Social Media Website – **Gatherly** - Milestone 2

By Abel Weldaregay and Neha Niphadkar (fordFanatics)

Milestone 2 is where we as a group had to think through our design choices, future implementation had to be taken into consideration. We realized early on how difficult it is to work with CSS. We aimed to make our layout more dynamic and scalable. We started with few basic requirement for M2 like the layout for profile picture and the registration page. The layout for uploading the profile picture/avatar was fairly easy, HTML gives you the flexibility to do this in a single line of code restraining the upload option to only formats supported for an image like .jpg, .jpeg, .png, and .pdf. HTML it prevents you from uploading a .txt file.. To have this displayed next to the username was challenging because then it meant that we had to change the layout. This as a requirement we should have anticipated, it should have been intuitive so that we could have had a layout that was more accomodating.

|  |  |  |
| --- | --- | --- |
| **TASK** | **Progress** | **Challenges faced** |
| 1. 1. Users should be able to add "reactions" to posts | Yes | The most challenging part about this was storing the reactions in the database and making sure there are no duplicate reactions. The way we ended up solving this problem was to make sure that the user id and message id in the reactions table do not repeat. |
| 2. At a minimum, users should be able to provide :+1: and :-1: reactions | Yes | The biggest challenge for this requirement is differentiating between a dislike reaction and a dislike reaction from a like, meaning that if the user dislikes a message that was previously liked by that same user, it should decrement the like twice since the message losses a like (-1) and then gains a dislike (-1). |
| 3. Users should not be able to add duplicate reactions | Yes | Clicking on the like button give it a value of +1, on a double click it does not increment it by +1. |
| 4. Posters should be able to reply directly to posts (creating a sub-thread) instead of simply posting to the open forum of a group. | Yes | The most challenging part about this requirement was differentiating between which post each comment belongs to. The way we fixed this issue was by having a column in the comments table that references the parent message. |
| 5. Replies to posts should be grouped -- in some intuitive way (e.g., indented) -- with their root post and other replies | Yes | The layout for indentation was achieved with creating a nested div inside the existing div. |
| 6. Users can create public groups (i.e., anyone can search for and join the group) | Yes | The biggest challenge for this requirement was that if a user in part of a private group and copies the url of that private group and logs out and a user not part of the group logs in, that new logged in user should not be able to access the groups page by using that url. The way we checked for this is that right when a user attempts to access a groups page, we first fetch all the groups the user is in from the database, and compare it with the group the user is trying to access. We redirect to the global page if the user is not part of the group and if the user is part of the group, we allow them access to the groups page. |
| 7. Users can invite members to groups | Yes | The biggest conflict we faced with this requirement was on how to store group invitations. We came to the conclusion that if we have the user id that is being invited and the group id, then we will be able to allow access to the user being invited. After the owner of the group sends an invitation to a user, the user id invited is added in the invite table along with the group id invited too. When the invited user logs in, they will have a notification in My Invitations bar. Inside that bar, it displays the group name and the option to either accept or reject the invitation. |
| 8. Users can make private groups (i.e., invitation by owner, only) | Yes | We faced similar problems discussed in tak 7 for task 8. The main difference is that for public groups, we made it so that any user can search and join the group and also get invited by the group owner. With private groups, only the owner of the group can send invitations, and the invited user can either accept or reject the invitation. |
| 9. An interface must be provided to allow a user to upload a profile picture (avatar). | Yes | Creating an interface was fairly simple, html gives you this option and there is no need to look at other technologies. However when it comes to the placement of on what stage of the registration or the login in page do we give user the access to upload his avatar, we decided on giving this option once the user and registered and logged in. It is strictly from the point of view of security. |
| 10. A user's avatar should be displayed alongside all posts by the user. | Yes | The positioning was achieved with html and css, if the user does not upload an avatar a default avatar will be assignment to the user. The file name for this is, “avatar.jpg”. |
| 11. Create a PROFILE PAGE that uses PHP query parameters to build the page's contents. This query parameter must indicate a value that can be used to uniquely query the user from your database. On this profile page, display the user's username, avatar, and channel membership (except private channels). | Yes | Creating a layout of the profile page for easy, the only additional part we had to learn was to uploading an image or an avatar. |
| 12. Provide an HTML form for new users to register for your website by providing a username, email, and password. When a username or email is provided for registration that already exists in your database, reject the rejections and ask the user to select another. | Yes | The biggest challenge for this requirement was checking not only if the username is taken, but also checking if the password is taken too. The way we ended up solving this issue was by querying the database for the username supplied and the email supplied. We wrote a simple function that returns a boolean depending if it is a valid user or not. Depending on the return value of this function, the user will be registered or declined. |