Adam Bucknell and Steven Leighton
COS 480
09 May 2016

Project Forum Project Report

Project Goals

The goal of this project was to create a web forum that implements basic features such as the following: users can view and make posts, comment on posts, like posts and comments, and subscribe to posts. We originally wanted to have it run on a Linux box, however we ran into problems when the webpage was trying to access the models that were located in a different directory, and couldn't figure out how to solve the problem. We did all of our tests running the website locally with XAMPP and LAMP.

Database Model

We used a relational database model using a MySQL database. The database consists of four tables:

- user - A record from this table describes a unique user account for the website.
    - userID - The primary key of this table, giving each user a unique identification number.
    - last_login - A timestamp recording when the user last signed in.
    - user_type - A enum type that denotes the type of the user, either 'MEMBER' or 'MODERATOR'
    - date_joined - timestamp recording when the user created their account
    - email - A unique email for the given account.
    - username - A unique username used to log into the account.
    - nickname - A name that the user can use as a display name that is separate from
    - password - A password for the user
- topic
    - topicID - primary key, giving each topic a unique identification number
    - name - A label for the given topic
- post
    - postID - primary key, giving each post a unique identification number
    - title - the title displayed when users are looking through multiple posts
    - content - the main content of the post, allowing users to have more detail in what they want to share; not as limited as the title
    - created_date - timestamp recording when the post was created
    - ups - records the number of users who upvote the given post
    - downs - records the number of users who downvote the given post
    - tags - keywords that can be used when searching for the post

- comment
    - commentID - primary key, giving each comment a unique identification number
    - comment_text - the text that the comment contains
    - comment_date - timestamp recording when the comment was created

We also have the following tables to hold relations between multiple tables, containing only the ids from each of the tables involved:

- user_post - matches a post to the user that created it
- user_like - keeps track of the posts that a user upvotes
- user_dislike - keeps track of the posts that a user downvotes
- user_comment - keeps track of the comments that the user creates
- user_like_comment - keeps track of the comments that users upvote
- user_dislike_comment - keeps track of the comments that users downvote
- user_subscribe - keeps track of the posts that users subscribe to, very similar to upvoting
- user_moderator - keeps track of the topics that a user is a moderator of
- topic_post - keeps track of the posts within a topic
- post_comment - keeps track of the comments that were created for a given post

Rather than explicitly giving foreign key constraints to each of the relations, we figured that because we are using the website itself to fill the tables, that they were unnecessary and better off as de facto constraints in the database and restricted in the PHP code instead. An EER diagram of the database is included at the end of this document.

User Interface

We created static versions of the web pages before worrying about any of the dynamic portions, focusing only on the locations of content and thinking of what users can implement on each page. The entire design of the website was created using HTML and CSS. We wanted to make it as simple as possible, though still have a somewhat "professional" look to it. The background images that we used are from www.allfreebackgrounds.com. When creating some of the CSS files we used www.w3schools.com for reference, as well as other tutorial sites for designing the text-fields and buttons.

When designing the website, we wanted to make something as minimalist as possible so new users wouldn't have to search very hard to find what they were looking for. Thus we included the login fields in the navigation bar, as well as the search bar. We also wanted something familiar to users so it wouldn't be difficult to learn. The final design point that we aimed for was to make pages that were reusable/exchangeable. For example, we used the same table designs and button designs throughout the website. We also used the same page for updating a user's credentials and registering for an account. This makes it easier for us developers to spend less time making the static pages and focus more on their dynamic aspects.

For the most part everything came together very easily and nicely, only needing to add offsets when positioning buttons with text. The most trouble we had though was making the scrollable tables that are on the page “view-account.php”. We couldn’t get the row column sizes to match the header, and when we didn’t manually give them a size, the rows were half the length of the table, and shifted to the left side of the page. From there we figured that if we made the columns the same size as the header columns that it would be fine, but this didn’t work either, still making the rows smaller than the table size. We figured this problem had to do with the header row itself, as each column would scale to fit the text inside it, plus the additional padding. For some reason if we changed the size of one column, it would affect all of the others. So we ended up playing with the row sizes individually in each table to try to get a best fit. The browser that was used for this was Safari, and when we got it to match, we opened it in Firefox on the same machine, and realized the row sizes didn’t match in its environment. Thus we ended up leaving it for what it was, since neither of us are specialists in CSS anyways.

