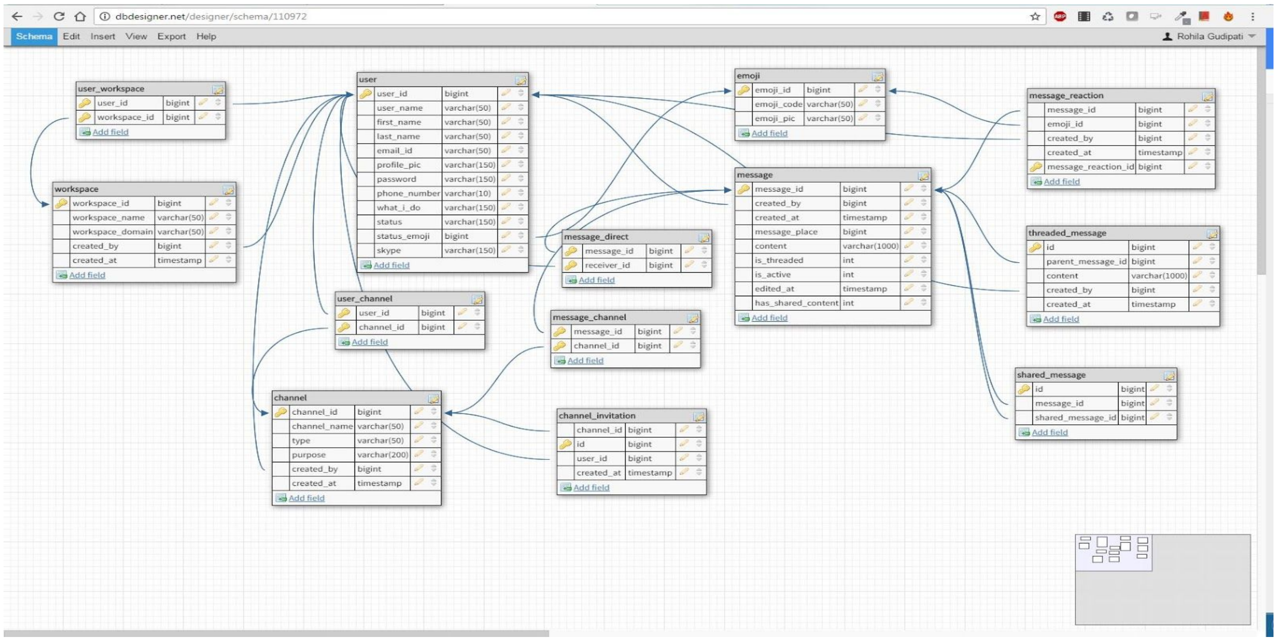
Web Programming – Milestone 1

InterConn – Mahesh, Maheedhar, Rohila (Group 2)

The End Goal of this course project is to develop a collaboration system like Slack. To enact a system of a kind, first one must get familiarize with an existing one. With that thought, we explored nearly all the features that slack provide. We learned that, heart of any application is to have a scalable DB. And together as a team we brainstormed and came up with most of the tables, each table attributes and the relationships between these identified tables. We took care of all the future requirements as far as possible with respect to DB Design. We tried to adhere to the 1st, 2nd and 3rd Normalization forms so as to have an optimized Database.

Our current DB snapshot looks as follows:



user_workspace	
user_id	bigint
workspace_id	bigint

workspace	
workspace_id	bigint
workspace_name	varchar(50)
workspace_domain	varchar(50)
created_by	bigint
created_at	timestamp

user	
user_id	bigint
user_name	varchar(50)
first_name	varchar(50)
last_name	varchar(50)
email_id	varchar(50)
profile_pic	varchar(150)
password	varchar(150)
phone_number	varchar(10)
what_i_do	varchar(150)
status	varchar(150)
status_emoji	bigint
skype	varchar(150)

user_channel	
user_id	bigint
channel_id	bigint

channel	
channel_id	bigint
channel_name	varchar(50)
type	varchar(50)
purpose	varchar(200)
created_by	bigint
created_at	timestamp

emoji	
emoji_id	bigint
emoji_code	varchar(50)
emoji_pic	varchar(50)

message	
message_id	bigint
created_by	bigint
created_at	timestamp
message_place	bigint
content	varchar(1000)
is_threaded	int
is_active	int
edited_at	timestamp
has_shared_content	int

message_direct	
message_id	bigint
receiver_id	bigint

message_channel	
message_id	bigint
channel_id	bigint

channel_invitation	
channel_id	bigint
id	bigint
user_id	bigint
created_at	timestamp

message_reaction	
message_id	bigint
emoji_id	bigint
created_by	bigint
created_at	timestamp
message_reaction_id	bigint

threaded_message	
id	bigint
parent_message_id	bigint
content	varchar(1000)
created_by	bigint
created_at	timestamp

shared_message	
id	bigint
message_id	bigint
shared_message_id	bigint

Features Implemented:

1. A user can login to the website using the details in course users list. The login process uses HTTP POST method as per the requirements. Session Implementation was taken care of.
2. The home page of the website has two sections, first section has set of existing channels which are populated dynamically, users in the workspace and name of the workspace. The second section, message display area contains all the messages in the selected channel.
3. User can post messages using the textarea at the bottom of the screen in second section. The messages are sent to the server using HTTP POST method.

