

IITB EKALAVYA SUMMER INTERNSHIP PROGRAMME - 2018

NOTIFICATION SYSTEM

FRG PROJECT REPORT



UNDER THE GUIDANCE OF
Professor. Deepak B. Phatak

Project In-Charge:

Mr. Nagesh Karmali

Mentor:

Ms. Nivia Jatain

Group Members:

Kanika Dhiman

Komal Chugh

T.Bharat Bhushan Reddy

Summer Internship 2018

Project Approval Certificate

Computer Science and Engineering

Indian Institute of Technology Bombay

The project entitled Notification System submitted by Ms. Kanika Dhiman, Ms. Komal Chugh, Mr. T. Bharat Bhushan Reddy is approved for Summer Internship 2018 Ekalavya programme from 15th May 2018 to 6th July 2018, at Department of Computer Science and Engineering, IIT Bombay.

Prof. Deepak B. Phatak
Dept of CSE, IITB
Principal Investigator

Mr. Nagesh Karmali
Dept of CSE, IITB
Project In-Charge

DECLARATION

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will cause a disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Ms. Kanika Dhiman
PDPM IITDM Jabalpur

Ms. Komal Chugh
IIT Ropar

Mr. T. Bharat Bhushan Reddy
NIT Calicut

Date : 6 July 2018

ACKNOWLEDGEMENT

We would like to express our profound gratitude and deep regards to our guide **Prof. D.B. Phatak** and Project In-charge **Mr. Nagesh Karmali**. They made us believe in ourselves and push our limits to emerge as a better software developer.

We would also like to thank **Mr. Abhijit Bonik** and **Ms. Firuza Aibara** for their exemplary guidance, valuable information and constant encouragement throughout the project. They were always there to show us the right track when needed help. With help of their brilliant guidance and encouragement, we all were able to complete our tasks properly and were up to the mark in all the tasks assigned.

We are also very thankful to our mentor **Ms. Nivia Jatain** for her valuable suggestions. During the internship, we got a chance to see the stronger side of our technical and non-technical aspects and also strengthen our concepts.

Last but not the least, we wholeheartedly thank all other colleagues working in different projects under Prof. D.B Phatak for helping us evolve better with their critical advice.

ABSTRACT

Notification System is a Fundamental Research Group Project aimed at providing notifications to the users of *Collaborative Communities*. Collaborative Communities is a system which allows individual users to form communities, participate in various activities defined in the community, and thus, enhance its value for the society in terms of education.

In the following sections, the design of this system is described along with the achieved results. Also, more insight about the used key technologies is given, justifying why they were necessary for the project. The details for building such a system with Python and its frameworks are explained and the necessary concepts for using it are introduced.

In this project, we successfully built a system which delivers notifications about the events relevant for the user. Two new modules namely, feeds and notifications have been added to the existing Collaborative Communities System.

Table of Contents

CHAPTER 1	9
1.1 Introduction	9
1.2 Purpose	9
1.3 Scope	10
CHAPTER 2	11
2.1 Motivation	11
2.2 Technologies Used	11
2.3 Requirements	12
CHAPTER 3	13
3.1 Overview	13
3.2 Architecture	14
3.3 Features	16
3.4 Implementation	27
3.4.1 Events	27
3.4.2 Types	31
3.5 Technical Specifications	35
CHAPTER 4	42
4.1 Selenium Testing	42

CHAPTER 5	44
5.1 Problems and Solutions	44
5.2 Future Scope	46
5.3 Results	48
CHAPTER 6	49
6.1 References	49

