huddleup

Swe573 Software Development Practice, Spring 2024
Final Report

Ömer Kaan Aslan
19/05/2024

Deployment URL:

https://huddleup.space

Git Repository URL:

https://github.com/qouv/swe573-omeraslan

Git Tag Version URL:

https://github.com/qouv/swe573-omeraslan/releases/tag/v0.9

HONOR CODE

Related to the submission of all the project deliverables for the Swe573 2024 Spring semester project reported in this report, I Ömer Kaan Aslan declare that:
- I am a student in the Software Engineering MS program at Bogazici University and am registered for Swe573 course during the Spring 2024 semester.

- All the material that I am submitting related to my project (including but not limited to the project repository, the final project report, and supplementary documents) have been exclusively prepared by myself.
- I have prepared this material individually without the assistance of anyone else with the exception of permitted peer assistance which I have explicitly disclosed in this report.

ÖMER KAAN ASLAN

PROJECT DETAILS

OVERVIEW

Huddleup is a community building platform where each community has its own post templates. Members can use these templates to create posts with different contents. Other members can respond to posts with either commenting or upvoting/downvoting. There are 2 types of communities: private and public. Public communities are open to any member to join whereas private communities require invitation to users to become members. Users can follow other users. Users can see all posts from the joined communities and follow users in their feed. Any user can create a community and become the owner of that community. Owners can assign new moderators from members. Owners have all the authorities that moderators have. Moderators can create new templates, ban/unban members and delete posts, comments. Only owners can archive communities and change ownership.

Application is developed to use in web browsers using Django and React and is not mobile friendly yet.

SOFTWARE REQUIREMENTS SPECIFICATION

Authentication:

1. A user shall be able to register and log in with a unique nickname and a password.
2. A user shall be able to reset the password of her/his account by requesting.
3. A user shall be able to change her/his nickname to any other unique nickname.

Community Creation and Membership:

4. A user shall have the capability to create and join multiple communities.
5. A user shall be able to join public communities directly.
6. A user shall be able to join private communities based on invitations.
7. A user shall enter a unique name, a description and select the type as private or public in order to create a new community.
8. A community shall have basic types of posts when created.

Moderation:

9. A community owner shall be able to authorize multiple community members as moderator.
10. A moderator shall be able to kick community members which are not moderator or owner.
11. A moderator shall be able to create a new template.
12. A moderator shall be able to edit the template.
13. A moderator shall be able to delete a template.
14. A moderator shall be able to delete a content or response.
15. A community owner shall be able to kick a moderator.
16. A community owner shall have the same authorizations with moderators.
17. A community owner shall change the type of community to private or public.
18. A community owner shall be able to transfer ownership to one community member.
19. A community owner shall not be able to leave the community without transferring ownership.
20. A community owner shall delete the owned community.
21. A community shall be archived if it is deleted by the community owner.

Communication:

22. A user shall be able to post to join the community with the provided template of posts.

23. A post shall have information about who shared, when shared and upvote/downvote counts.
24. A user shall be able edit or delete a post shared by herself/himself.
25. A post shall have a 'edited' label if it is edited by the owner.
26. A user shall be able to respond to a post shared in a joined community with text.
27. A user shall be able to edit or delete a response shared by herself/himself.
28. A response shall have a 'edited' label if it is edited by the owner.
29. A user shall be able to upvote or downvote a post shared in a joined community.
30. A user shall be able to follow multiple users.

Content:

31. A user shall see active and promoted low-engagement communities on the homepage if there are no communities or other users he/she follows.
32. A user shall be able to make basic searches for communities and users on every page.
33. A user shall be able to make basic search with input text within communities.
34. A user shall be able to make advanced search with input text, filter and sort options within communities.
35. A user shall be able to see activities of the users followed by himself/herself on the feed.

STATUS OF THE PROJECT

Until final deliverable, critical requirements are done and the application is running fully functional.

Only the below issues remained not completed:

- Create missing community settings options.
- Add edit template endpoint and necessary user interface changes.
- Add edit profile for users.

Requirement	Status
Authentication:	
A user shall be able to register and log in with a unique nickname and a password.	Completed
A user shall be able to reset the password of her/his account by requesting.	Not Completed
A user shall be able to change her/his nickname to any other unique nickname.	Not Completed
Community Creation and Membership:	
A user shall have the capability to create and join multiple communities.	Completed
A user shall be able to join public communities directly.	Completed

A user shall be able to join private communities based on invitations.	Completed
A user shall enter a unique name, a description and select the type as private or public in order to create a new community.	Completed
A community shall have a basic template of posts when created.	Completed
Moderation:	
A community owner shall be able to authorize multiple community members as moderator.	Completed
A moderator shall be able to kick community members which are not moderator or owner.	Completed
A moderator shall be able to create a new template.	Completed
A moderator shall be able to edit the template.	Not Completed
A moderator shall be able to delete a template.	Completed
A moderator shall be able to delete a content or response.	Completed

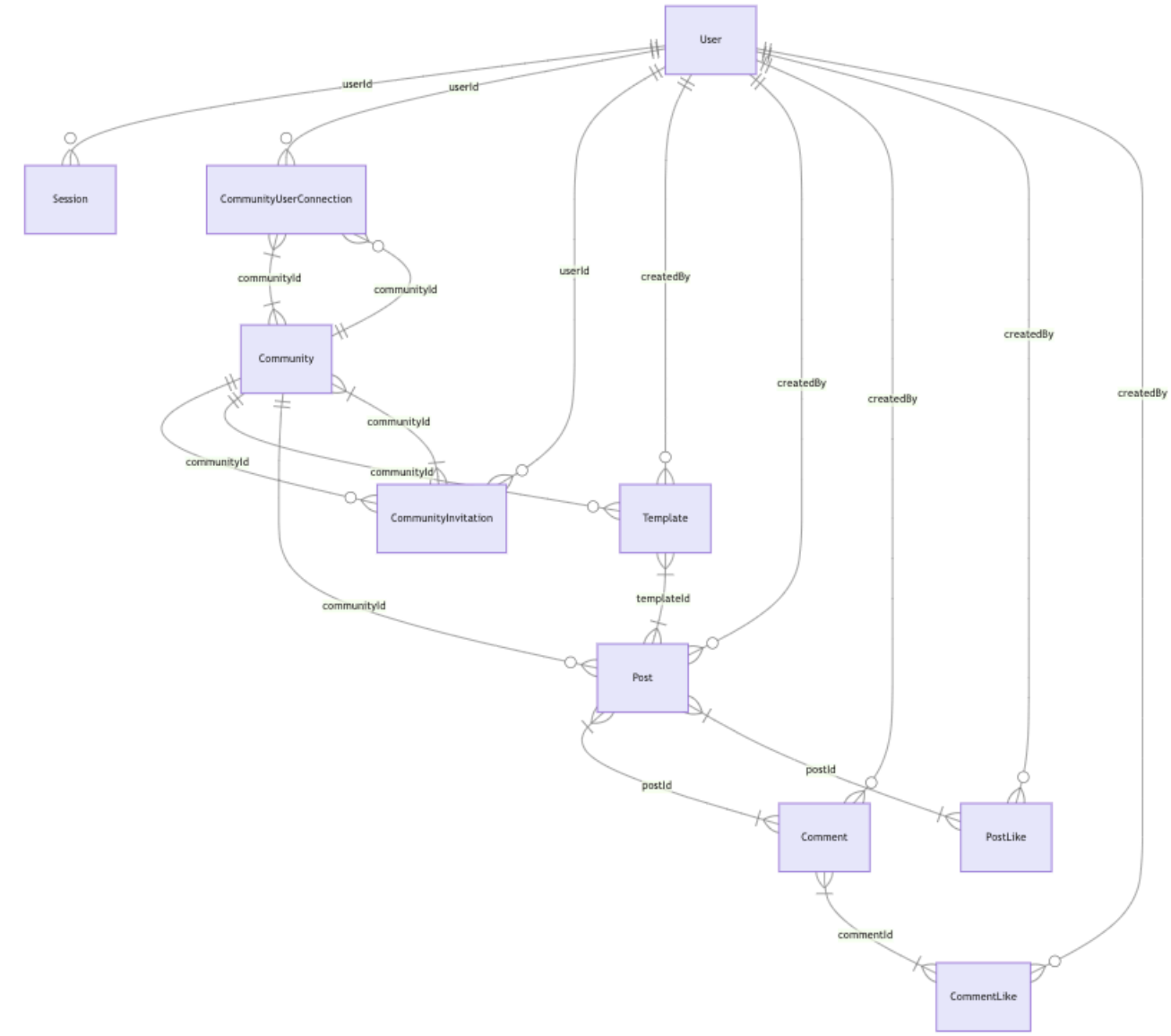
A community owner shall be able to kick a moderator.	Completed
A community owner shall have the same authorizations with moderators.	Completed
A community owner shall change the type of community to private or public.	Not Completed
A community owner shall be able to transfer ownership to one community member.	Completed
A community owner shall not be able to leave the community without transferring ownership.	Completed
A community owner shall delete the owned community.	Completed
A community shall be archived if it is deleted by the community owner.	Completed
Communication:	
A user shall be able to post to join the community with the provided template of posts.	Completed
A post shall have information about who shared, when shared and upvote/downvote counts.	Completed

