



JoLa™

Discussion forums Relmaged, ReDiscussed



Final Report

UBC 360: Web Programming

Professor Ifeoma Adaji

Team Members

Abdulaziz Almutlaq - 79960175

Onwuvuche Jordan Chukwuka - 61007530

Walk Through the Site

1. Home Page

This is JoLa's homepage, which allows unregistered users to browse the most popular posts without logging in. However, unregistered users are limited in their access to the site. Clicking on a post will redirect them to the login page, preventing them from viewing all comments. They are also unable to create a new thread without logging in. Registered users have access to all posts and can use the menu buttons to search through threads. The 'Prime' Threads section on the right displays the threads with the greatest number of post engagements. To sign up or login, users click the "Sign in" button in the top right corner on the navigation bar.

2. Login Page

Upon clicking the "Sign in" button on the home page, the user is redirected to the login page. In case the user has an existing account, they can enter their credentials and proceed to click the "Login" button. Alternatively, if the user does not have an account, they must click on the "Sign Up" button to proceed.

In the unfortunate event that the user has forgotten their password, they have the option of clicking on the 'Sort it Out!' link below the "Login" and "Sign Up" button options, to figure out a way to get into their accounts again.

3. Register Page

Once the user clicks on the "Sign Up" button on the Login page, they will be redirected to the registration page. Here, they will be prompted to enter their desired username and then proceed to the next step by clicking the "Next Step" button. On the subsequent page, they will be required to provide their email, username, and password, in steps. If the user accidentally navigates to the next steps, which are on different pages, they can easily return to the previous page by clicking the "Previous Step" button.

Once a user has completed the registration process, they are required to verify their account via email. **Upon** successful verification, the user can click on the "TAKE ME BACK" button to be directed to the login page. Then, they can input their registered email and password to log into the website. Upon successful login, the user's username and profile picture will appear on the navigation bar, indicating that they are logged in.

The user will be notified that their "Passwords don't match" if that is the case.

If the user tries to login before verifying their email, they will be notified that their "Email is not verified."

4. Restore Account Page

In case the user cannot remember their login details, they can click on the "Sort it Out!" link on the login page. From there, they will be taken to the Restore Account Page where they can enter the email address they used to sign up and initiate the password reset process.

5. Create Thread Page

When the user clicks on the "Discuss Something New" button on the Home Page, they will be redirected to the "Thread Create" page where they fill in the fields required. After filling in the required fields, they can create the thread by clicking the "Post Your Thread" button. Upon successful creation of the thread, they will be redirected to the newly created thread.

6. Thread Posts Page

This is a thread page where posts belonging to various threads are displayed. Users have the option to either join or leave the thread, and the buttons will update in real-time without requiring a page refresh to reflect votes up or down, for the posts and the comments too.

- There is a "Prime Profiles" tab, which displays the top five users' profiles with the most posts in a specific thread.
- Users can vote on a post without having to access the post itself.
- The posts can be sorted based on either "Leading" or "Recent." Sorting by "Leading" will display posts with the most votes at the top, while sorting by "Recent" will arrange them based on their creation time in ascending order. This sorting feature is dynamic meaning that it will reflect current statistics.
- The post owner or an admin can choose to "Hide" or "Remove" a post without having to access it. Removing a post will delete it from the thread, whereas hiding a post will disable it, preventing any comments or votes being made on it. However, the post will still be visible to users. It is also possible to hide or delete a post from within the post itself.
- In the page images, we can see comments on the thread post. These comments can be voted on as well without having to enter the post. And of course, the admin, and the post owner can delete any comments on this post without entering it.

7. Create Post Page

To initiate a post on a specific thread, users must click the "Discuss Something New" button located on the thread page. To successfully create a post, users need to provide a valid title and fill in any other piece of information which could be the Post Body, a Post Image, and a YouTube URL. Once all the required fields have been filled, users can click the "Create Post" button. This action will direct users to the thread where the post resides, and they can see their newly created post.

8. Posts Page

On this page, users have the ability to vote and comment on posts and the comments on the post. This is demonstrated in the images. Comments are sorted chronologically from newest to oldest. Clicking on the sub-thread below the thread title redirects the user to the main sub-thread page. Post owners and admins can delete or hide posts and comments. Comments and posts are dynamically updated within seconds.

Posts which are disabled by the administrator can't be voted on or commented on. This feature was created for the Admin in the event of the Violation of Community Guidelines.

If the post owner or administrator decides to delete a post from the Post page instead of the Threads page, the user will be redirected back to the main thread page after the post is deleted.

9. User Notifications Page

Upon clicking the megaphone bullhorn icon on the notification bar, users will be able to view various activities occurring within their associated active sub-threads.

10. Account Page

No matter where one is on the website, one can always click one's username on the navigation bar to access this Account Page. The account page displays all posts associated with the user, including those they have created or commented on. One can also access their profile settings from this page through clicking on the "Profile Configuration" button under the Profile Image container.

11. Settings Page

Upon clicking the "Profile Configuration" button on the Account Page, one can access this page which allows them to modify their profile by updating their profile picture, changing their username or password, or even deactivating their account.

12. Search Page

To easily access this page, click the "JoLa" Home logo on the navigation bar, click on "Threads" in the left-hand side menu bar, and finally click on "Search Something" in the same menu bar. The Search page allows the user to conduct a search for specific posts, threads, or comment reactions.

ADMIN AND USER LOGIN DETAILS

a. Admin:

- Email:
 - Password:
- b. **User (Non-admin):**
- Email:
 - Password:

13. Admin Main Page

To easily access this page, click on the “JoLa” logo on the navigation bar, and then click the “Mgmt Portal” option on the left-hand side menu bar. This option will only be visible if the account is an admin account. This page displays various metrics and trending posts to the administrators. It's worth noting that the administrators now have access to a **new menu bar – Mgmt Portal**, which allows them to navigate to the homepage or access additional functionalities related to users and threads.

14. Admin Users Page

To easily access this page as an admin, click on the “JoLa” logo on the navigation bar, and then click the “Users” option on the left-hand side menu bar. On this page, administrators have the ability to search for specific users or browse through the entire user list. Clicking on a username will lead to that user's profile. Additionally, the admin can view the email confirmation status and admin status of each user. The admin also has the power to promote or demote a user to or from an administrator role and can also block or unblock user accounts.

15. Admin Threads Page

To easily access this page as an admin, click on the “JoLa” logo on the navigation bar, and then click the “Threads” option on the left-hand side menu bar. This page is specifically designed for admins, who can search for a particular thread by entering its name in the search bar or scroll through the list of threads. By clicking on the thread URL, they can inspect the thread and view details such as who created the thread, whether the thread is active or hidden, and the number of members who have joined the thread. The admins can perform several actions on the thread, including hiding, restoring, or deleting it, based on their preferences.

Implementation From a System Perspective

Outline of Implemented Features

A. Non Logged In Users

- Users are able to view posts directly on the main page.
- Users can retrieve their passwords via email by selecting the "Get Help Sign in" option from the login page.

- Users will have access to the same styling as a registered user.
- Users have the ability to search for posts, threads, and comments by clicking on any of the left-hand side menu bar options, and then using the search bar.
- Users can create an account by clicking on the "Sign-in" button located on the top-right of the main page's navigation bar.

B. Logged In Users

- Users can create threads with custom background images and profile pictures.
- Users can create posts in any thread with options to include body text, images, or YouTube URLs.
- Users can vote on both comments and posts made by other users.
- Users can add comments to any post within a thread.
- Users have the ability to hide or delete their own posts, as well as delete comments from their posts.
- Users can update their profile pictures, usernames, passwords, and email addresses.
- Users can use the search bar, available on the left-hand side menu, to search for posts, threads, and comments.
- Users can log in by simply entering their email and password.
- Users have the ability to view every comment they have made.
- The session is stored to maintain user state, including staying logged in between page navigations.
- Users receive different alerts:
 - The Admin [**Username**] has deleted your thread [**Thread**].
 - The Admin [**Username**] has deleted your post in thread [**Thread**].
 - [**Username**] liked your post in thread [**Thread**].
 - [**Username**] disliked your post in thread [**Thread**].
 - [**Username**] has posted in your thread [**Thread**].
 - [**Username**] replied to your comment in thread [**Thread**].

C. The Admin

- The Admin has the ability to search for specific users by entering their usernames on the dedicated tab.
- Admins can perform actions such as blocking or unblocking users and granting them admin privileges.
- The admins can hide, restore, or delete threads.
- Admins have the ability to delete any comments or posts as appropriated by Community Guidelines.