Implementation

The dynamic aspects of the website are implemented in PHP. The idea was that we should be able to fill the database entirely with the website, essentially starting with empty tables. All SQL done within the models of the project. They are run by calling functions instead of having SQL calls within HTML lines of the code. There were many different SQL commands but the general functions are: Create, Modify, Delete, and Get functions.

Models:

Topics:

The broad subject where posts can be made. Users can be moderators of topics.

Posts:

The bread and butter of this project. All posts are visible to anyone going to the site and can be rated good or bad and be commented on to allow collaboration by other users.

Comments:

The essential tool for collaborating on a post in a given topic. These can be voted good or bad as well as a post.

Users:

They are your everyday people, the average joes, the professors and experts, amateurs, anyone who comes across this site. They can make a post, comment, vote on comments/posts, subscribe to posts (to come back to later if they want), view others and their own account, edit their account information, and potentially moderate certain topics.

Features:

Registering:

Allows a new user to post/comment/upvote/downvote/subscribe.

Login/Logout:

Users can control being logged in or not.

Account view:

A logged in user can go to their account and see their posts/comments/ subscriptions and delete them and go to each respective post easily. Users may also view other users accounts but don't have the same ability of removing their items.

Submitting A Post:

Users can submit a post with a title and some content attached. This post shows up under whichever topic they chose to post under.

Commenting:

Users can make a comment on posts, which is their tool to communicate to the author of the post.

Upvoting:

Users that like a particular post OR comment may give it an upvote. This provides other users to see what other people think about a certain post/comment. It's a tool for the user, besides explicitly commenting, that they like something.

Downvoting:

Similar to upvoting but more towards disliking an idea mentioned.

Subscribing:

When viewing a post, a user can subscribe to it and see that subscription on their personal account page so they can come back without filtering through all posts.

Sorting:

All topics/posts/comments can be sorted with some respective filtering options.

- Topics:
  - Topic Name (Asc)
  - Topic Name (Desc)
  - Most Posts
  - Most Comments
  - TopicID
- Posts:
  - Post Name (Asc)
  - Post Name (Desc)
  - Most Upvotes
  - Most Comments
  - PostID
  - Most Recent

Comments:

Most Recent
Oldest
User (organizes comments by username)
Most Upvotes
Most Downvotes

Searching:

The site can be searched with the given field. The results are found by taking the string given and searching through the post titles first (for relevancy purposes) then post content. For future applications this should have a filtering option like above.

We also had two types of users, a standard user and a moderator, where the moderator has the ability to remove and edit posts and comments within one or more topics. However, we ran out of time to implement any of the moderator functionality.

Though it wasn't planned, we did notice that when a user fills in text that they can add any html script that will be displayed when it is shown on the site. The nice thing is that this allows users to post images and videos, however, this could potentially be a problem for the website because there are no limits.

Future Plans

- Tags that the writer of a post can use so it will be easier for users to search for.
- Get it to run on the actual web server.
- More functionality for moderators. Since moderators are limited to specific topic too, we could add another user type such as 'ADMIN', that gives the user permission to control all topics, and possibly other functions like blocking/removing users.
- Fix the scrollable tables on the account page, most likely using Javascript since CSS is limited in this aspect. We'd also like to redesign some of the tables to be more appealing, and change it so we aren't loading every record from the database every time a topic or post table is loaded.
- Add a user friendly way for users to embed images and videos in posts and comments, as well as adding maximum character limits and image dimensions. We'd also make sure that users can't embed items into specific parts of the forum, such as titles.
- Implement higher level searching: searching for posts from a specific user if they can't find a link to their profile. Another example would be better matching of strings, with priority to find a "best match" feature. Currently we do have some priority, but the search function we are using is still pretty simple and very limited.

An EER diagram of our database: