Description of Implementation

Header (header.php)

The header includes the logo and navigation menu. The navigation menu is generated dynamically using PHP session variables set on login. If the user is logged in, their username is displayed in the menu and linked to their profile page. If the user is not logged in, a link to the login page is provided instead. If the user is an admin then the menu displays links to the admin specific pages. CSS files are linked for styling. The header file starts the session for all pages using session variables as it is included on each page.

Login/Signup (login.php, signup.php, banned.php)

The login page features a generic login form. This form is validated first with js with the loginVerification.js file to make sure the form is not empty. Users can enter either their username or email and then their password to login. User info is used to query the users database, if their information returns a user then they will be logged in. If not they are notified that they're provided information is invalid. The query also checks whether they are an admin and that their account is enabled. Session variables are set based on these values. The username session variable is also set. Successfully logged in users are directed to the threads page, admin are directed to the manage threads page and disabled users are directed to a page that alerts them that they are banned.

The signup page is where users can choose a username and enter their email and password. The form is validated using javascript in the signupValidation.js file. This file checkes that no fields are left empty, both passwords match, that the password is at least 8 characters, that the email follows the right format and the imagge file provided is an image file. Once the information is provided and they click signup a post request is sent to the php in the file where the username and email provided is used to query the users database to make sure there is not already a user with the same credentials. If a user is found then the user will be alerted that the username or email is already taken and the form is not submitted. If the credentials are unique another query is run to enter them into the users databse and then are redirected to the login page to login. For the profile image, When the page receives a POST request it checks to see if an image was uploaded. If so it saves the image on the server in the userImages folder with the filename being the user's username. Anytime a user image is displayed on the site, a page will search the userImages folder for a filename matching the user's username and then display that image. If no image is found then a default image is displayed.

Profile (profile.php, editProfile.php)

The profile page is written to display information about a user while also implementing signout and edit profile functionality. To view a user profile the page is accessed via a GET request with a username specified in the url. When the page receives a GET it checks to see if a user is specified in the url and otherwise defaults to the currently signed in user. The page searches for the user in the database and displays and error message if the user does not exist. Otherwise it displays information from the user including username, user image, bio and location. It also querries the database to calulate the users total post count and to generate a list of recent threads created by the user which it displays and links to. The page also checks the session variable to see if the user is signed in as an admin, and if so displays a button which allows the user to be disabled/enabled via a request to changeState.php which redirects back to the

profile page. If the profile page user matches the session variable for the currently signed in user the page also displays displays a link to the editProfile.php page which it accesses via a GET request. When accessed via GET the editProfile page accesses the user information from the database for the currently signed in user and then displays a form which the user can use to edit their bio and location and upload a new profile image. The form then submits to editProfile.php via POST with the image upload being validated via javascript first. When editProfile.php receives a post request it update the users information in the database based on the provided data. If an image was uploaded the page deletes any current image for the user from the server and then replaces it with the uploaded one. After updating information the page redirects back to the profile page. Finally, if the profile page user matches the currently signed in user a logout button is displayed which sends a POST request to profile.php. When profile.php receives a POST request it resets all session variables and then redirects to the login page.

Main threads page (threads.php, getThreads.php, createThread.php, deleteThread.php)

This page (threads.php) acts as the main page for the site and is implemented asyncronously via JQuerry and AJAX. The page uses a javascript function called updateThreads which asycronously sets POST requests to getThreads.php which returns a list of threads from the database via JSON which are rendered on the page using jQuerry. When requsting threads, a region category (or all), a search term, and a page number are specified. getThreads.php querrys for only pages which match the search term and region category and uses the page number to return a set of 10 threads ordered by most recent update (page 0 = first 10, page 1 = next 10, ect.). First, previous, next and last buttons are shown or hidden on the page when relevent to allow the user to traverse though pages of results. getThreads returns whether the are more results to be shown in order to implement button show/hide logic. updateThreads is called anytime the page number, search term, or region chages based on user input. getThreads() is also called on a 10 second interval if the usder is on the first page of results, meaning that if another user creates a new thread, it will show up automatically. In this case the page only updates if there is new content to show in order to prevent flickering. Thread titles each link to their individual thread page and usernames/images link to coresponding user profile pages. If a user is signed in as indicated by a session variable the create a thread button is shown which links to createThread.php. createThread.php displays a form which allows a user to create a thread. The form is validated via javascript and then submitted back to createThread.php via a POST request. When createThread.php recieves a POST request it adds a thread to the database with the given information and then redirects back to threads.php. If an admin user is signed in, threads.php adds a delete button for each thread. This button asyncronously sets a POST request to deleteThread.php which deletes the thread from the database and then calls updateThreads() in order to reflect the change on the page. threads.php also checks to see if region is set in the GET request which enables other pages to link directly to the threads page with only a specific region selected. If no region is set it defaults to all regions.

Viewing a specific thread page (thread.php, getReplies.php, createReply.php, deleteReply.php, getQuote.php)

thread.php works in a very similar way to threads.php. It displays the content of a thread along with a list of the thread's replies in order 10 at a time. It uses a function updateThread() which asycronously requests getThreadContent.php via POST and then updates the list of replys. It also asyncronously updates replies every 10 seconds so that new replies from other users show up automatically much like the threads page. The reply button works with createReply.php in the same way as createThread.php and the flow of requesting replies and displaing navigation buttons works the same way as threads.php with

getThreads.php. There are also reply delete buttons which asyncronously request deleteReply.php in the same way that threads are deleted. The differences between the pages are that thread.php does not have any search functionality within a thread, thread replies and thread replies implement a quote system. The quote sysyem allows users to reference other replies in their reply via quotes. Each reply has a checkbox to sepcify if a user wants to imbed a quote of that reply in their reply. When a request is sent to createReply via POST an array of all checked reply id's is sent. embed tags for the quoted replies are then added automatically to the reply textbox. The reply embed tags are then stored within the text of the reply in the database. When replies are rendered via jQuerry, the reply text is parsed for any quote embed tags. The tags are then replaced in the reply content with a div which contains the content of the quoted reply which is accessed from the database via another asycronous POST request to getQuote.php. This process is recursive so if the quoted reply also includes quotes, those are rendered as well.

Admin (threads.php, admin.php, stats.php, changeState.php)

Admin users are must be inserted directly into the database, there isAdmin value is set to 1. When an admin user is logged in their navigation bar will have options for 'Manage Threads', 'Mange Users' and 'Analytics'. The manage threads page still uses the threads.php file, it uses the isAdmin session variable to display a delete button which when pressed will remove that thread from the database. The thread page does the same thing for replies, if the user in the session is an admin the delete button is displayed on each reply and will remove that reply from the database when pressed aswell. The Manage users page or admin.php file, this page queries each user from the users table in the database and displays their info in a table. This query is adjusted based on the input of the search form so admin can search for specific users based on their username or email. The search form must be submitted to requery the results. In the last row beside each user that is not an admin there is a form with a disable/enabled button. When clicked the form sends the user's username and their isEnabled value to the changeState.php file via a POST request. This file changes the users enabled value in the database. The button acts as a switch and displays enable or disable based on their current status.