Description of JavaScript and PHP Project Files

A. PHP

- **Account-settings:** component in View. The file utilizes the UserController class to present up-to-date user information, and dynamically updates this information using Javascript.
- **Account:** component in View. This file utilizes the UserController class to showcase up-to-date information about the user and their respective threads.
- **Admin-threads:** component in View. This file utilizes the AdminController class to exhibit all the threads present on the website. It incorporates Ajax to enable the dynamic deletion/hiding/restoration of threads on the website. Access to this page is restricted to Administrators only.
- **Admin-users:** component in View. This file uses the AdminController class to show a list of all registered users on the website. It employs Ajax to allow for dynamic deletion, user promotion to admin, or unblocking of users. Access to this page is limited to administrators only.
- **Admin:** component in View. This file utilizes the AdminController class to exhibit website statistics and trending threads. Access to this page and its actions is limited to administrators.
- **Error:** component in View. This file works in conjunction with Router.class.php to display an error message when a page is not found on the website.
- **Index:** component in View. This file manages the rendering of Views, titles, and security checks to ensure that access to certain pages is only granted to authorized users, admins, or guests.
- **Login:** component in View. This page is restricted from access if the user is already authorized to the system. Users cannot log in if their email is unconfirmed or if their account is disabled. The page employs Ajax to validate user input with the server and establish the current session with the correct user credentials.
- **Notifications:** component in View. This file utilizes the NotificationController class to exhibit the activity of other users towards the content created by the current user on the website. The content is exclusively visible to the current user.
- **Post-create:** component in View. This file utilizes the PostMiddleware and PostController classes to facilitate the creation of posts for their corresponding threads. It leverages Ajax and Javascript to interact with databases and perform security checks to ensure information is securely stored in the database or file storage systems such as images.
- **Post:** component in View. This file uses the PostController class to display the details of a single post, along with its respective information in the thread. It utilizes Ajax and Javascript to ensure the security of comments posted in the post and thread. This page is only available for logged-in users, and the post won't be displayed if it has been deleted by the owner or an administrator. Users are not allowed to post comments or vote if the post is marked as "hidden."

- **Register-confirm:** component in View. This page is exclusive to users who possess a unique link containing a "token" attribute in the URL. Users need to enter a randomly generated code on this page to confirm their account registration. During the registration process, the token is randomly generated and saved in the database. The token length fluctuates to protect against brute-force attacks. The security code to access the page is sent to the user's email, along with a secure link to validate the account registration.
- **Register:** component in View. This page enables the user to register a new account. It is only available to users who are not logged in. The page uses Ajax and Javascript to validate user input, ensuring that the fields are not empty and that the username and email are unique and meet the required password format. Once validated, the information is stored in the database, and a confirmation email is sent to the user's email address containing a token and confirmation code. The user must enter this code on the website to confirm the account registration.
- **Restore-confirm:** component in View. This page allows users to reset their account password by entering new and confirmed passwords. Access to this page is restricted if the URL attribute is missing a randomly generated "token" or if the token has expired. The token is generated randomly with different symbols and lengths when the user initiates the "password recovery" process on the website's restore page.
- **Restore:** component in View. The page allows users to initiate the password recovery process by entering their email address. Access to this page is restricted to users who are not currently logged in. If the email address entered by the user is not found in the database or is not verified, the user will not be able to proceed with the password recovery process. Otherwise, a secure link will be sent to the user's verified email address to proceed to the next step of the password recovery process.
- **Search:** component in View. This page is designed to enable users, whether authorized or not, to search for information on the website, including comments, posts, and threads. The server will sanitize the search query to protect against potential attacks before sending it to the Model for execution. Ajax and Javascript are utilized to dynamically validate, and update page content based on the search query and parameters.
- **Thread-create:** component in View. The creation of a thread is facilitated by the ThreadMiddleware and ThreadController classes, which use Ajax and Javascript to communicate with databases and perform security checks before storing information in database or file storage systems such as images.
- **Thread:** component in View. Authorized users can access this page to view details about threads, including associated posts and comments. Users can take various actions on this page, such as joining or leaving a thread, creating posts,

and voting for posts and comments. The page also displays a list of top users for the thread based on the number of posts they have made.

- **Main:** component in View. This page is accessible for all authorized and non-authorized users to show the trending posts and threads. This page is also used as the main page of the website.
- **DatabaseConnector.class:** component in Controller. This class establishes a connection to the database and returns a "Connection" object for other controllers/models to access the database. If the connection is unsuccessful, an error page will be displayed with a message that the website is not working. For security purposes, specific details regarding the error will not be displayed.
- **DotEnv.class:** component in Controller. This class reads the ".env" file to retrieve the database credentials, and then passes this information to the DatabaseConnector class to establish a connection with the database.
- **Router.class:** component in Controller. This class handles routing of all incoming GET requests to specific pages by returning the page title and corresponding file path for rendering.
- **AdminMiddleware.class:** component in Controller. This file class implements validation for user input and directs information to the appropriate middleware method based on the request type (POST/GET). Upon successful validation, the middleware methods communicate with the AdminController Model to perform database operations such as updating, deleting, or creating information. After the operation is complete, the server returns a JSON object with either information or error to the Ajax method.
- **CommentMiddleware.class:** component in Controller. This file class implements validation of user input to ensure its validity and prevent security risks. The middleware directs the input to the appropriate method in the CommentController Model, which is responsible for updating, deleting, or creating information in the database. After the operation is complete, the server returns a JSON object containing either information or an error message to the Ajax method.
- **ThreadMiddleware.class:** component in Controller. This file class performs data validation on user input like that done in the JavaScript file. It then sends the information to the appropriate method in the middleware based on whether the request is a GET or POST. After validation is complete, the server calls the corresponding method in the ThreadController Model to modify, delete, or create information in the database. Finally, the server returns a JSON object with either information or error back to the Ajax method.
- **TokenMiddleware.class:** component in Controller. This file class includes validation to confirm that user input is valid, like the validation in the javascript file. Depending on whether the request is a POST or GET request, the information is sent to a specific method in the middleware. After completing the validations,

the server calls the specific method in the TokenController Model to update, delete, or create information in the database. When the operation is finished, the server sends a JSON object with information or an error back to the Ajax method.

- **UserMiddleware.class:** component in Controller. This file class includes a validation process that verifies the validity of user information passed in the request. Depending on whether the request is a POST or GET, the information is forwarded to a designated middleware method. After validation, the server calls the respective method in the UserController Model to create, update or delete user data in the database. Once the operation is executed, the server returns a JSON object with relevant information or an error message back to the Ajax method.
- **Controller.class:** component in Controller. This is an abstract class that enforces Models to implement specific methods such as get, post, update, and delete to interact with the database.
- **AdminController.class:** component in Model. This controller performs insertion or deletion based on the information received from the AdminMiddleware. It also retrieves information based on method calls from the respective view in the admin pages.
- **CommentController.class:** component in Model. The CommentController class executes insertion/deletion of comments based on the information passed from the CommentMiddleware. It also executes retrieval of information based on method calls from the respective Views in the thread, comment, post, and search pages.
- **NotificationsController.class:** component in Model. This controller processes insertion and deletion based on the user's actions on the website and information passed from various middlewares. It also retrieves information based on method calls from the notification page's respective view.
- **PostController.class:** component in Model. This class handles the insertion and deletion of information based on the data sent by the PostMiddleware. Additionally, it retrieves information through method calls from Views related to threads, comments, posts, and searches.
- **ThreadController.class:** component in Model. This controller performs database insertion/deletion based on the information received from the ThreadMiddleware. It also retrieves information based on method calls from the corresponding Views in the thread/main/admin/search pages.
- **TokenController.class:** component in Model. This controller manages the creation and deletion of user accounts based on information passed from the UserController and TokenMiddleware. It also retrieves and validates information based on method calls from Views in the register-confirm and restore-confirm pages.

- **UserController.class:** component in Model. This controller facilitates insertion and deletion of data by verifying the user's credentials through various middlewares (e.g., email confirmation, admin status, etc.). Additionally, it retrieves data and performs validation operations based on the requests made by the user from Views such as account and account-settings pages.

B. JavaScript

- **Script.js:** This file manages the real-time updating of website pages through the use of JQuery and Ajax.

How The Website Works at a High Level

The codebase is segregated into client-side and server-side components.

A. Client-side

The client-side code aims to create a dynamic and responsive website. Using AJAX, we update styles and information dynamically, avoiding page refreshes while using search bars or sorting threads. Additionally, we utilize RESTful API calls to retrieve the required information from the server. This code also includes CSS style definitions and page creation for the entire website.

B. Server-side

The server side of our codebase is responsible for establishing a connection to our database and facilitating user access and modification of data. To ensure consistency in connecting to the database, we utilize a DatabaseConnector class that is called in our controllers.

Middleware classes were developed to enhance security. Whenever an API call is initiated, it is first routed through a PHP middleware class to verify the validity of the transmitted data. In case the data is invalid, an error is returned. However, if the data is valid, the middleware passes it to the relevant PHP controller.

Our querying is centralized in the PHP controllers. We have implemented various controllers for different functionalities: Admin, Comment, Notification, Post, Thread, Token, User.

The communication with the required database models defined in our database folder is handled by each controller. Additionally, a PHP router class has been created to ensure proper navigation between pages and to set the webpage's title.