A user shall be able edit or delete a post shared by herself/himself.	Completed
A post shall have a 'edited' label if it is edited by the owner.	Completed
A user shall be able to respond to a post shared in a joined community with text.	Completed
A user shall be able to edit or delete a response shared by herself/himself.	Completed
A response shall have a 'edited' label if it is edited by the owner.	Completed
A user shall be able to upvote or downvote a post shared in a joined community.	Completed
A user shall be able to follow multiple users.	Completed
Content:	
A user shall see active and promoted low-engagement communities on the homepage if there are no communities or other users he/she follows.	Completed
A user shall be able to make basic searches for communities and users on every page.	Completed
A user shall be able to make basic search with input text within communities.	Completed

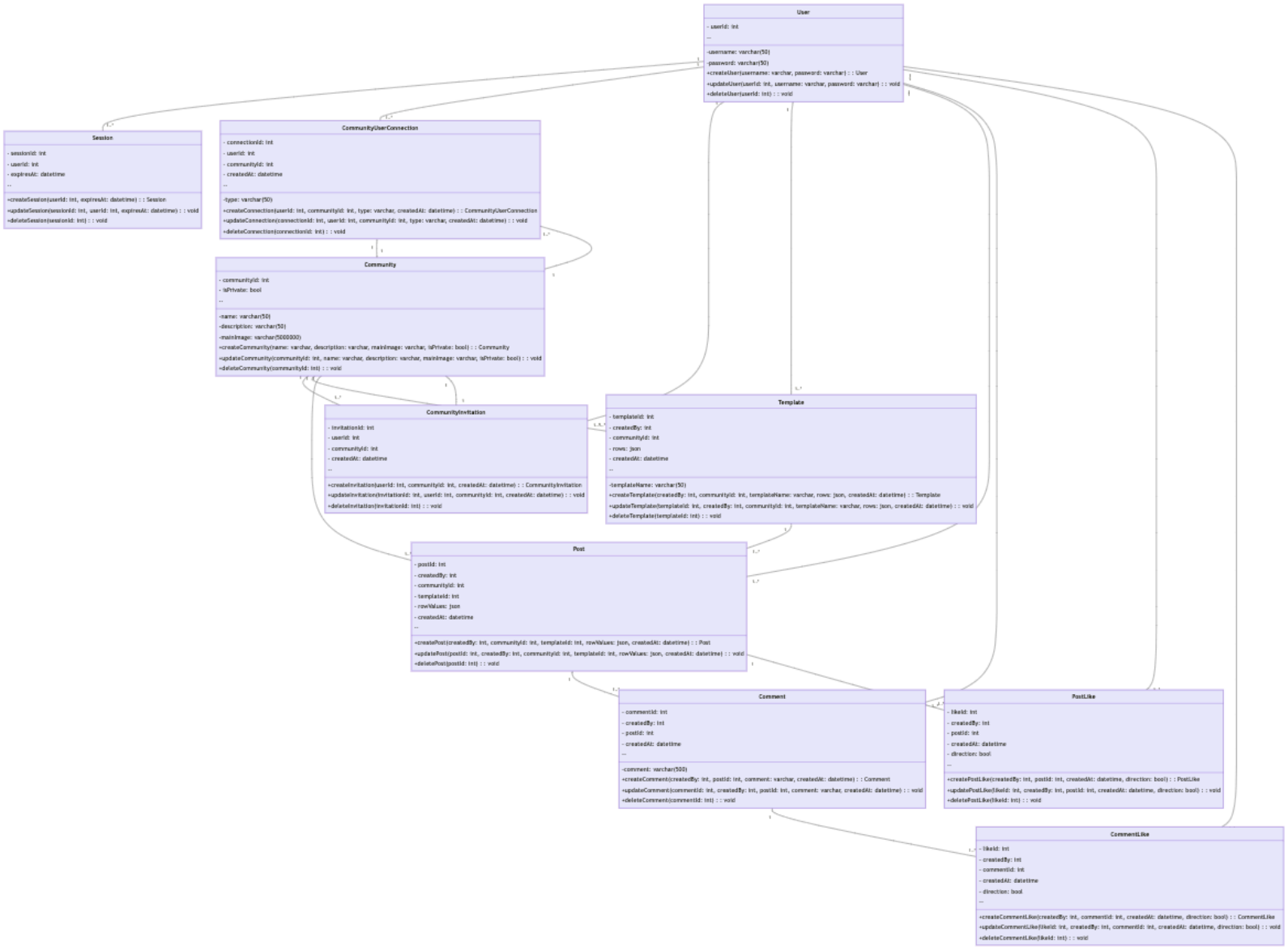
A user shall be able to make advanced search with input text, filter and sort options within communities.	Completed
A user shall be able to see activities of the users followed by himself/herself on the feed.	Completed

DESIGN DOCUMENTS

Entity Relationship diagram that show how entities relate to each other:



Class diagram that show structure of system:



STATUS OF DEPLOYMENT

Application is dockerized and deployed to AWS EC2 instance. Also SSL certificate added to use secure transfer protocol.

Deployed on the URL:

https://huddleup.space

HOW TO SET UP?

STEP 1:

Clone the repo using https

```
git clone https://github.com/qouv/swe573-omeraslan.git
```

or using ssh

```
git clone git@github.com:qouv/swe573-omeraslan.git
```

STEP 2:

Start docker on your computer.

STEP 3:

Inside “swe573-omeraslan” folder run below command:

```
docker-compose up --build
```

or run docker in daemon mode:

```
docker-compose up --build -d
```

STEP 4:

In your browser go to below address after docker build all containers:

```
http://localhost:3000
```

VIDEO TUTORIAL

This is link to Huddleup video tutorial:

https://youtu.be/VN5yRPnrn9o?si=x4jbaQwlhHo3t4WX

USER MANUAL

NEEDED INFORMATION FOR TESTING

Any of the below users can be used to test:

Username: merlin

Password: isthecutestcat

Communities: Owner of the Board Game Geek

Connections: Following boardgamelovr

Username: boardgamelovr

Password: boardgamelovr

Communities: Member of the Board Game Geek, Moderator of the Homebrewers (private)

Connections: Followed by merlin

Username: foody

Password: iamsohungry

Communities: Owner of the Homebrewers (private)

Connections: No connections

Username: solenopsis

Password: solenopsis

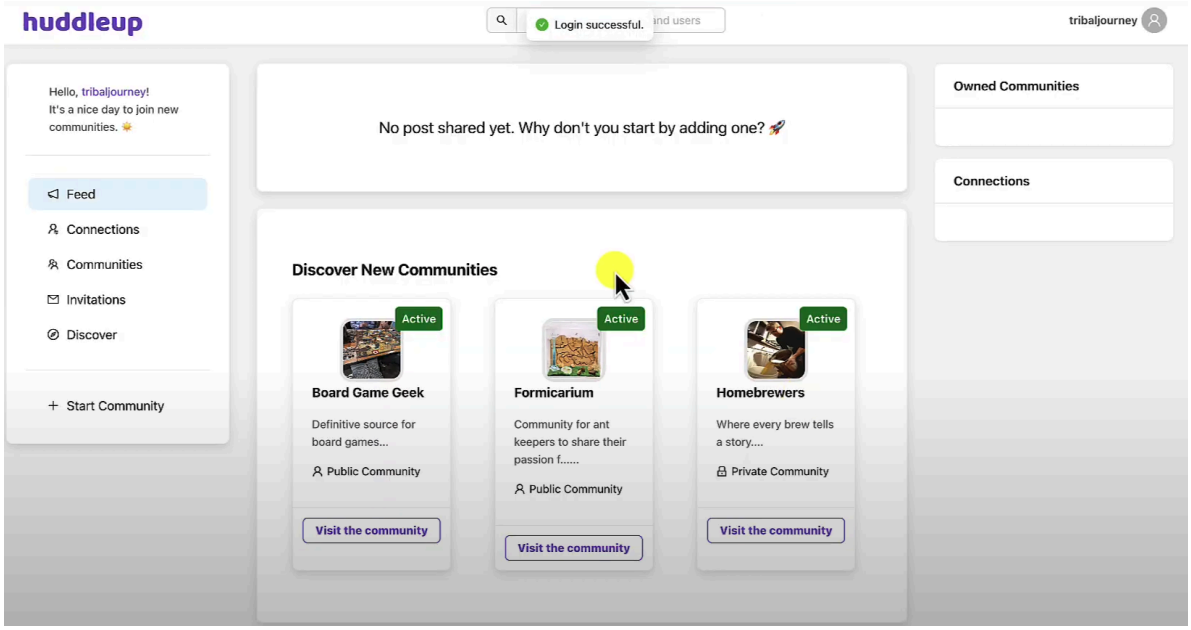
Communities: Owner of the Formicarium, Banned from Board Game Geek

