Web Programming – Milestone 2

InterConn – Mahesh, Maheedhar, Rohila (Group 2)

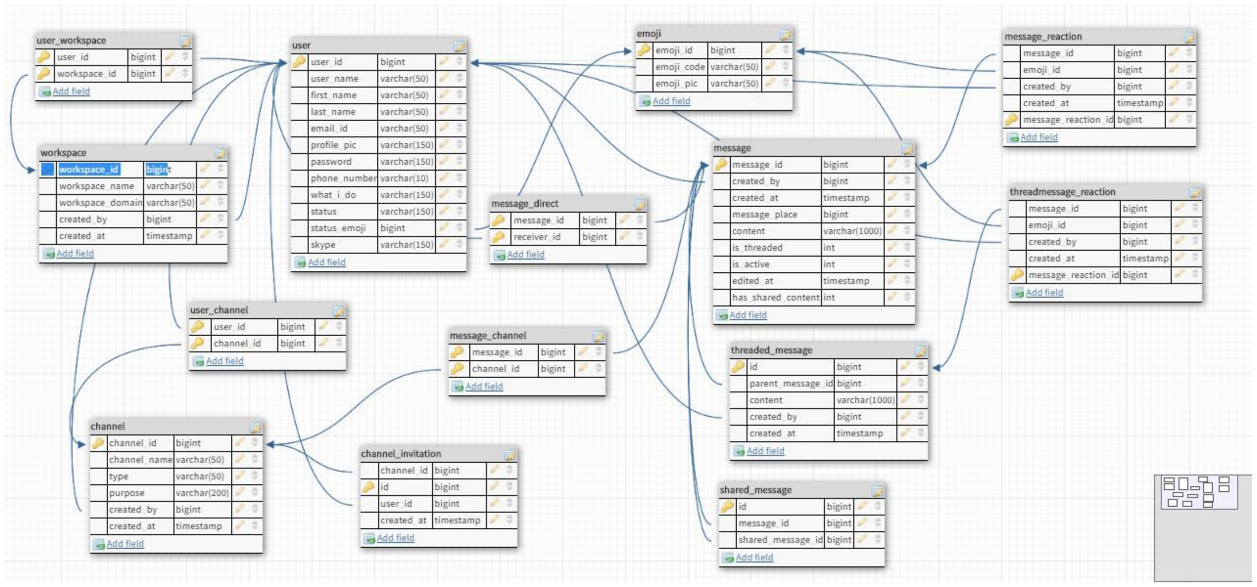
Features Implemented:

1. User can add two reactions (like and dislike) to messages in channel and also to replies for a message in thread. A reaction to a message cannot be duplicated by same user but many users can react to a message.
2. Users can create sub-thread to a message in open forum of a channel by clicking on a thread button that appears on hover of any main message content. Replies can be added to this thread using the small chat box beside the main chat window.
3. The main thread and their replies are grouped together and displayed together, Also on clicking the thread header (on the message content), the corresponding main message is shown with a different background color dynamically.
4. Users can upload a profile picture using the edit option on their profile page. User is navigated to their profile page onclick of their full name present on their home page. Also, all the user messages are tagged with their profile picture.
5. New users can register and create an account using the 'Create your account now' link available on our login page. The email id used to register shouldn't exist in our database. Newly registered user into a workspace will have two channels namely, general and random by default.
6. Users can view their profile consisting of their full name, email, avatar, status and public channel details by clicking on their name on the homepage of our website. Also a logged in user can view any other user's profile information (only public data) on clicking on the name of the user of interest on the message header.
7. Users can create new channels (both public and private). New members can be invited to the channels while the channel is created and also from an existing channel. Members can be invited to a private channel only by the owner (one who created the channel). Members cannot be invited to default channels (general and random) as everyone in the workspace are members of those by default.

Tasks which took more attention:

➔ **Creating threads to a message:** We made sure that we have the DB designed right initially by keeping the threads functionality in mind, but however we had to add an additional table (threadmessage_reaction) to take care of reactions for the messages in threads and properly normalised the DB.

Our current DB snapshot looks as follow:



For the thread message submission instead of using the form submit we have incorporated the AJAX way of posting the data. And on success of AJAX post, updating the UI elements like **number of replies** on both the main message and on the thread header is taken care of.

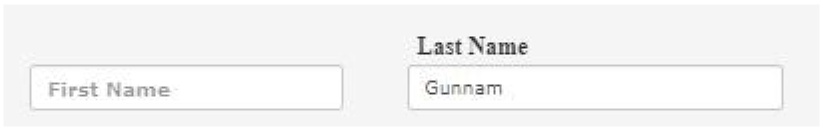
➔ **Profile Picture**: This was an easy walk in the beginning, the tasks like rendering the image on the UI, soon after selecting an image and before the image gets updated on the server is done after a bit of research and is accomplished by FileReader(), a native method for which the reference is added below. Once the image is read in the UI then it is scrutinized for the dimension validation. We learnt that the folder that holds these images on the server should have writable permission for which we had to impose 777 on folder holding images. All happies until we figured out that .png couldn't be uploaded to dockerized version by saying permission denied and whereas all the other formats .PNG, .JPG, .jpeg, etc., are able to make it. And the weirdest part is any type of image including .png is working fine with in our local and as well as on one of the lab servers when hosted, same username and password have to work for this set up as well (http://qav2.cs.odu.edu/rohila/WebProgramming-CS518/index.php).