Website Limitations

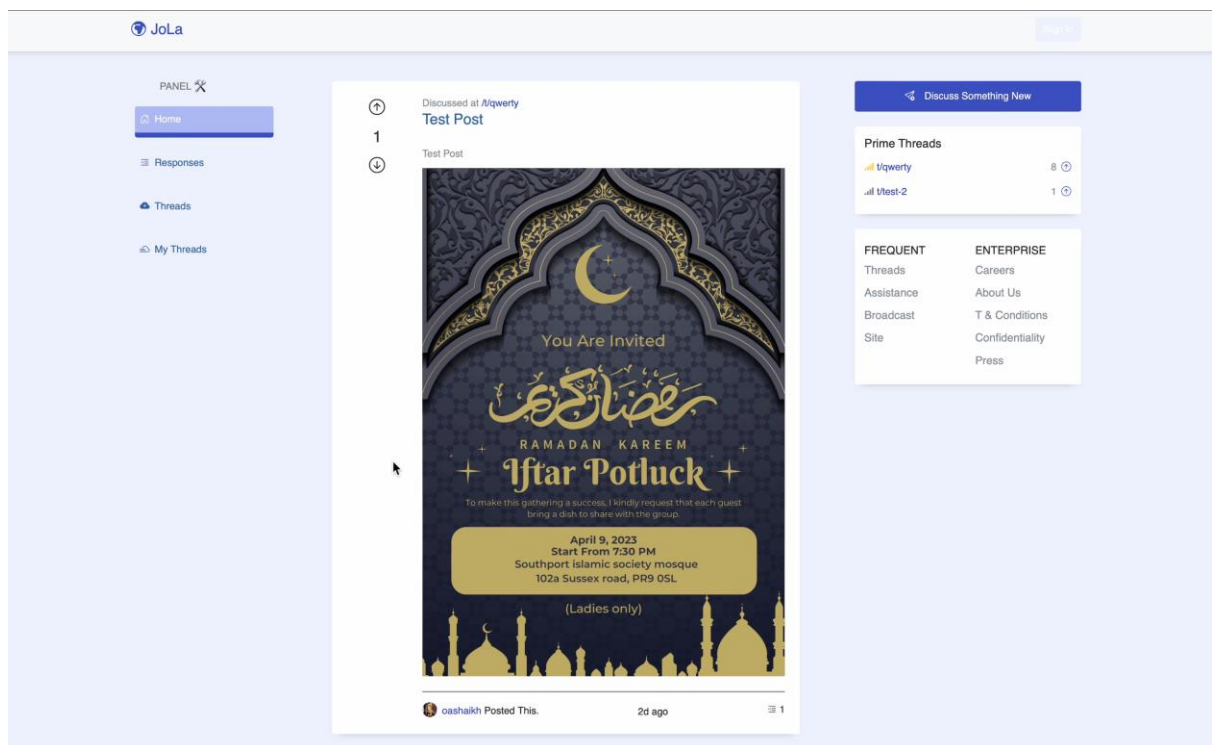
- Due to the unavailability of an SMTP server/service, the system is unable to send an automatic email containing the recovery/register link. Therefore, the

administrator needs to provide the link to the user manually by accessing the database.

- Users do not have the privilege to edit their comments once they are posted.
- The system does not support replying to another user's comment. Users can only comment on a post directly.

Examples of Each Page Type

I. Home Page



II. Login Page

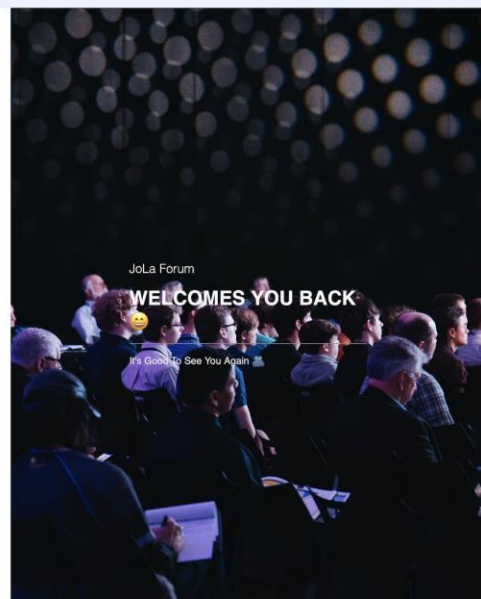
JoLa

EMAIL ADDRESS
aziz@gmail.com

PASSWORD
Aziz's Password

Login Sign Up

Forgot your Password? Sort it Out!



III. Register Page

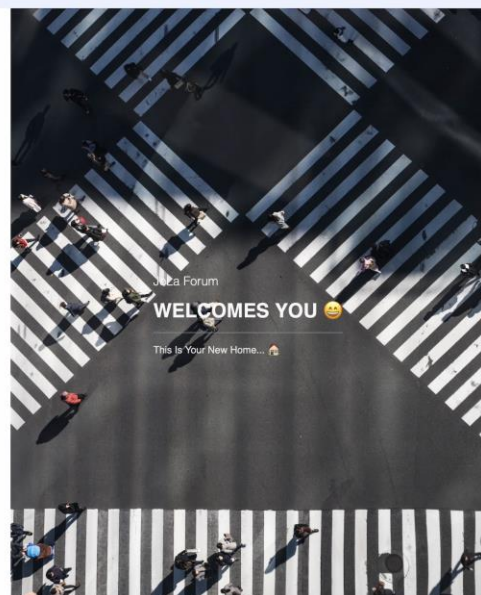
JoLa

Your Quest Is Just Beginning... 🗺️

USERNAME
Fill in Here %

Previous Step Next Step

Username Criteria
Please ensure your username is free of any special characters and does not go beyond 8 characters.





Your Quest Is Just Beginning... 🍷

EMAIL

qwerty@qwertyj

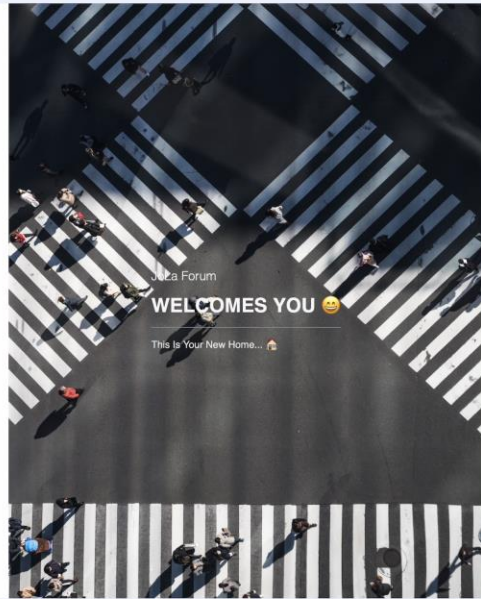
Previous Step

Next Step



Your Email

Our community guidelines say that your email should not be longer than 25 characters.



Your Quest Is Just Beginning... 🍷

PASSWORD

Secret Password



REPEAT SECRET PASSWORD

Repeat Secret Password

Previous Step


Register



Your Password

We recommend to create a password with minimum length of 8 characters, one uppercase letter, one lowercase letter, one special symbol.






Your Quest Is Just Beginning... 🗺️

PASSWORD


REPEAT SECRET PASSWORD

[Previous Step](#) [Register](#)


Your Password

 We recommend to create a password with minimum length of 8 characters, one uppercase letter, one special symbol.

Oops...

 Passwords don't match.



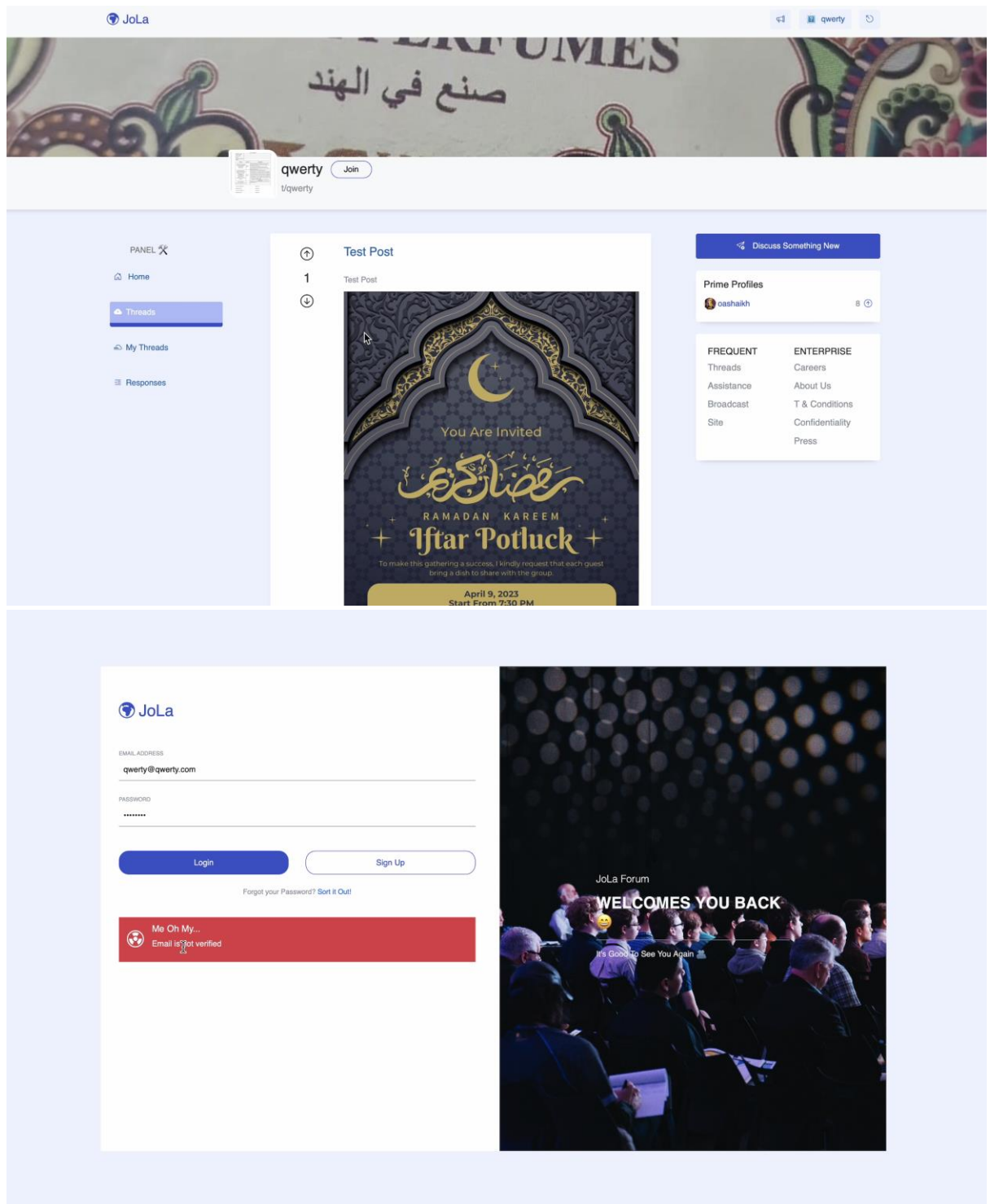


Thanks for Joining Our Community.