Connections: No connections

TUTORIAL

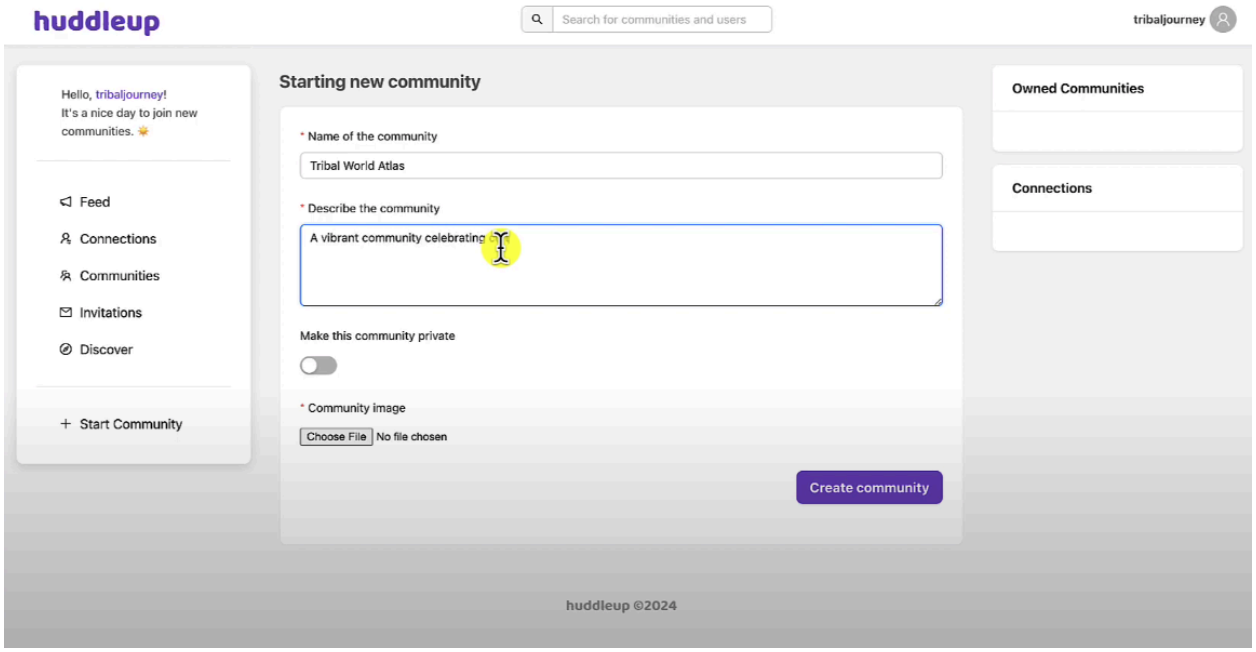
First go to https://huddleup.space

After register/login you will see the most active and some promoted communities in your feed if there is no post yet.

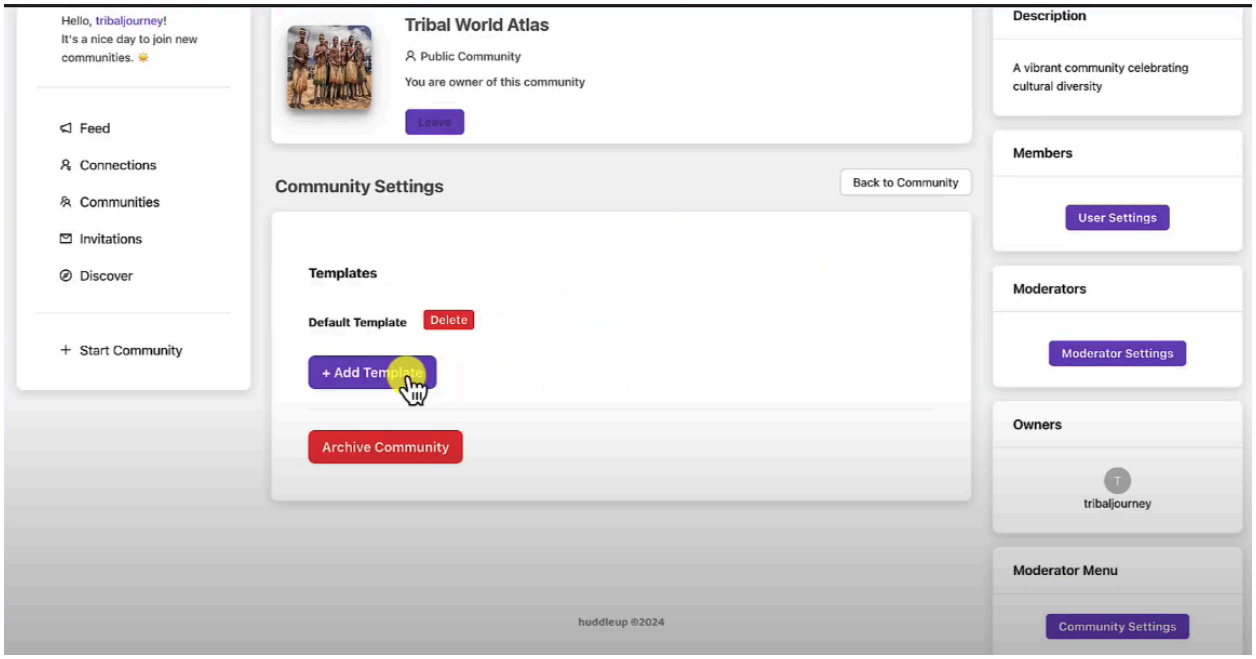


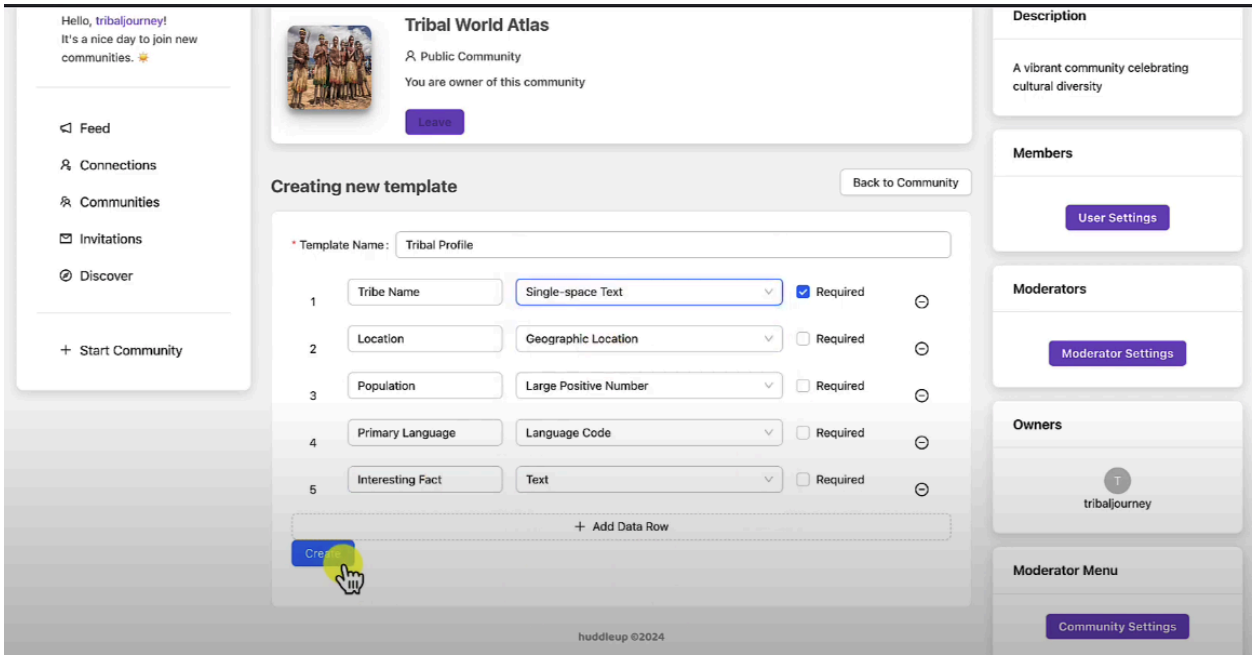
Go to Start Community from the left menu.

Fill out the community form and select an image.