4. The message display area contains name of the user,who sent a message, message sent time and the message. Additionally messages are sorted in the order of the time they are sent.
5. All messages are stored in the MySQL database in the message table.
6. The first section on the page provides links to other existing channels and user can navigate to the channels provided.

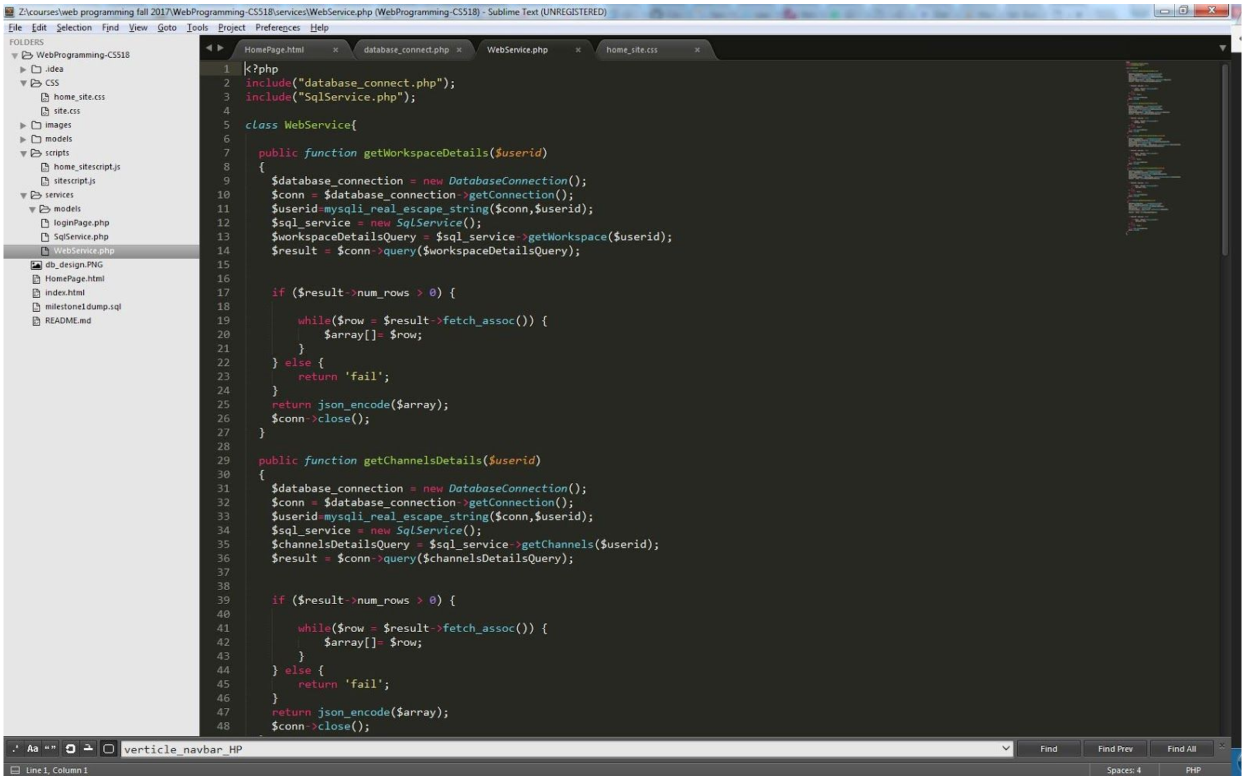
Tasks which took more attention:

- **Name Cards Produced Dynamically:** One interesting feature implemented is to have a default picture which is dynamically generated for every user posting a message in the channel, for nearly unique identification, for which we have used the first letter of first name and last name and the background color to be decided by the user_id of the user being modulo divided by the 5 inturn working as a hashing function.

- We used the method **mysql_real_escape_string** to prepare the input string to be safely used in a MySQL string declaration by escaping certain characters so that they can't be misinterpreted as a string delimiter or an escape sequence delimiter and thereby allow Sql Injection attacks. We have used htmlspecialchars method in php and preserve tag to display the strings on the web page.

- **CSS Position Attributes:** Different Position attributes possible are Static, Fixed, Absolute and Relative. Dom Elements which are positioned Static will be render as it appeared in HTML code and this is the Default position. Position Fixed will render an Element with respect to viewport and requires attributes like top and right to be set. Position Absolute is to render an element with respect to the positioned parent if not any then is positioned according to the viewport. Position Relative makes an element appear related to its original static position.