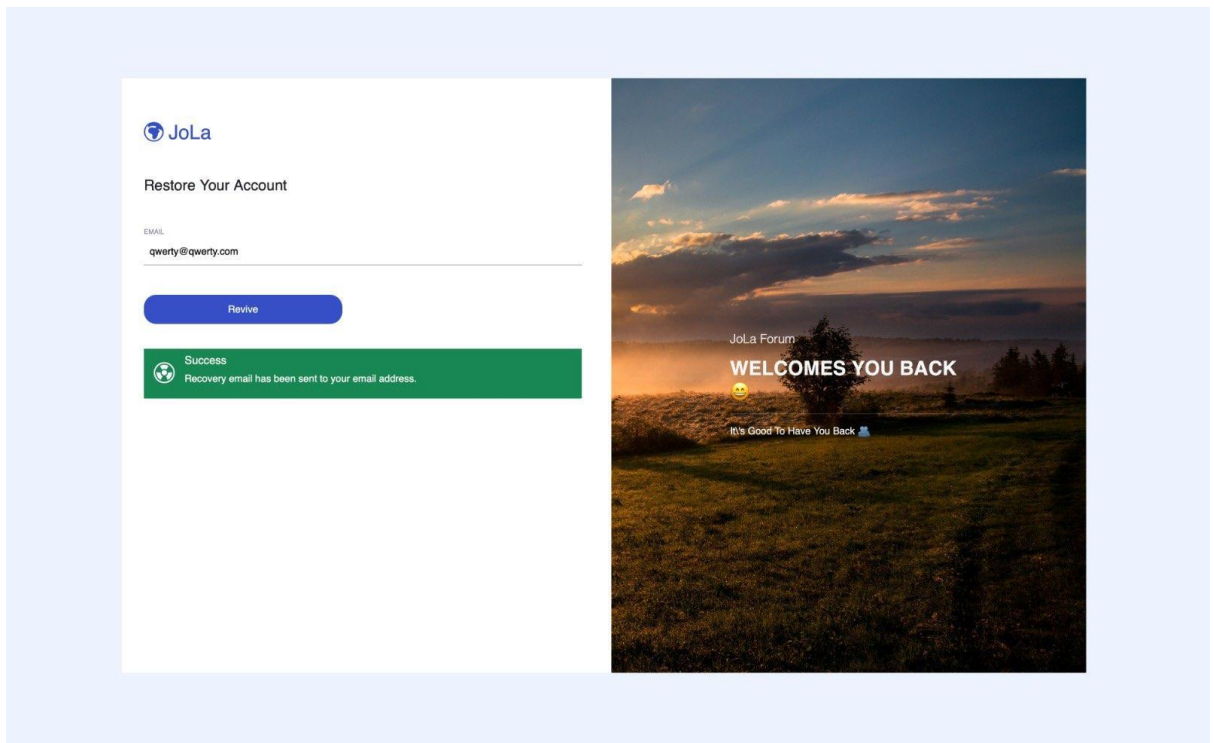
We are almost done. Check your email to verify your account creation.

[TAKE ME BACK](#)





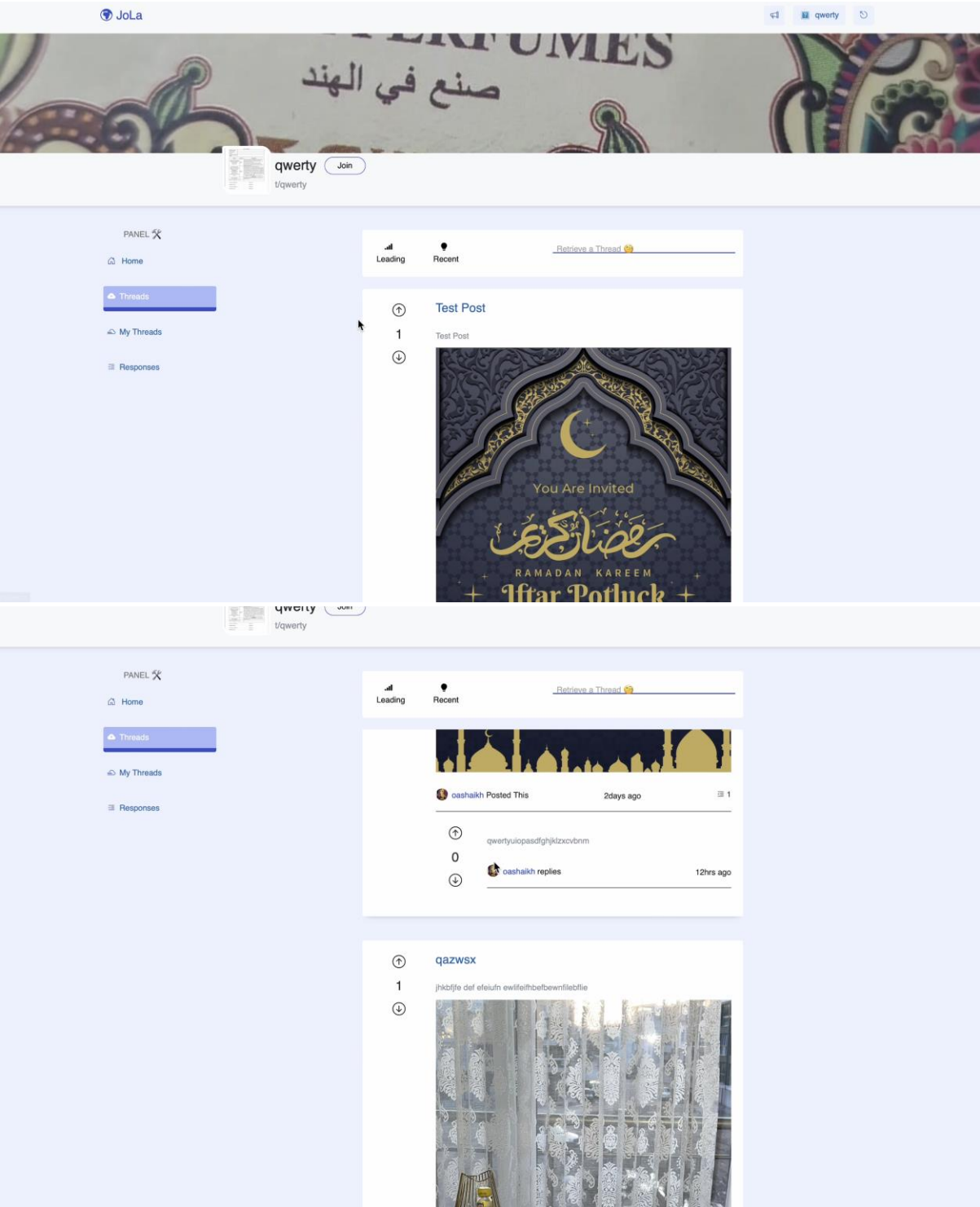
IV. Restore Account Page

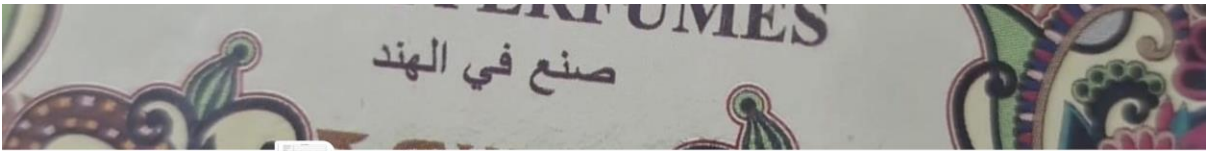


V. Create Thread Page

The image is a screenshot of the 'Start A Thread' page in the JoLa forum. The page has a light blue background. On the left is a sidebar with a 'PANEL' header and a close icon. Below the header are five navigation links: 'Home', 'Mgmt Portal', 'Threads' (which is highlighted with a blue bar), 'My Threads', and 'Responses'. The main content area is titled 'Start A Thread'. Below the title is a section for 'Thread Name' with the instruction 'Enter your unique thread Title'. It contains a text input field with the placeholder 'Fill In Here %' and a 'Link:' label. Below this is a section for 'Select Your Background Image' with the instruction 'This is compulsory.' It contains an 'Upload Image:' label and a 'Choose File' button with the text 'no file selected'. Below this is a section for 'Select Your Profile Image' with the instruction 'This is compulsory.' It also contains an 'Upload Image:' label and a 'Choose File' button with the text 'no file selected'. At the bottom of the form is a blue button labeled 'Post Your Thread'.