➔ **Client Side Validations and how we cheated HTML form submission** : To get HTML5 validation working even though we are making AJAX Post call for data submission we did play a trick with the HTML form submit functionality. The following code snippet explains it all:

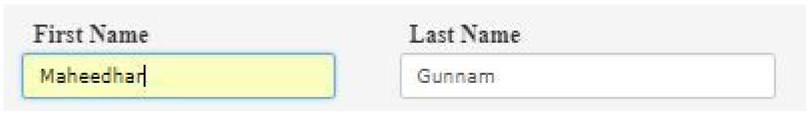
```
$(".not_reallyrequired").removeAttr("required");
if(!$("#updateForm")[0].checkValidity()){
        $("#dummysubmit").trigger("click");
        $(".not_reallyrequired").attr("required",true);
        Return;
}
```

```
$(".not_reallyrequired").attr("required",true);
```

There is an other real need for us to follow this route, we are using a special CSS for our HTML forms for the desired UI aspect as explain below with pictures



As show in the above picture the First Name in the input field is serving as both place holder and the field title. The moment user start typing in the field the placeholder turns to be the field title as show below. But to get this, the requirement is to have required attribute mandatory on the input field. To preserve both this look and feel and as well as for the HTML form validations on the actually required fields we got to do this circus with in our code.



- ➔ Search input tags- We have used the select2 plugin for inviting multiple people into a channel at one time. This plugin is an enhancement of select input field, which allows us to select multiple items at once. It has an feature to search among the given suggestions. All the selected options are displayed in form of tags in the select field and there is also an option to remove the selected items. By giving the data consisting of ids and names of members, it constructs the suggestion box by itself.
- ➔ When a user registers into the site, we give him an option to choose the workspace and register into that workspace. User is prompted with all the available workspaces on our site to choose from. At a later point, we have to modify this into creating a new workspace or user can only join if they get an invite from existing members of the workspace. A user in one workspace cannot interact with others in another workspace. Also, members of one workspace can only invite (to a channel) members belonging to the same workspace.

Why AJAX :

As very well known to everybody AJAX stands for Asynchronous JavaScript and XML. This technique basically is to get asynchronous nature to javascript, hence preventing from long waiting data to be fetched from server turning to be a blockage for the execution of independent client side script.

➔ In other words, with the context of our slack like site that we are designing, If we put our code that gets all the messages of a channel to be done through AJAX, as soon as messages are got to the client side, even before these messages are rendered an AJAX call to get the threads replies for the most recent messages can be fired and while the thread messages are being fetched, parallely main messages can be rendered without having to wait for the threads messages to be downloaded.

➔ The other huge advantage using the AJAX is that instead of re-fetching and re-rendering the whole DOM (Document Object Model), only the sections which needs to be modified with the updated data can be modified dynamically by fetching the updated data alone using AJAX. Again in the context of our slack like site, when a user leaves a message in the channel, only this new message can be dynamically appended to the message list instead of reloading the whole document (i.e even the unchanged Channels list, site header, footer and stuff) hence saves lot of network traffic and load time. AJAX comes in aid for such kinds of practice.

Why jQuery and JavaScript :

➔ What all PHP could do is execute server side PHP code, connecting to database, fetching the data and preparing the HTML content by including the data fetched and pushes the content to client browser on request. Now once the client browser gets this HTML content and renders it to appeal the user with the beautiful look and feel. Every browser engine comes with a native JavaScript . Without JavaScript , the HTML content that browser displays is like a skeleton without life. Now JavaScript gets the dynamism to the static HTML and jQuery is a framework built on top of javascript. jQuery is very well known for its selector based mechanism for modifying DOM elements. The code written in jQuery is more readable than basic JavaScript code, for example, jQuery replaces document.getElementById and document.getElementByClass by '**$('#eleID')'** and '**$('.eleClass')'** respectively. $.ajax method is used for making AJAX requests, whereas $.post and $.get are flavors of AJAX methods.

References

1. https://www.w3schools.com/php/
2. https://www.w3schools.com/bootstrap/bootstrap_modal.asp
3. https://stackoverflow.com/
4. https://jquery.com/
5. https://www.phpmyadmin.net/
6. http://www.writephponline.com/
7. http://jsonviewer.stack.hu/
8. https://slack.com/
9. https://jsfiddle.net/rustybailey/2b7dD/
10. http://select2.github.io/select2/
11. https://www.w3schools.com/php/php_file_upload.asp
12. http://jsfiddle.net/hardiksondagar/t6UP5/
13. https://codepen.io/peiche/pen/xOVpPo