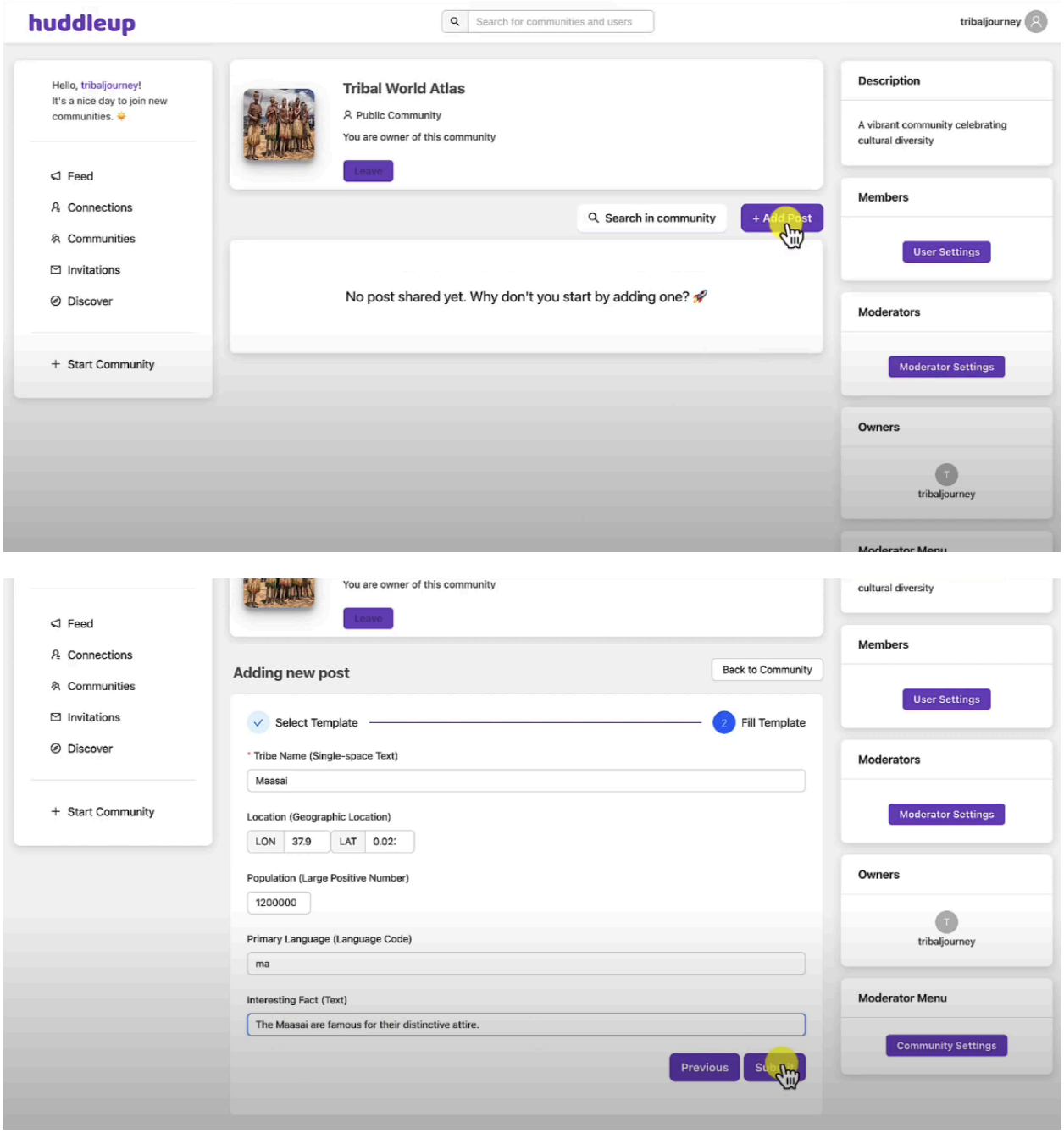


Owners have all the authority that moderators have. Moderators can create new templates from community settings.

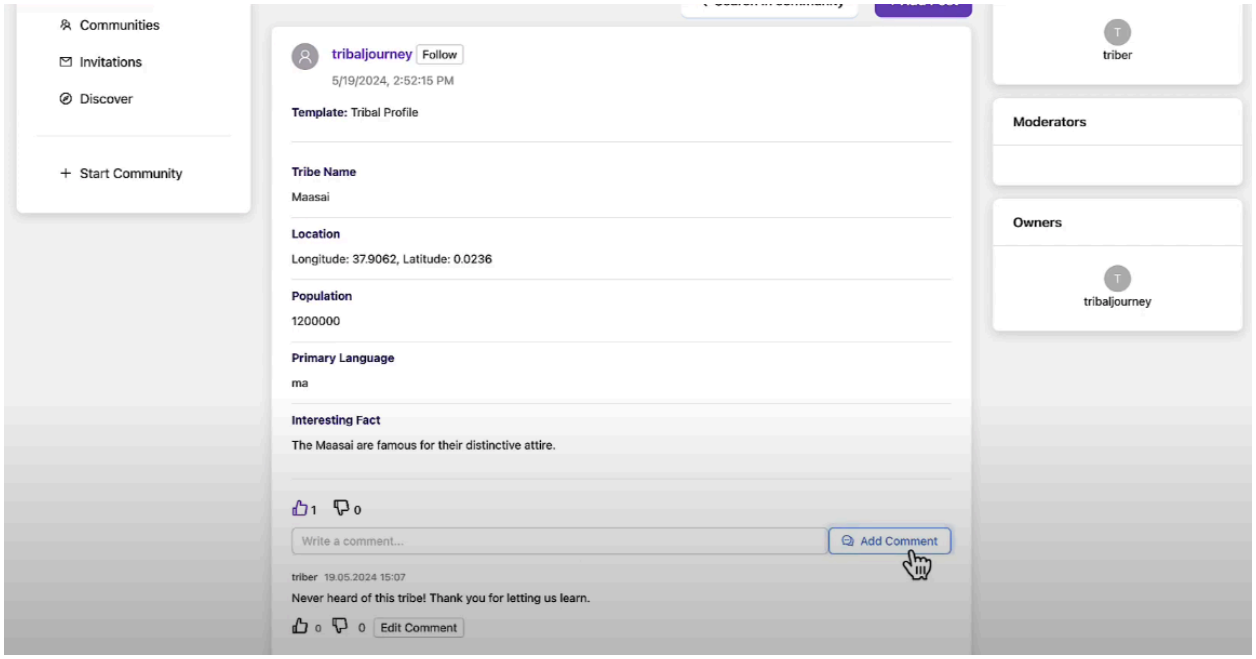




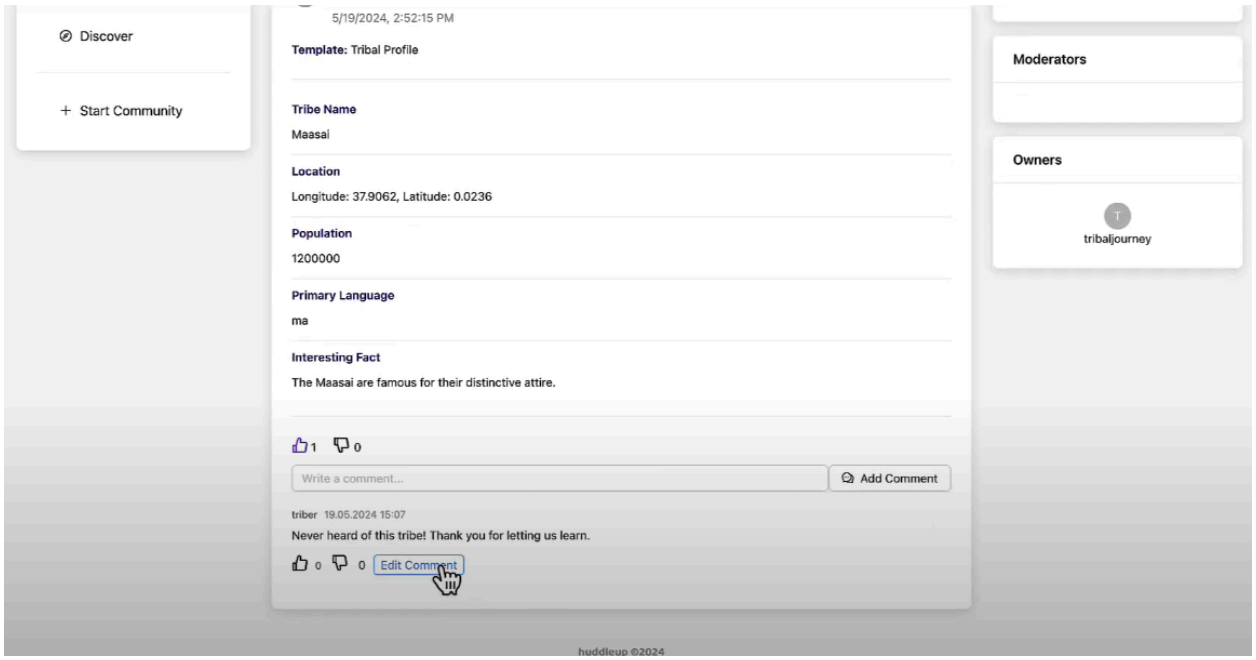
Afterwards you can select the template to create a post with it. Fill the content and submit to other members to see your post.