VI. Thread Posts Page





qwerty

Join

t/qwerty

PANEL ✕

Home

Threads

My Threads

Responses

Leading

Recent

Retrieve a Thread 🧵



oashaikh Posted This

2days ago

0

①

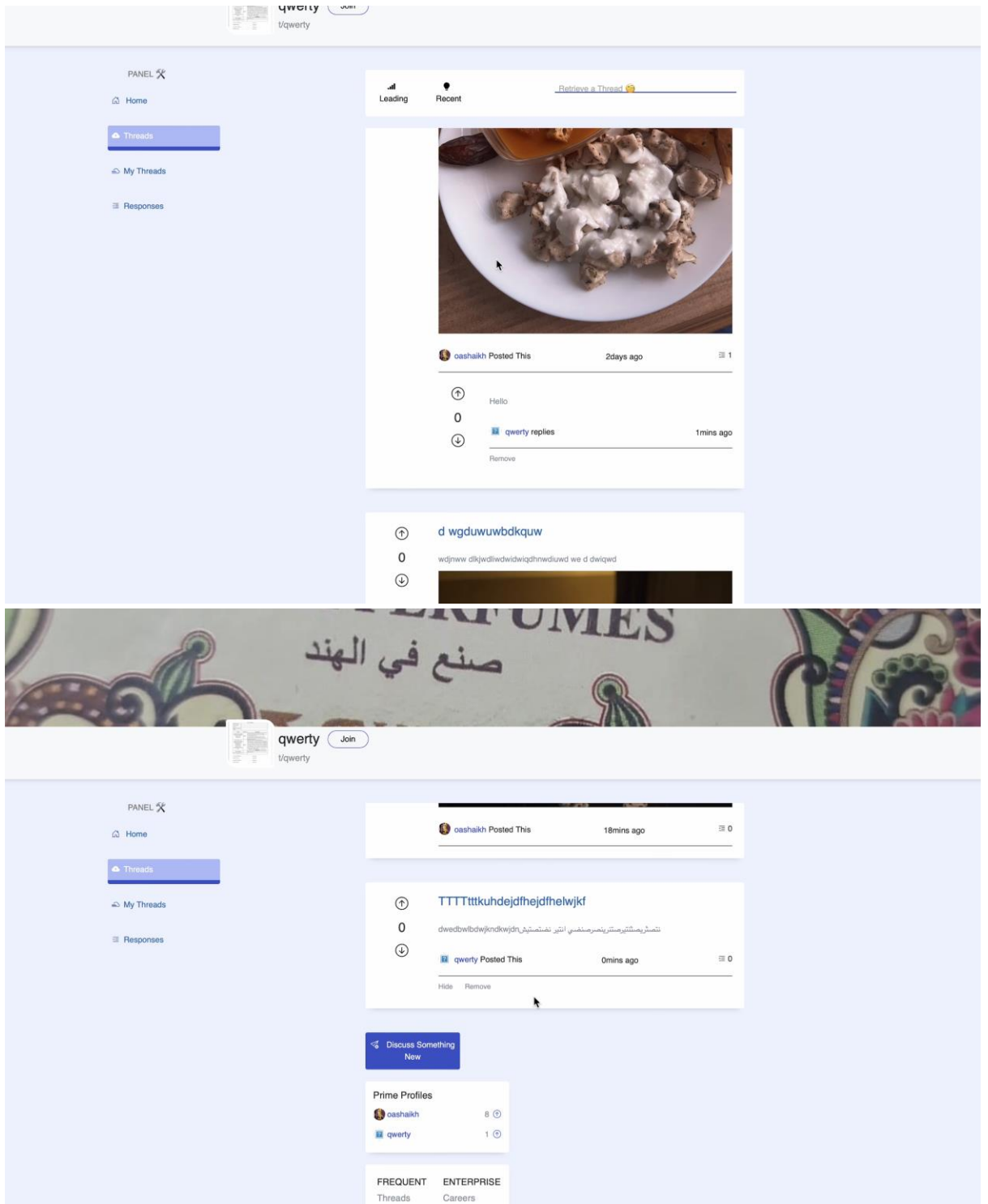
edljehck bd dbqwe

0

d ewdjh bckejwdneiwdfne ikdewq

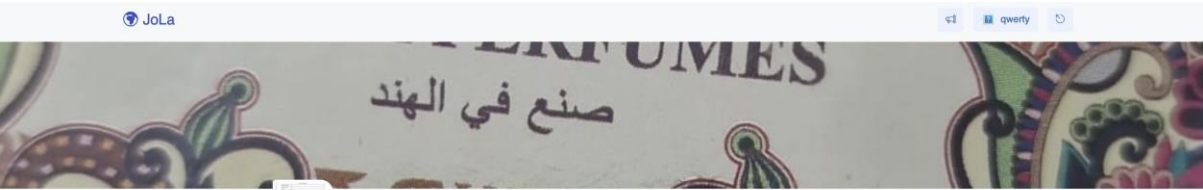
②





VII. Create Post Page

VIII. Posts Page



qwerty

Join

t/qwerty

PANEL ✕

Home

Threads

My Threads

Responses



edljehek bd dbqwe

0



d ewdjh bdkejwdfnreldne ikkdwq



Discuss Something New

Prime Profiles

oashakh

8

FREQUENT

Threads

Assistance

Broadcast

Site

ENTERPRISE

Careers

About Us

T & Conditions

Confidentiality

Press

Watermark



qwerty

Join

t/qwerty

PANEL ✕

Home

Threads

My Threads

Responses



oashakh Posted This

2days ago

0

Comment as qwerty.

