SUMMARY DOCUMENT: ‘We-Chat Discussion Forum’

Joss and Aryan

https://cosc360.ok.ubc.ca/joss/WeChat-Discussion-Forum/home.php

ADMIN LOGIN: username: admin24 password: Test1234 email: admin24@gmail.com

REG LOGIN: username: red password: Test1234 email: red@gmail.com

phpMyAdmin login: 80772049 for username and password

Functionality Implemented:

- Website Deployment
    - Website deployed on cosc360.ok.ubc.ca server. Can by accessed by https://cosc360.ok.ubc.ca/joss/WeChat-Discussion-Forum/home.php
- User Login/Sign-up page
    - The user is prompted to create an account filling out the following credentials: Username, email, password, profile picture and confirm-password.
    - When the user goes on the website, they are able to view the posts, communities, and etc but are not able to create a post, comment or join a community. They must have an account.
    - Once a user account is created, their information is stored into the database and then they are able to login. Remember, information/credentials must be correct and valid or error messages will be displayed such as “Incorrect password, email, etc” when a user tries to login.
- Logout page
    - User can logout of their account and are prompted to the login page where they can log in again OR go to home page to view the content but not interact with anything.
- Database Functionality

    - Database is complete with the necessary tables included for retrieving and storing user website data. Discussiondatabase.sql file included in the repo.
- Home Page
    - Logged in user or without registering is able to filter searches by searching for a post, user, or community keyword. Logged in User is also able to create a community and/or join one that is already made by other users. The top communities are listed and ranked by the amount of users joined. The downvotes can also go in the negatives if the comment/post is not liked by others.
    - Logged in User is also able to create a post by entering a title, which community it relates to, the description and possibly an image (future iteration).
    - Logged in User can decide to leave a comment on a post as well which stacks along with the other user's comments.
- View Account/User page
    - This is where the user can see the communities they are in currently, when they joined the website (made an account), and view their previous posts. They will also see their profile picture.
- Settings
    - User is able to change their password. User has to enter old password, new password, and then confirm password. If the password does not match previous password, an error shows up and asks to retry as it does not match the database stored values. User must fill out all the fields before saving.
- Email forgot password
    - Forgot password implementation sends user code and expiry time to the database. Code also sent to the user email. User can start over if the wrong email is entered or if the email is not in the database.
- Create/Join Community
    - User is able to create a community (community name, description, and create button).

    - User is able to join previous communities made by other users. The community is then displayed on the home page and the user page if they view their account. If a user is not logged in, the community sidebar will tell them to login to join communities

- Admin Authentication/functionality
    - ‘isAdmin’ column in ‘users’ table in the db. Type is boolean and default value is 0 for not admin, manually changed in database to 1 if given admin authentication.
    - Admin is able to search for user by email, name, or post. Can search for posts, communities and users in general posts column, as well as can search for specific users in their user search sidebar. Admin can press garbage icon to delete a post or user
    - Admin is able to delete a user, this will delete all posts by that user, comments, and their communities joined. If the user has not posted, commented, or joined a community their record will be permanently erased. If the user has posted, their actual account will remain in the database. This is to allow for reactivation in the future, however this can be changed to completely remove the users record regardless of previous post/comment history.
- Asynchronous updates
    - Use .AJAX, JavaScript, and JQuery, along with JSON (exchanging data from client and server) to enable the browser to send requests to the server in the background without requiring a full page refresh on upvotes and downvotes (ongoing)
    - When the new data is retrieved from the server database, the JavaScript code updates the relevant parts of the page for a seamless and efficient user experience
        - Real time comment updates using AJAX requests, upvoting and downvoting vote count update in real time, user notifications.
- Server side security and Client Side
    - Required parameters (password, email, user etc)
    - Secure password storage

        - Hashed passwords
    - SQL injection attack prevention
        - Input validation through password (specific length, requirements)
    - Database priviliges
        - Limit priviliges to only admins that are manually given the authentication. Prevents hackers from modifying sensitive data.
    - Prepared statements

Limitations (to be fixed for final iteration):

Upvote comment functionality has a bug where if a user tries to vote the score changes displays as undefined. Unregistered users are also able to vote

Asynchronous update on upvote/downvote functionality in progress, doesn't show for other users when upvotes are made (need to refresh)

When clicking on a username, the site should redirect you to that account page

No picture upload for posts functionality yet

User cannot sort posts by popularity (filter button)

Admin can only delete whole posts instead of specific comments

Admin cannot delete communities

Unable to leave community

Should be able to upvote from post page not just home page

When posting or creating community, should redirect to home page for better flow

Database

Store the user information upon account creation

- Name, login details (hashed password, email, user id)
- Posts
- Date and Time

- Communities
- Upvote/downvote score

Store users and validate upon login

- Once user logs in and is validated, they can view their account
- Posts and comments from this user will be added to their account

Store posts

- Posts include text content, user, community, image(optional), upvotes, and comments
- Retrieved on view account page

Store communities

- communities have to store their corresponding posts
- Store their username/id that are registered to that community
- Retrieved on view account page

Store comments

- comments stored in database

Store forgot password

- Store expiry time and verification code and email in users table for forgot password verification details

Store Admin → 0 for not admin, 1 → admin

- manually alter in database 0 is default if user not admin, 1 if user admin, thus given admin privileges.

General Functionality Roadmap

- ✓ ~~Posted on cosc360.ok.ubc.ca~~
- ✓ ~~Discussion thread storage in database~~
- ✓ ~~Asynchronous updates~~

- [x] ~~Database functionality complete~~
- [x] ~~Browse discussions without registering~~
- [x] ~~Allow user login by providing user id and password~~
- [x] ~~Create and comment (specific for each project) when logged into the site~~
- [x] ~~Server-side implementation complete~~
- [x] ~~Client-side security~~
- [x] ~~Server-side security~~
- [x] ~~Preliminary summary document, indicating implemented functionality~~
- [x] ~~Search for items/posts by keyword without registering~~
- [x] ~~Register at the site by providing their name, e-mail and image-~~
- [x] ~~-image for registration~~
- [x] ~~Users are required to be able to view/edit their profile~~
- [x] ~~User password recovery (via email)~~
- [x] ~~ADMIN: Search for user by name, email or post~~
- [x] ~~ADMIN: Enable/disable users~~
- [x] ~~ADMIN: Edit/remove posts items or complete posts~~