- We have implemented two layers at the server side, WebService layer (Which returns a JSON Object after calling a SQL Query method and iterating over the results) and Sql Query Layer (Has all the Query being written here) and a Third layer Controller (to decide which WebService method to be called) like layer to be implemented for future milestones.Our current folder structure for the code looks as follows:



```
<?php
include("database_connect.php");
include("SqlService.php");

class WebService{

  public function getWorkspaceDetails($userid)
  {
    $database_connection = new DatabaseConnection();
    $conn = $database_connection->getConnection();
    $userid=mysqli_real_escape_string($conn,$userid);
    $sql_service = new SqlService();
    $workspaceDetailsQuery = $sql_service->getWorkspace($userid);
    $result = $conn->query($workspaceDetailsQuery);


    if ($result->num_rows > 0) {

        while($row = $result->fetch_assoc()) {
            $array[]= $row;
        }
    } else {
        return 'fail';
    }
    return json_encode($array);
    $conn->close();
  }

  public function getChannelsDetails($userid)
  {
    $database_connection = new DatabaseConnection();
    $conn = $database_connection->getConnection();
    $userid=mysqli_real_escape_string($conn,$userid);
    $sql_service = new SqlService();
    $channelsDetailsQuery = $sql_service->getChannels($userid);
    $result = $conn->query($channelsDetailsQuery);


    if ($result->num_rows > 0) {

        while($row = $result->fetch_assoc()) {
            $array[]= $row;
        }
    } else {
        return 'fail';
    }
    return json_encode($array);
    $conn->close();
```

```
verticle_navbar_HP
```

- ➔ **Grouping the Messages:** We have grouped all the messages from a person sent within a minute back to back and they have a single header. All the messages are grouped together under a date divider**.**

Frameworks Used : We have used a CSS framework called Bootstrap mainly for responsiveness, which we will have to master yet progressively.

CSS stands for cascading style sheets, mainly decides how a DOM Elements appears visually, CSS style can be written Inline styles (Written on HTML Element itself using style attribute), Embedded style (Written with in the same document itself using <style> tag) and External (Linked as external file using <link> tag).

A Dom Element is selected to apply styling in different ways by Tag Name, by Class Name and by ID attribute in the case of Embedded and External Styling.

The rule of thumb to make a webpage look responsive without using any CSS Framework is to give width and height of an element in percentages(%). The child elements automatically inherit properties from its parent.

Why BootStrap:

- Core functionality on which bootstrap is built is responsive 12 column grid structure, layers and components.
- Styling of every single element follows a consistent theme.
- Scales to view properly on different size devices (Avoid writing Medium queries to an extent)
- Rather than coding from scratch, Bootstrap enables you to utilize ready made blocks of code, and can be Customizable.
- Offers class for the ready made styling for most of the HTML Elements like Forms,Buttons,Tables etc.
- Offers ready made utility classes like pull-right , pull-left, hide.
- Lastly, a very good documentation and abundant help on web is available.

An attempt to Elaborate more on Bootstrap Grids:

The Bootstrap grid comes in 4 tiers namely :

1. Extra small (for smartphones **.col-xs-***)
2. Small (for tablets **.col-sm-***)
3. Medium (for laptops **.col-md-***)
4. Large (for laptops/desktops **.col-lg-***).

These grid sizes enable us to control grid behavior on different widths. These different tiers are controlled by CSS media queries internally in Bootstrap. In a Bootstrap's 12-column grid col-sm-3 is 3 of 12 columns wide (25%) on a typical small device width (> 768 pixels), col-md-3 is 3 of 12 columns wide (25%) on a typical medium device width (> 992 pixels). The smallest tier set (xs, sm or md) also defines the size for larger screen widths.

As an example, the following element <div class=”col-sm-6 col-lg-4”’> </div> occupies 50% of the width on small screen devices and occupies 33% on the large screen devices.

References

1. https://www.w3schools.com/php/
2. http://php.net/manual/en/function.mysql-connect.php
3. https://stackoverflow.com/
4. https://www.w3schools.com/bootstrap/bootstrap_grid_basic.asp
5. https://www.w3schools.com/bootstrap/bootstrap_forms_inputs.asp
6. https://www.w3schools.com/icons/bootstrap_icons_glyphicons.asp
7. Login page design Inspiration https://blog.kissmetrics.com/wp-content/uploads/2012/03/hunch-sign-up-page.png
8. Profile Picture Inspiration https://demo.rocket.chat/channel/general
9. https://www.phpmyadmin.net/
10. http://www.writephponline.com/
11. http://jsonviewer.stack.hu/
12. https://slack.com/