Say something

Comment

Discuss Something New

Prime Profiles

oashakh

8

FREQUENT

Threads

Assistance

Broadcast

Site

ENTERPRISE

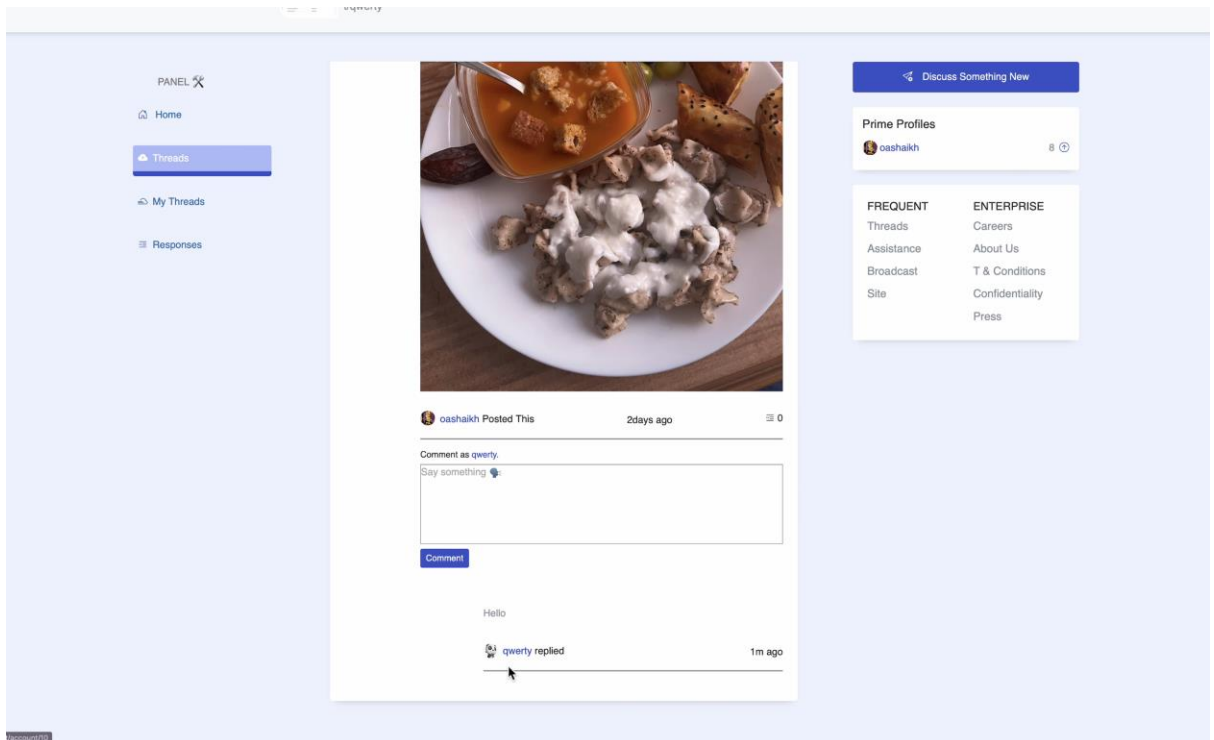
Careers

About Us

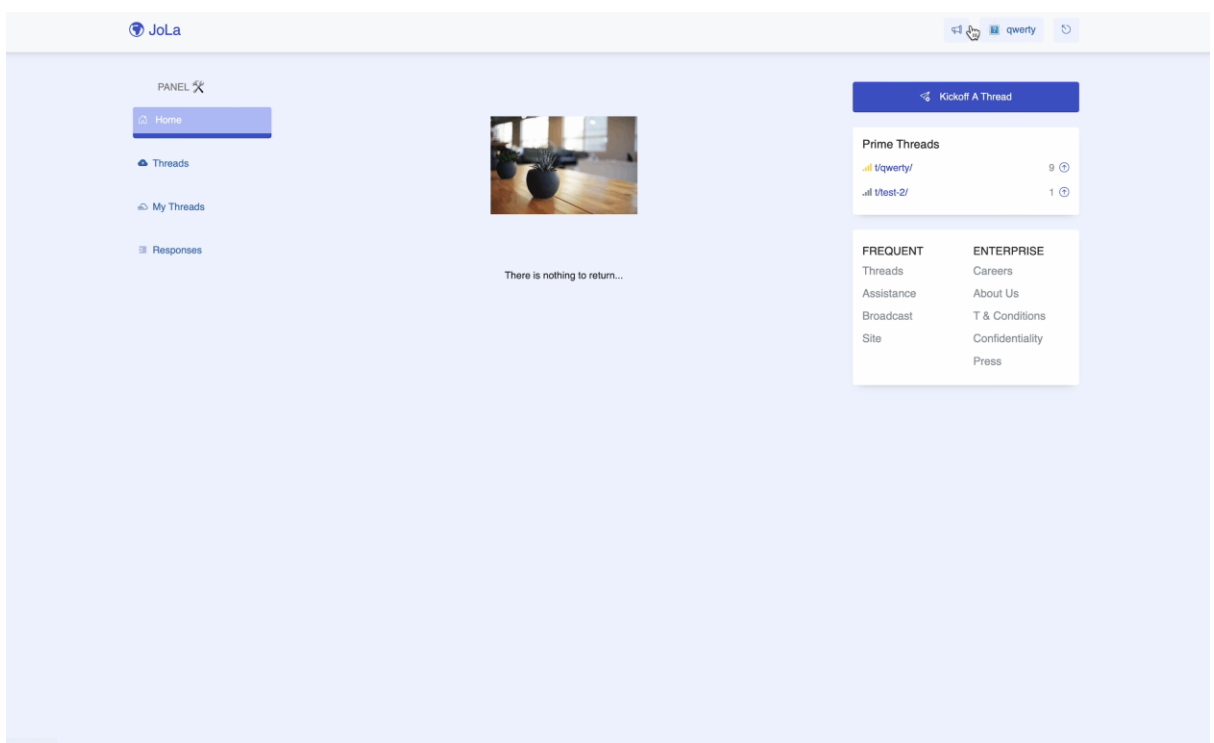
T & Conditions

Confidentiality

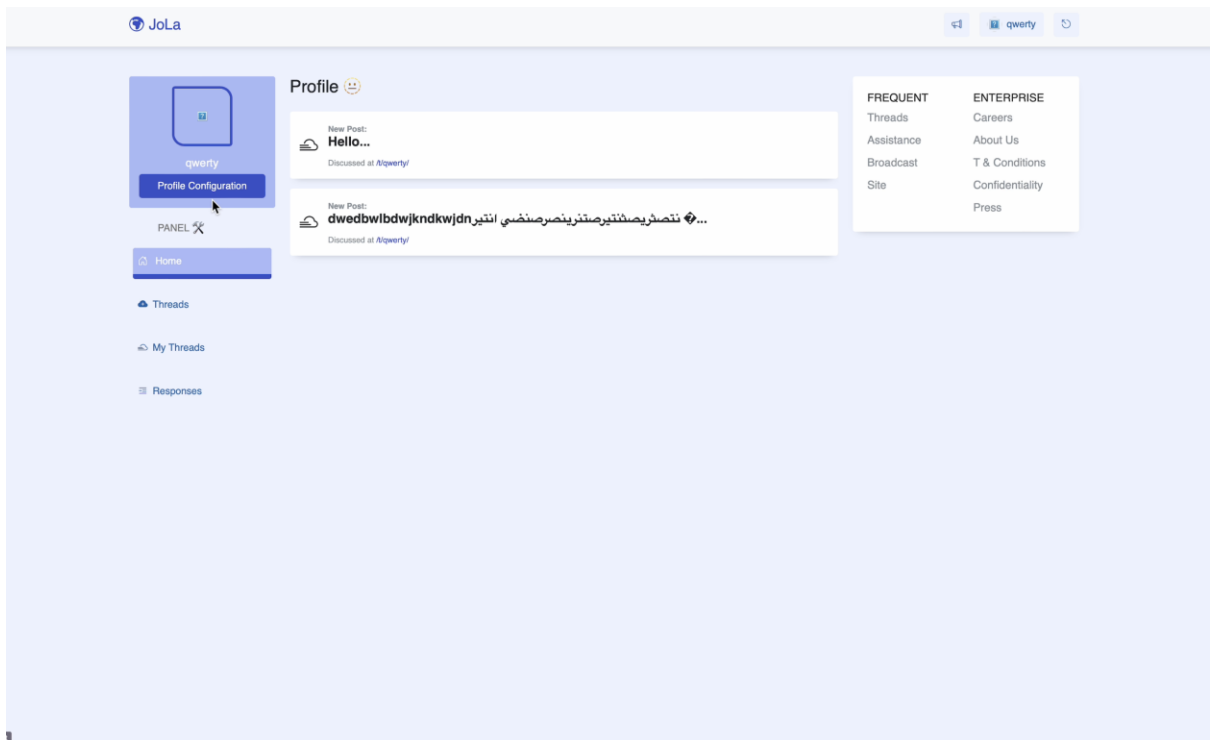
Press



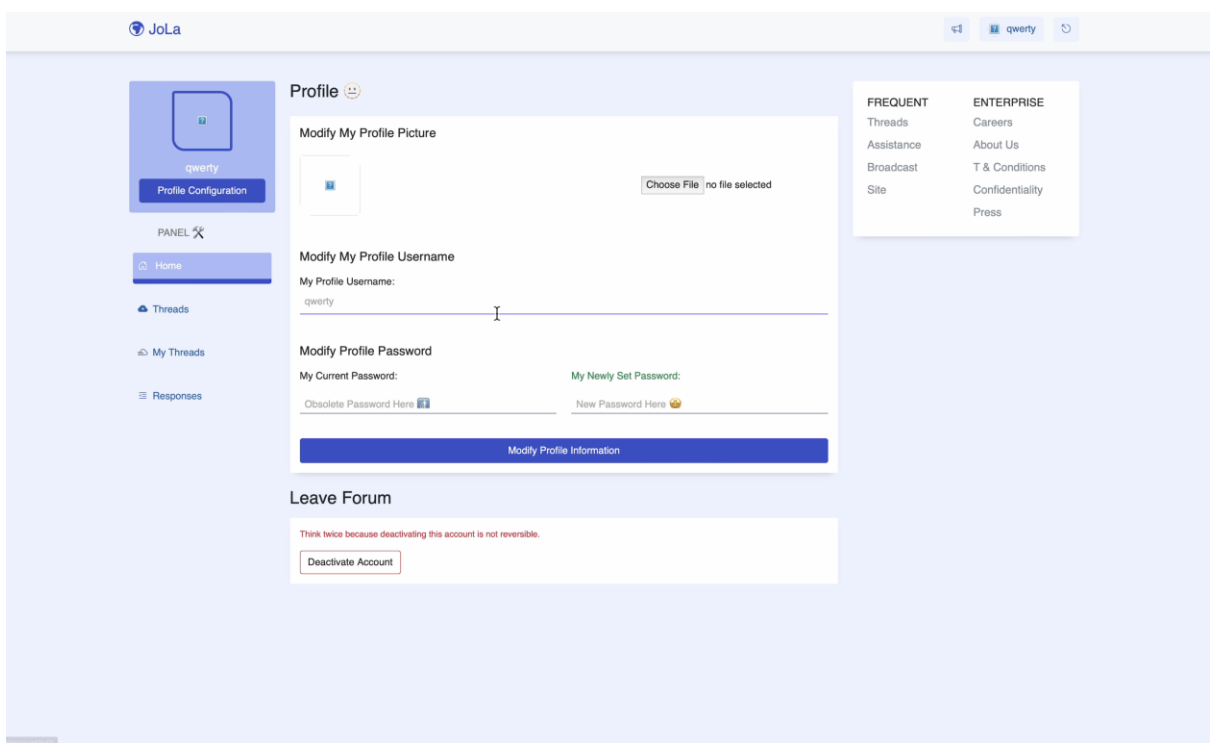
IX. User Notifications Page

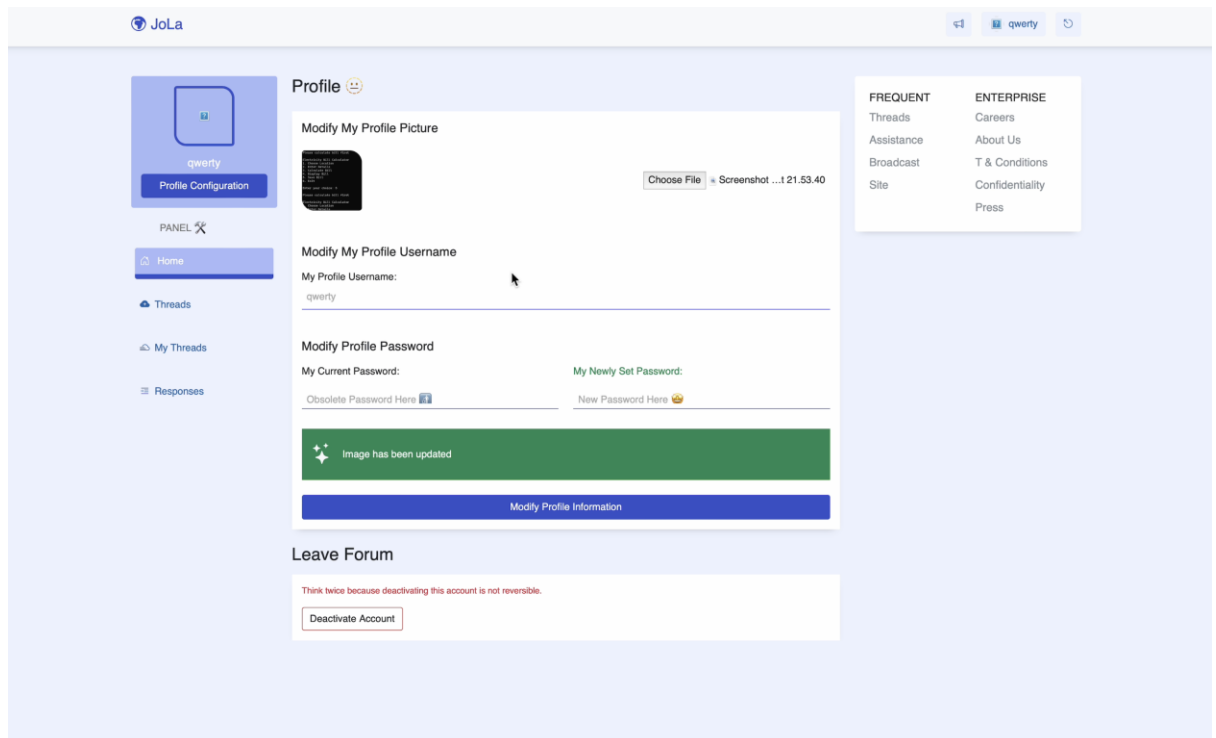


X. Account Page

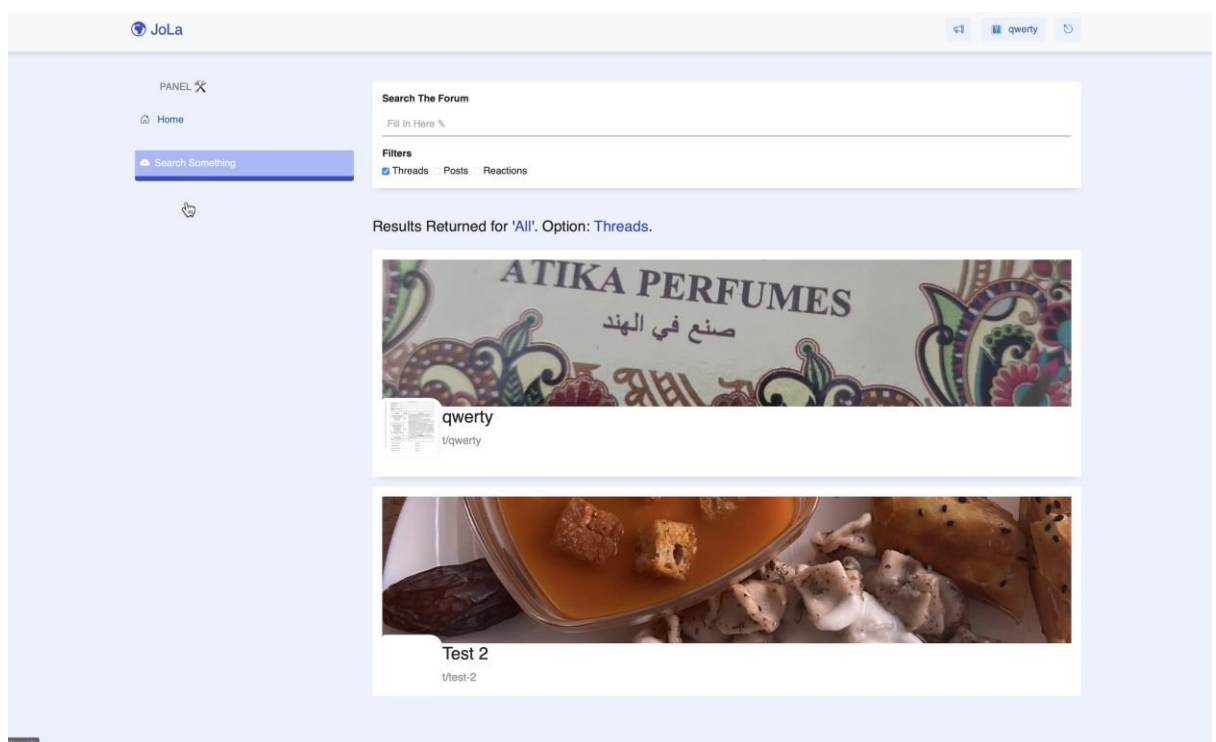