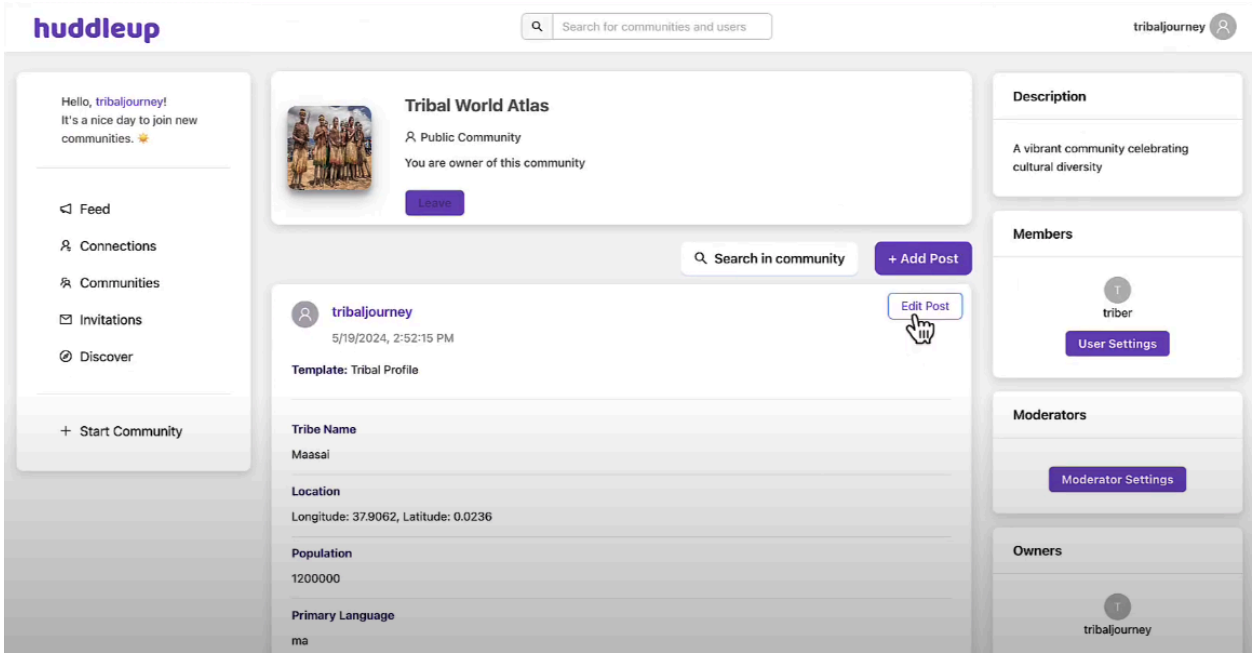
Only members can respond to posts with upvote, downvote or comments.



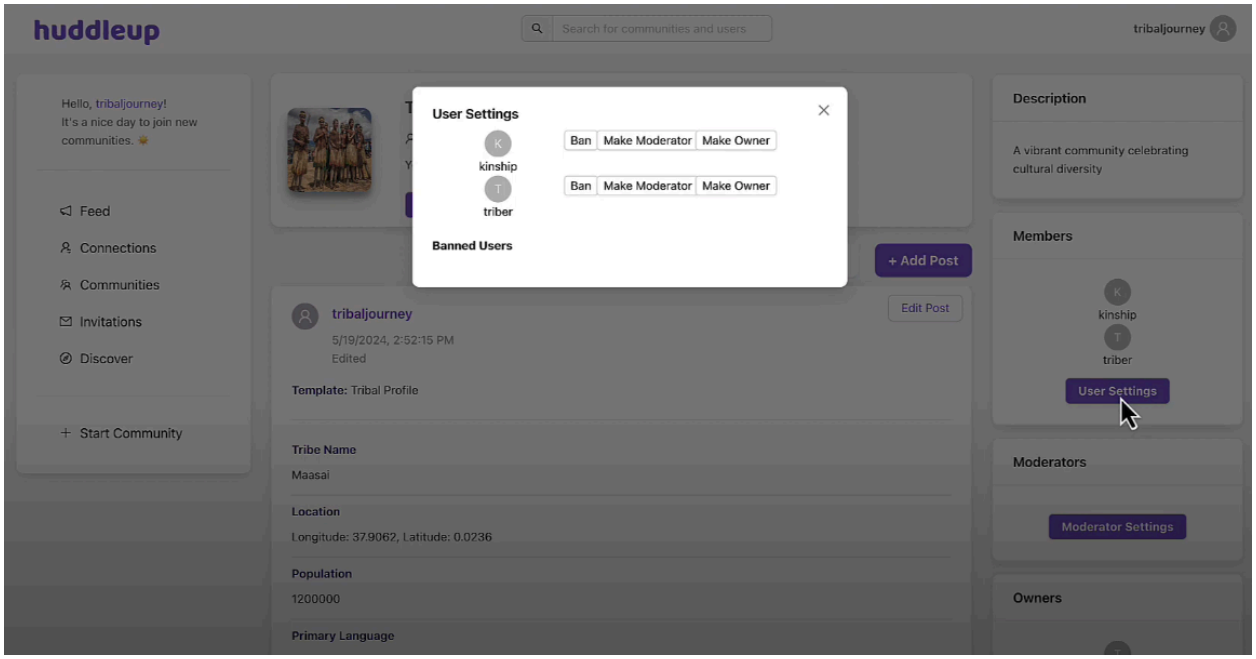
Comments can be edited only by creators and they will have edited label.



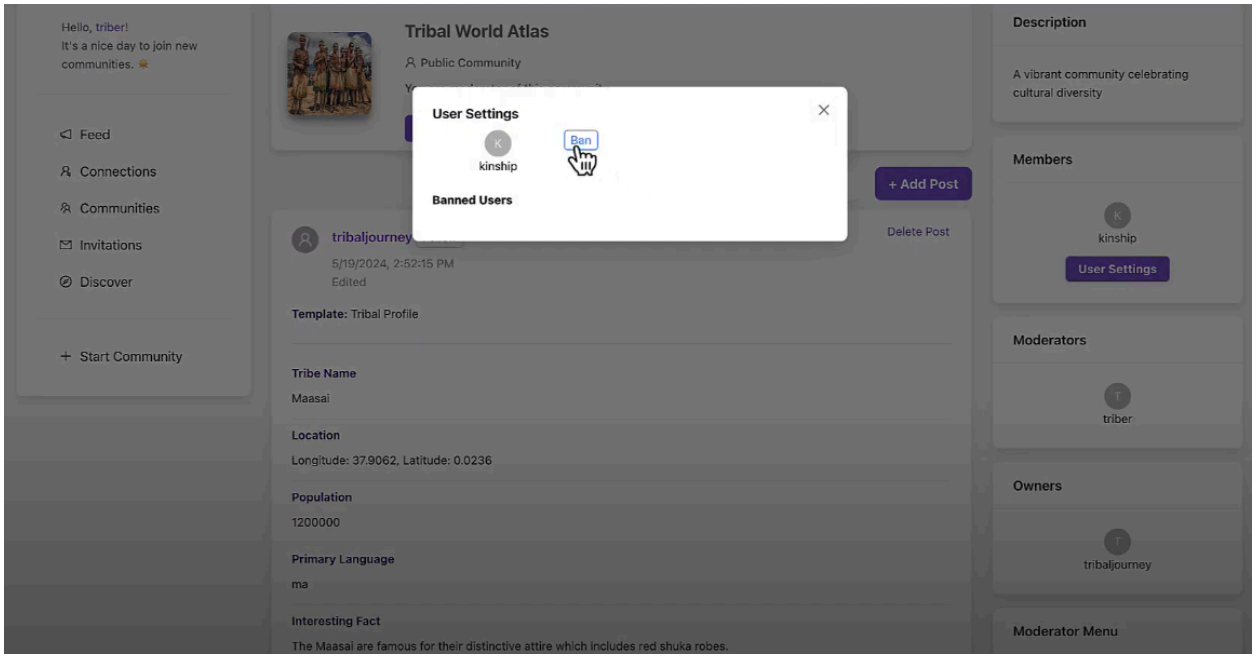
Just like comments posts can be edited only by creators and they will have edited label.



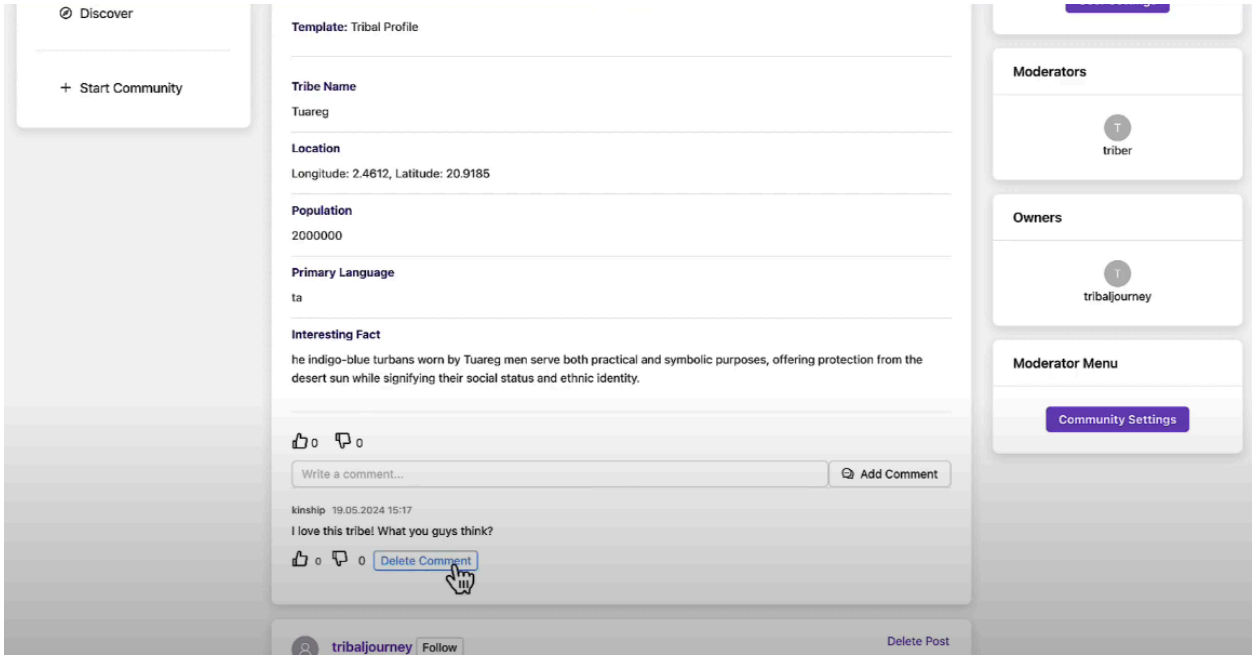
Owners of communities can assign moderators from members on user settings. And unassign moderators from moderators settings.



