

## Code of Above Explanation:

```
void dislike(User *user, string usernameInput, string id)
{
    if (!postIdVerification(id))
    {
        cout << "<<<<<<<<<<<<<<< INVALID POST ID >>>>>>>>>>>>>>>" << endl;
        return;
    }
    int postId = stoi(id);
    int liked = search(search(usernameInput));
    if (liked == -1)
    {
        cout << "USER NOT FOUND";
        return;
    }
    Post *post = NULL;

    int count = 1;

    if (matrix[liked][0]->user->posts.size() < postId)
    {
        cout << "\n <<<<<<<<<<<<<<< INVALID POST ID >>>>>>>>>>>>>>> \n";
        return;
    }
    for (auto const &item : matrix[liked][0]->user->posts)
    {
        post = item;
        if (count >= postId)
        {
            break;
        }
        count++;
    }
    post->dislikes++;
    post->postUser.push_back(user);

    int liker = search(user);
    for (int i = 1; i < SIZE; i++)
    {
        if (matrix[liked][i] != NULL)
        {
            if (matrix[liked][i]->user->username == user->username)
            {
                matrix[liked][i]->weight += 4;
```

```
            }
        }
        if (matrix[liker][i] != NULL)
        {
            if (matrix[liker][i]->user->username == matrix[liked][0]->user-
>username)
            {
                matrix[liker][i]->weight += 4;
            }
        }
    }
}
```