List of Figures

1	Flowchart for Notifications	14
2	Flowchart for Feeds	15
3	Before clicking on mark as Read	16
4	After clicking on mark as Read	16
5	Before clicking on mark as Unread	17
6	After clicking on mark as Unread	17
7	Before deleting a notification	17
8	After deleting a notification	18
9	Before clicking on "Mark all as read"	18
10	After clicking on "Mark all as read"	19
11	Before clicking on "Mark all as unread"	19
12	After clicking on "Mark all as unread"	20
13	Before clicking on "Unread notifications"	20
14	After clicking on "Unread notifications"	21
15	Before clicking on "Delete all read notifications"	21
16	After clicking on "Delete all read notifications"	22
17	Before clicking on "Delete all unread notifications"	22
18	After clicking on "Delete all unread notifications"	23
19	Before clicking on "Delete all notifications"	23
20	After clicking on "Delete unread notifications"	24
21	Unread Count	24
22	Clicking on a notification	25
23	Redirecting to a page	25
24	Automatically marked as read	26
25	Different states for community article	27
26	Different states for group article	29
27	Structure of a notification	38
28	Example of notification page	38
29	Paginator on notification page	39
30	Structure of a feed	39

31	Example of Community Feeds	40
32	Example of Group Feeds	40
33	Paginator on group feeds page	41

List of Tables

1	Events related to Community	28
2	Events related to Groups	30
3	Notifications	33
4	Community Feeds	33
5	Group Feeds	34
6	Status of test cases	43

CHAPTER 1

1.1 Introduction

A really important aspect of modern consumer web applications is a notification system. Notifications simplify user interaction and thus stimulate usage of system. Thus having a notification system can be an important lever for growth.

This project aims to handle notifications in Collaborative Communities portal. It forms an integral part of Collaborative Communities as it manages the task of keeping users of system updated about the events and activities related to him/her.

The process generally involves two stages, i.e. Creation of notification and Sending notification to appropriate recipient. When a user does some action, a notification is created by notification module which automatically decides the recipient and appropriate message, and then it sends the notification to the recipient. Recipient maybe a single user or a group of users.

1.2 Purpose

A fine-grained notification system is needed in order to allow any Collaborative Communities module to send any type of notification to any existing user of the system. By creating such a system all the users of the Collaborative Communities are notified about all the events and activities related to him/her.

The aim of this project is to develop a flexible, extensible real-time notification system that sends these notifications to the Col-

laborative Communities users. A real-time system is a system subject to a time constraint, performing an action within a period of time and as soon as possible. These systems are very popular nowadays because they meet the expectation of the users, who want immediate feedback when interacting with computers.

1.3 Scope

The Notification system is designed in such a way that it gives Collaborative Communities flexibility to add new types of notifications and also modify the existing ones very easily. In this project we have made efforts to provide notifications for all possible actions related to communities and groups.

CHAPTER 2

2.1 Motivation

- Notification System is an integral part of web applications which involve user interactions with the system.
- It gives a quicker way of providing information to the user or a group of users.
- It saves consumers time and automates the process of listing all actions related to user.
- For example, in Collaboration System, publisher of a community would be mostly interested in publishable content of that community. Notification System manages the task of providing all this information at a single place.

2.2 Technologies Used

- **Django:** Django is an advanced Web framework written in Python that makes use of the model view controller (MVC) architectural pattern and its key objective is to ease the development of complicated, database-driven websites. Collaborative Communities makes use of Django 1.11.7, two new django apps namely, feeds and notification have been added to the existing system which make use of below mentioned frameworks for the functioning of notification system.
- **Django Frameworks:** A framework provides structure upon which we can build our software easily and also that struc-

ture can be changed by the individual user accordingly. We have made use of the following django web frameworks:

1. **Django Notifications:** It is a general framework which stores and sends notification to a user based on his/her action. For example, it can be used to build Github like "Notifications".
 2. **Django Activity Stream:** It is used for generating and displaying streams of actions. For example, it can be used to build Github like "News Feed".
- **JavaScript:** JavaScript enables interactive web pages and thus is an essential part of web applications. We have used JavaScript for providing real time unread notification count to the user.
 - **AJAX:** Ajax allows content on Web pages to update immediately when a user performs an action, unlike an HTTP request, during which users must wait for a whole new page to load.
 - **MySQL:** MySQL is an open source relational database management system (RDBMS) based on Structured Query Language (SQL). Generated notifications and feeds are getting stored in MySQL database.

2.3 Requirements

- django-notifications-hq==1.4.0
- django-activity-stream==0.6.5
- django-jsonfield==1.0.1
- django-jsonfield-compat==0.4.4

CHAPTER 3

3.1 Overview

Notification System is divided into two sub-systems, one for sending personal notifications and another for feeds. Notifications and Feeds are similar in a sense that both are communication tools but feeds are common for all the community or group members whereas notifications are user specific.