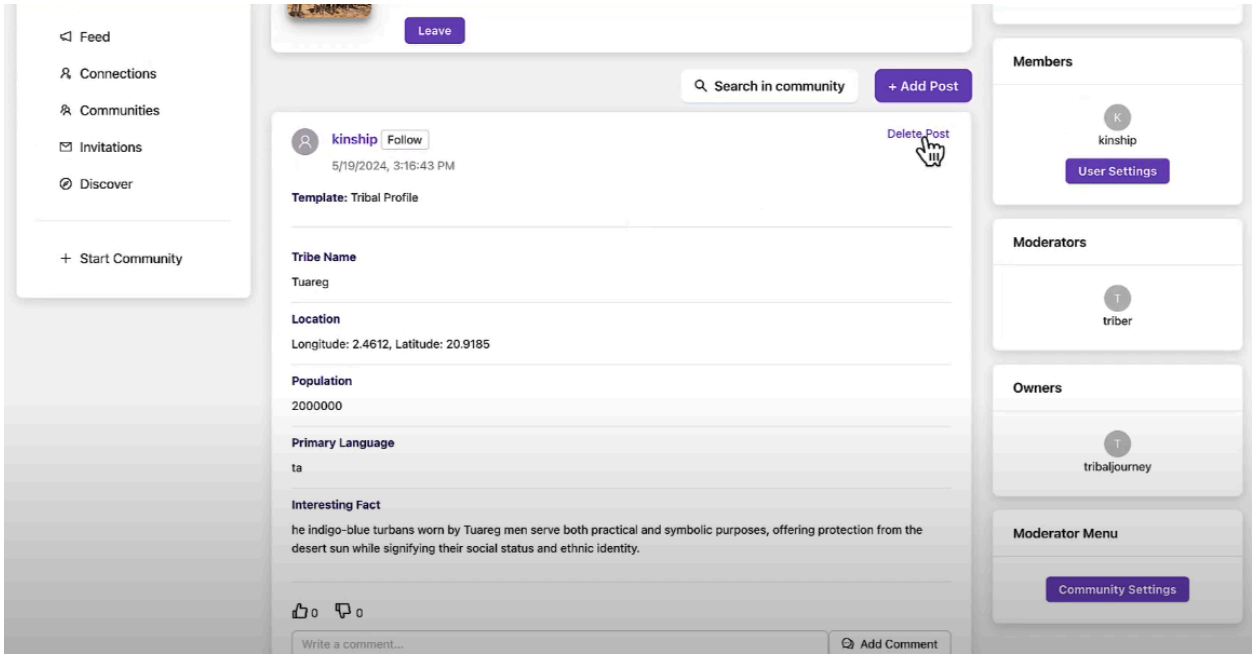
Moderators can ban and unban users from user settings.



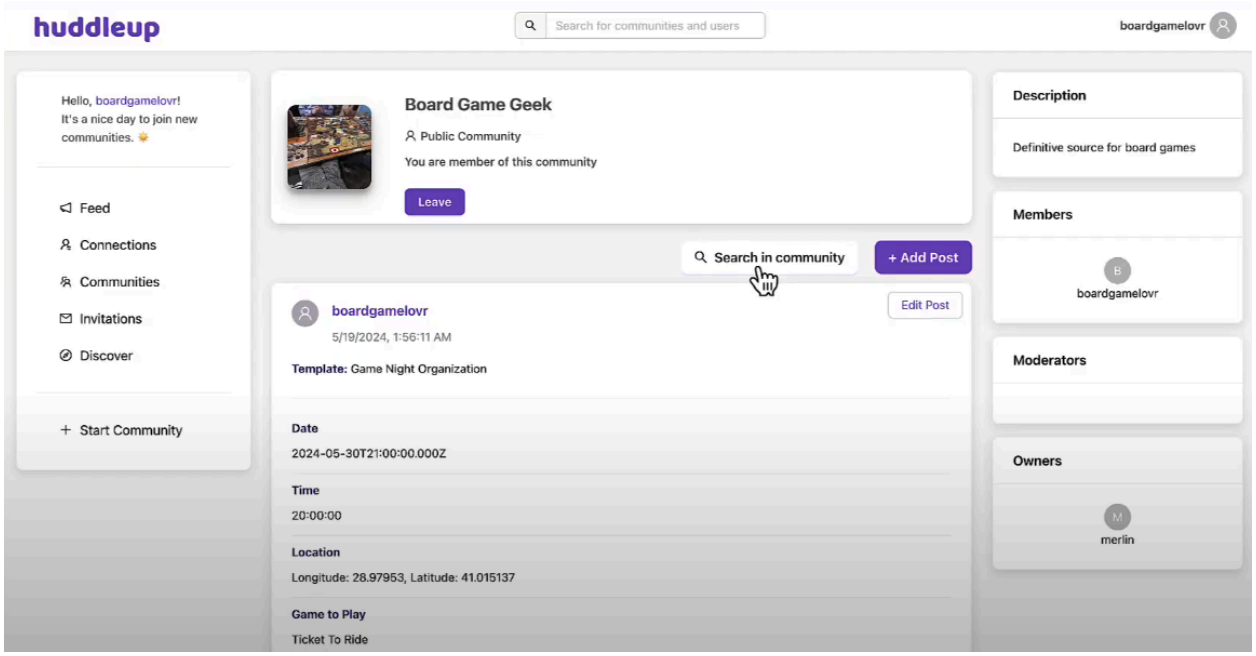
Moderators can delete comments.



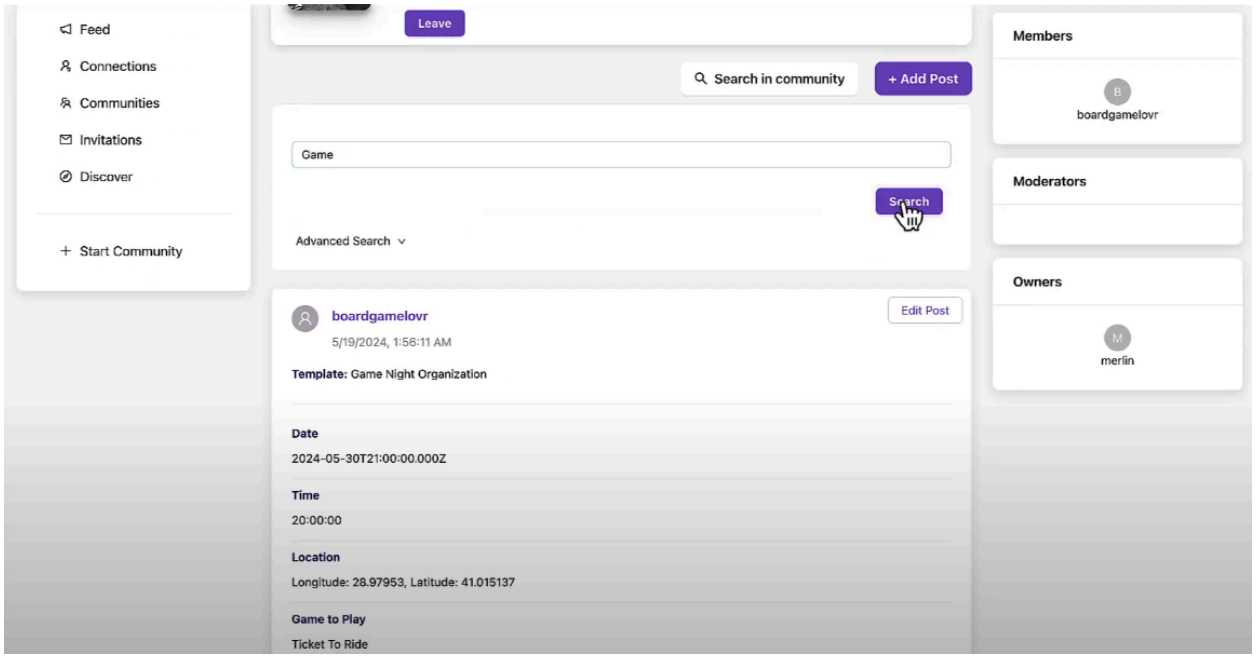
Moderators can delete posts.