XI. Settings Page

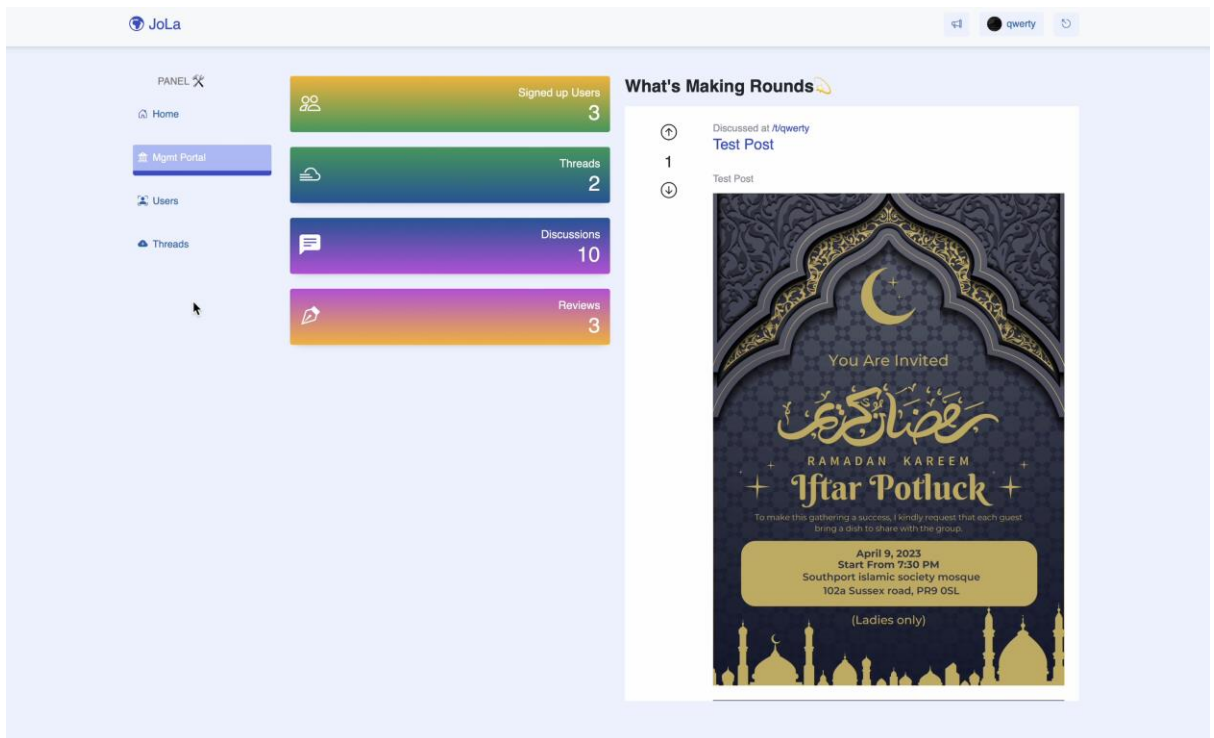




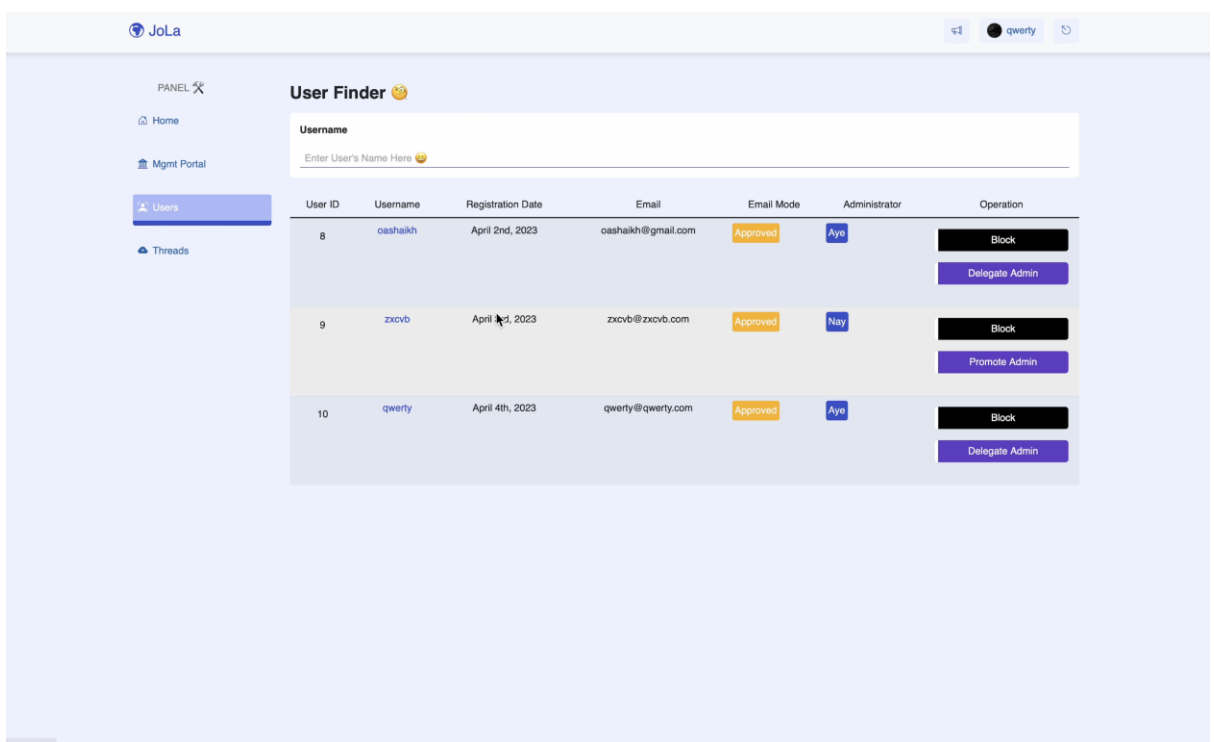
XII. Search Page

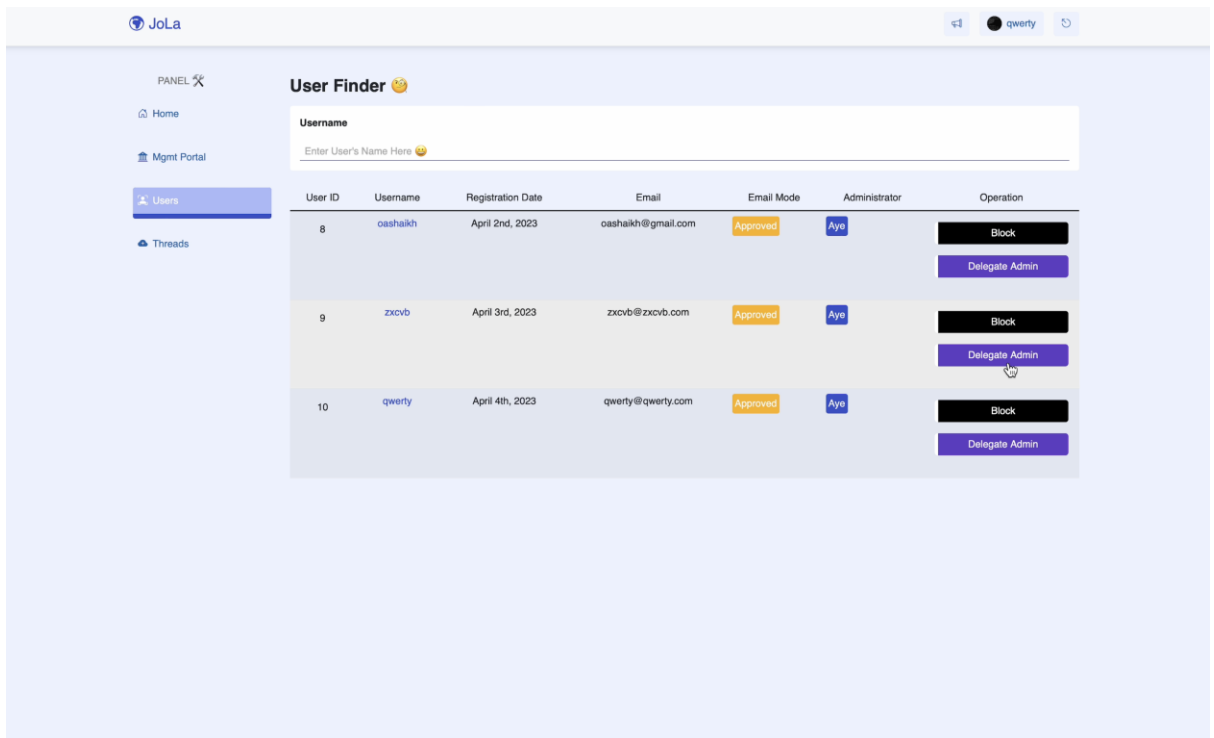


XIII. Admin Main Page

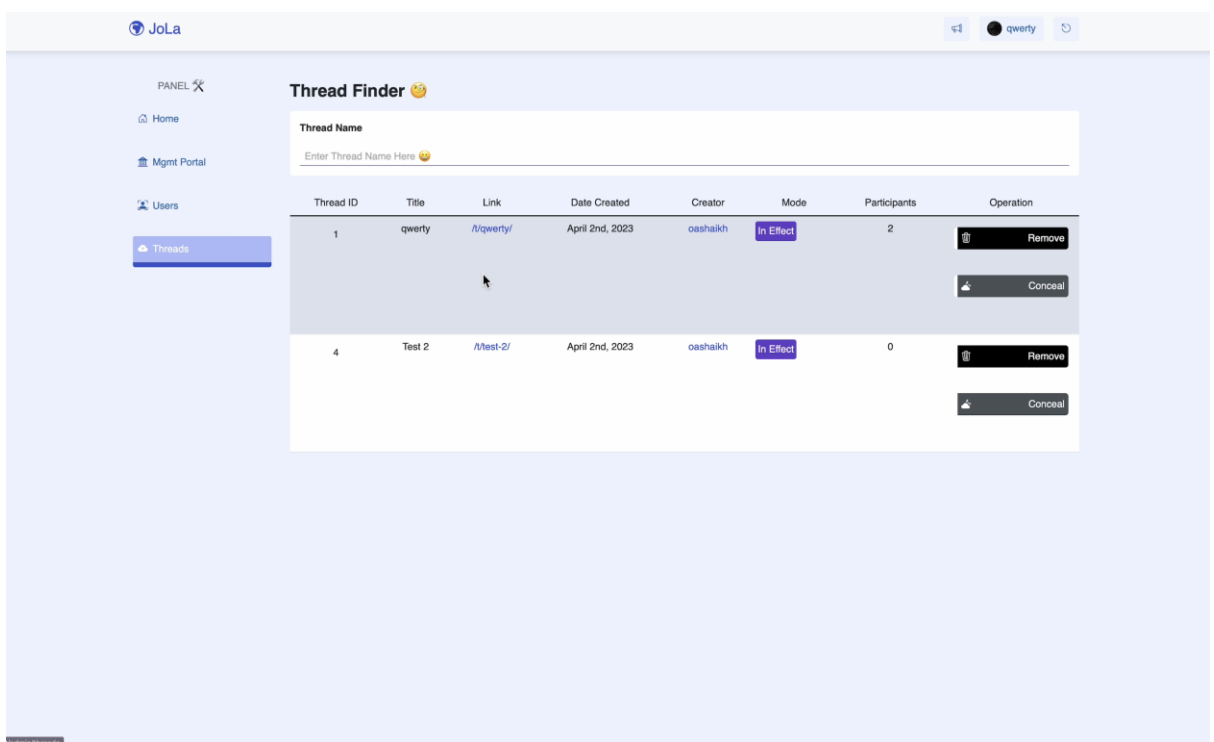


XIV. Admin Users Page





XV. Admin Threads Page



JoLa

🔍🌑🕒

PANEL

🏠 Home

🏢 Mgmt Portal

👤 Users

☁️ Threads

Thread Finder

Thread Name

Enter Thread Name Here

Thread ID	Title	Link	Date Created	Creator	Mode	Participants	Operation
1	qwerty	/!qwerty/	April 2nd, 2023	oashaikh	Concealed	2	👤 Recover
4	Test 2	/!test-2/	April 2nd, 2023	oashaikh	In Effect	0	🗑️ Remove 👤 Conceal

JoLa

🔍🌑🕒

PANEL

🏠 Home

🏢 Mgmt Portal

👤 Users

☁️ Threads

Thread Finder

Thread Name

Enter Thread Name Here

Thread ID	Title	Link	Date Created	Creator	Mode	Participants	Operation
1	qwerty	/!qwerty/	April 2nd, 2023	oashaikh	Removed	2	👤 Recover
4	Test 2	/!test-2/	April 2nd, 2023	oashaikh	In Effect	0	🗑️ Remove 👤 Conceal