In this project notifications are used to convey those messages to user which are personal and important to him/her. For example, in Collaboration System, when an Article gets published, Author of that article gets a personal notification saying "Your article xyz got published".

Feeds are basically used to convey messages which needs to be accessed by a group of users. The basic aim of feeds is to keep the users up-to-date regarding the events which happen in a community or a group of which he/she is a part of.

Now, the advantage of such bifurcation is that it doesn't mix up the information of different priorities which needs to be conveyed to the user, rather it helps in conveying all the messages in an organized and hassle-free way.

In collaboration system, 'Bell' icon in the navigation bar is used for displaying personal notifications whereas for feeds, a separate tab called 'Feeds' is used. This tab is visible in community/group page for community/group feeds.

3.2 Architecture

In a nutshell, notification system is all about storing data into the database and then retrieving and displaying it to the user. The detailed explanation of the frameworks used is as follows:

- **django-notifications**

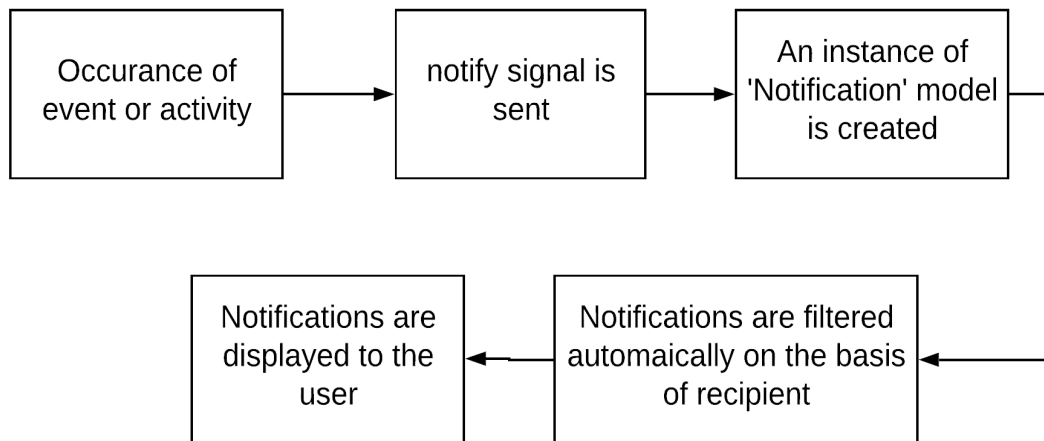


Figure 1: Flowchart for Notifications

Django notifications framework has been used for building notification system. Occurrence of an event is followed by triggering of `notify.send()` signal with appropriate parameters including recipient. This signal, in turn performs an action of saving the notification into database.

This framework provides a functionality of displaying notifications filtered on the basis of recipient i.e. a user can only see the notifications in which he/she is a recipient.