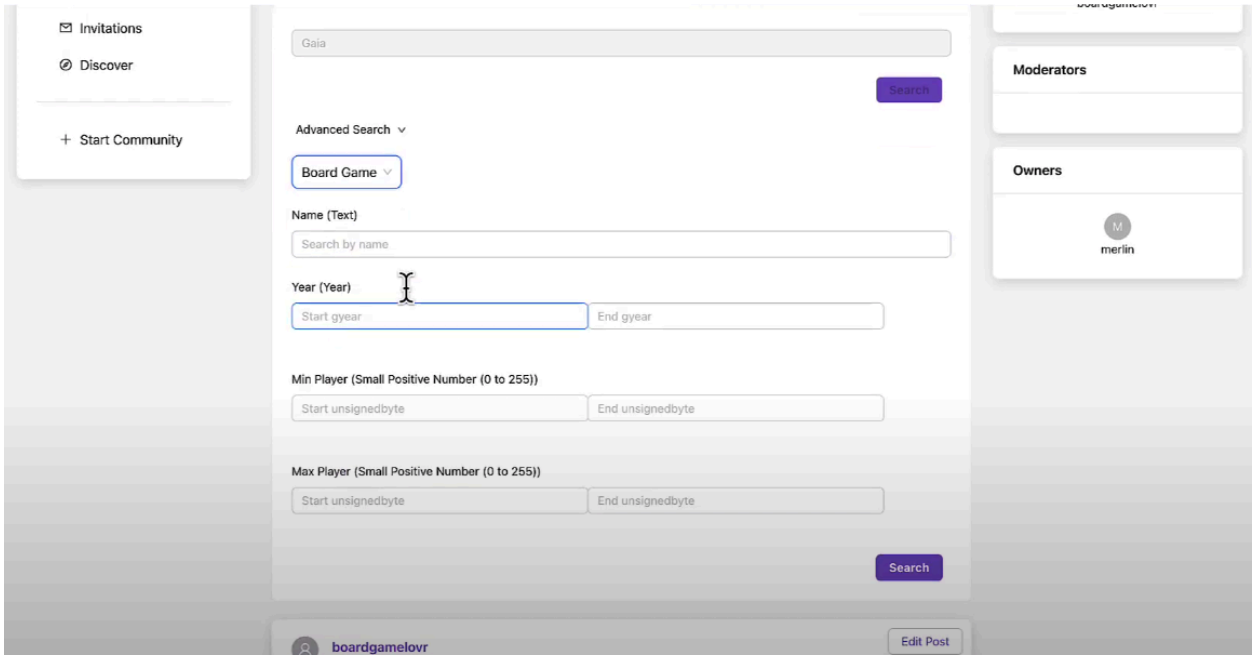
There are two kinds of searches within a community.



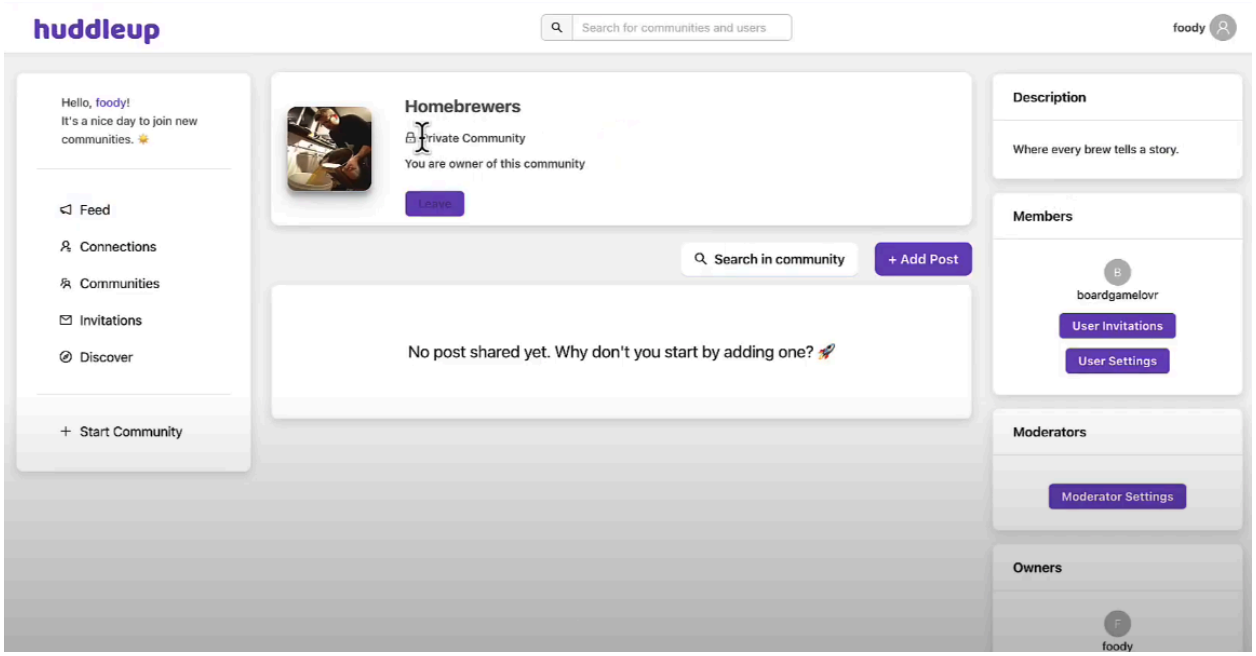
Basic search check for a query in any content or template name.



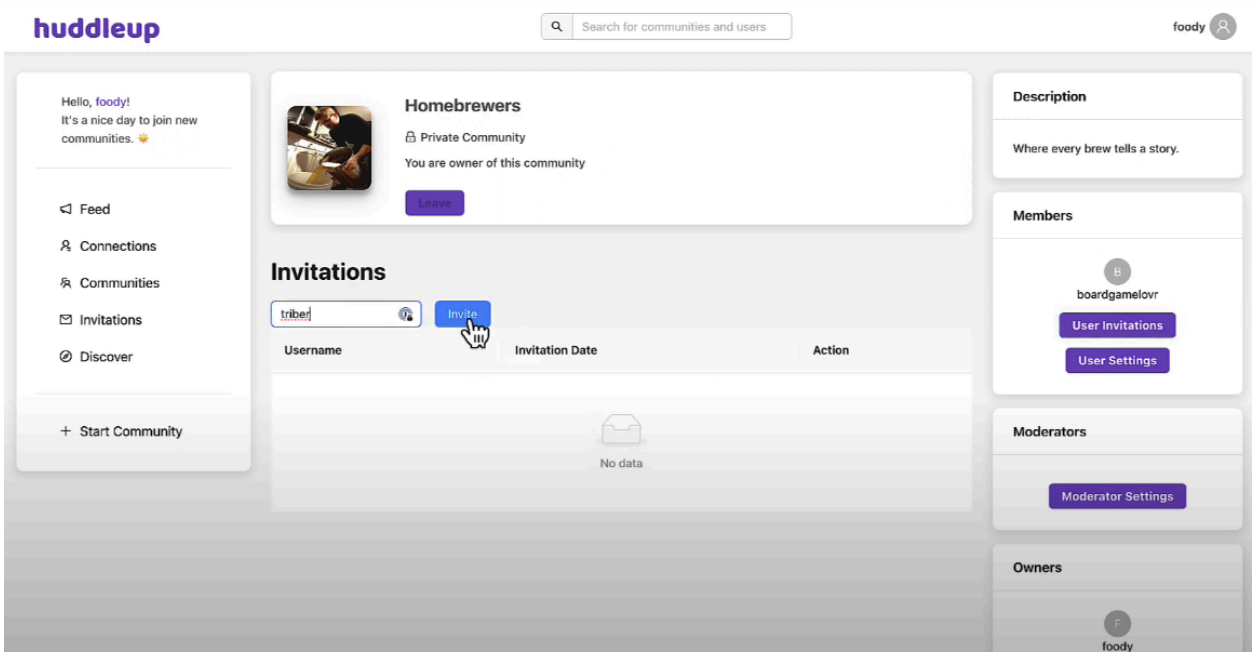
In advanced search you can select a template and filter posts according to any data row.



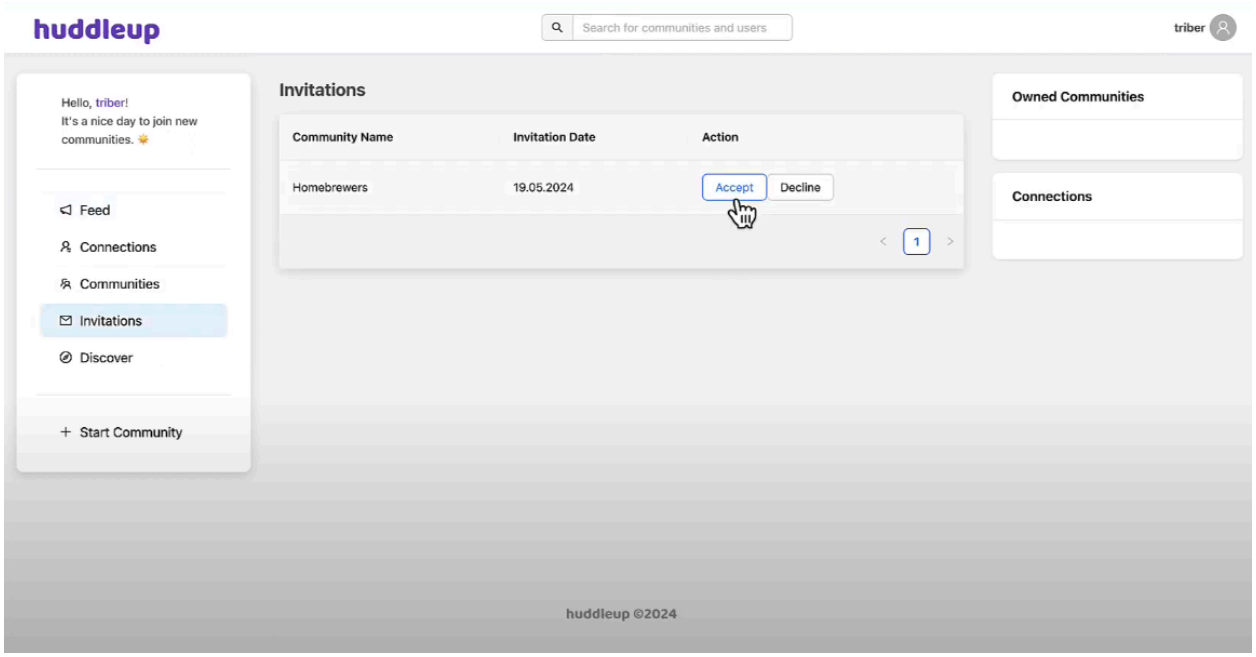
In private communities users need an invitation to join and see the posts.



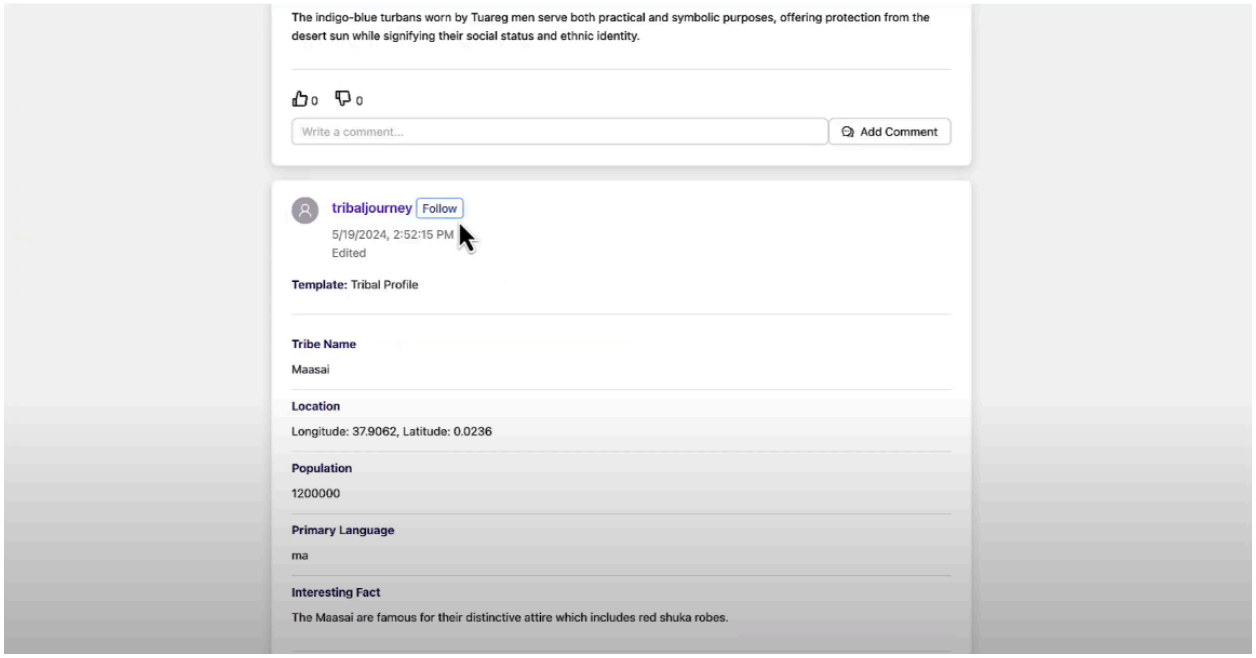
Only moderators and owners can create an invitation.



Later invited users can accept or decline invitation from invitations on the left menu.



Users can follow other users. And the follower user will see posts shared by the followed user on the feed.



TEST RESULTS

USER TESTS

Test Case	Steps	Expected	Actual	Notes	Post-condition
1: Register	1. Navigate to /register page. 2. Fill in the registration form. 3. Click "Register".	Redirect to the login page with a success message.	True		A new user is created in the database.
2: Login	1. Navigate to /login page. 2. Enter valid credentials. 3. Click "Login".	Redirect to /feed page and user is logged in.	True		The user is logged in.
3: Create Community	1. Login as an existing user. 2. Navigate to /communities/new. 3. Fill in the form. 4. Click "Create Community".	New community is created, and redirect to community feed.	True		A new community is created in the database.

4: Join Community	1. Login as an existing user. 2. Navigate to /discover. 3. Select a public community. 4. Click "Join".	User added to community as a member, and appears in user's community list.	True		The user is a member of the selected community.
5: Create Post	1. Login as an existing user. 2. Navigate to community feed. 3. Click "Create Post". 4. Fill in the form. 5. Click "Create Post".	New post is created and appears in the community feed.	True		A new post is created in the community.
6: Like Post	1. Login as an existing user. 2. Navigate to community feed. 3. Click "Like" on a post.	Post is liked by the user and like count increases.	True		The user has liked the post.
7: Follow User	1. Login as an existing user. 2. On any of the other user's posts click "Follow".	User follows the selected user, and the selected user appears in the user's connection list.	True		The user is following the selected user.

8: Create Template	1. Login as an owner/moderator. 2. Navigate to /communities/{communityId}/create-template. 3. Fill in the form. 4. Click "Create Template".	New template is created and appears in the community templates list.	True		A new template is created for the community.
9: Accept Invitation	1. Login as an existing user. 2. Navigate to /invitations. 3. Click "Accept" for an invitation.	User is added to the community as a member, and invitation removed from list.	True		The user is a member of the community.
10: Delete Comment	1. Login as owner/moderator. 2. Navigate to community feed. 3. Click on a post to view comments. 4. Click "Delete" for a comment.	Comment is deleted and no longer appears in the list.	True		The comment is deleted from the post.
11: Edit Post	1. Login as post owner. 2. Navigate to community feed. 3. Click "Edit" for the post. 4. Modify details. 5. Click "Save".	Post is updated with new details and appears in the community feed.	True		The post is updated with new details.

UNIT TESTS

Test Case	Steps	Expected	Actual	Notes	Post-condition
1: Create Community	1. Set up a test user. 2. Create a POST request with community details. 3. Set request.user to the test user. 4. Call create_community view directly with the request.	New community is created, and the response status code is 201.	True	Community created successfully with the owner set to the test user.	A new community is created in the database.
2: Get Community Members	1. Create a community (from the previous test). 2. Set up a new member user. 3. Create a member connection for the new user. 4. Create a POST request to get community members. 5. Set request.user to the test user. 6. Call get_community_members view directly with the request.	Member list is retrieved with the newly added member.	True	Member users successfully added to the community and retrieved.	The member user appears in the community members list.

3: Get Community Banned Users	1. Create a community (from the previous test). 2. Set up a new banned user. 3. Create a banned connection for the new user. 4. Create a POST request to get banned users. 5. Set request.user to the test user. 6. Call get_community_banned view directly with the request.	Banned user list is retrieved with the newly added banned user.	True	Banned users successfully added to the community and retrieved.	The banned user appears in the banned users list.
4: Get Community Moderators	1. Create a community (from previous test). 2. Set up a new moderator user. 3. Create a moderator connection for the new user. 4. Create a POST request to get community moderators. 5. Set request.user to the test user. 6. Call get_community_moderators view directly with the request.	Moderator list is retrieved with the newly added moderator.	True	Moderator user successfully added to the community and retrieved.	The moderator user appears in the community moderators list.

5: Get Community Owners	1. Create a community (from previous test). 2. Set up a new owner user. 3. Create an owner connection for the new user. 4. Create a POST request to get community owners. 5. Set request.user to the test user. 6. Call get_community_owners view directly with the request.	Owner list is retrieved with both the initial owner and the newly added owner.	True	Both the initial owner and the new owner are correctly listed.	The owner user appears in the community owners list along with the initial owner.
6: Assign Moderator	1. Create a community (from previous test). 2. Set up a new member user. 3. Create a member connection for the new user. 4. Create a POST request to assign the member as a moderator. 5. Set request.user to the test user. 6. Call assign_moderator view directly with the request.	The member is promoted to moderator, and the response status code is 200.	True	Member user successfully promoted to moderator.	The member user is now a moderator in the community.

USE OF GENERAL LICENSES

Below licenses belong to used libraries and frameworks in this project.

1. **React:**

   https://github.com/facebook/react/blob/main/LICENSE

2. **Djongo:**

   https://github.com/doableware/djongo/blob/master/LICENSE

3. **Antdesign**:

   https://github.com/ant-design/ant-design/blob/master/LICENSE

4. **Luxon:**

   https://github.com/moment/luxon/blob/master/LICENSE.md

5. **Cryptography:**

   https://github.com/pyca/cryptography/blob/main/LICENSE

6. **Gunicorn**

   https://github.com/benoitc/gunicorn/blob/master/LICENSE

7. **Django Rest Framework**

   https://github.com/encode/django-rest-framework/blob/master/LICENSE.md