- **django-activity-stream**

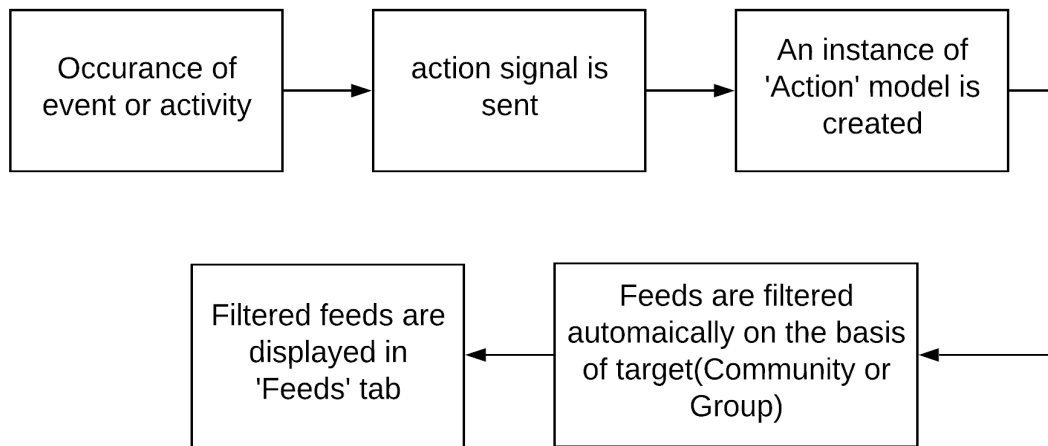


Figure 2: Flowchart for Feeds

Django activity stream framework has been used for building feed system. Whenever an event or an activity occurs, `action.send()` creates an instance of Action model based on the parameters given to `action.send()` signal. This signal, in turn performs an action of saving feeds into database.

Finally, community/group feeds are displayed by filtering on the basis of particular community/group.

3.3 Features

Notification system has the following features :

- **Mark as Read** : It marks a specific notification as read.

Before clicking on "*Mark as Read*" button the notification appears as shown below.

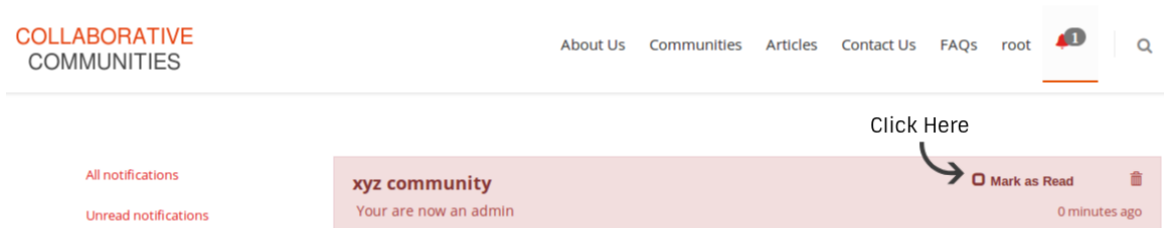


Figure 3: Before clicking on mark as Read

After clicking on "*Mark as Read*" button the notification changes its state to "Read".

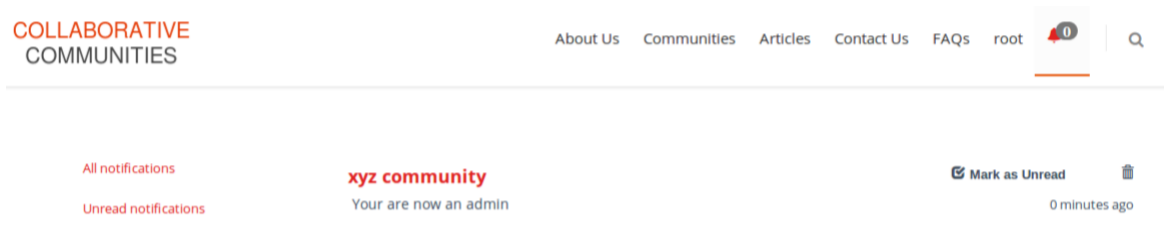


Figure 4: After clicking on mark as Read

- **Mark as Unread** : It marks a specific notification as unread.

Before clicking on "*Mark as Unread*" button the notification appears as shown below.

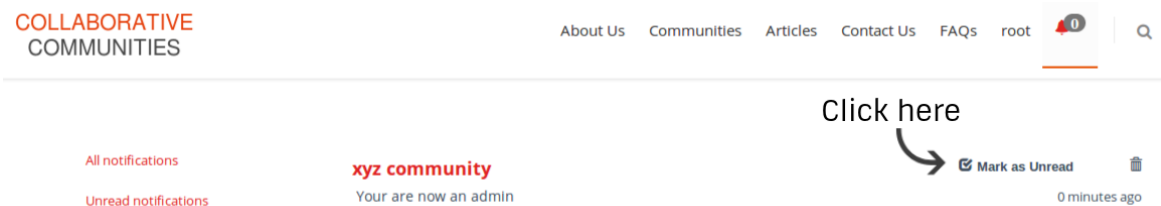


Figure 5: Before clicking on mark as Unread

After clicking on "*Mark as Unread*" button the notification changes its state to "Unread".

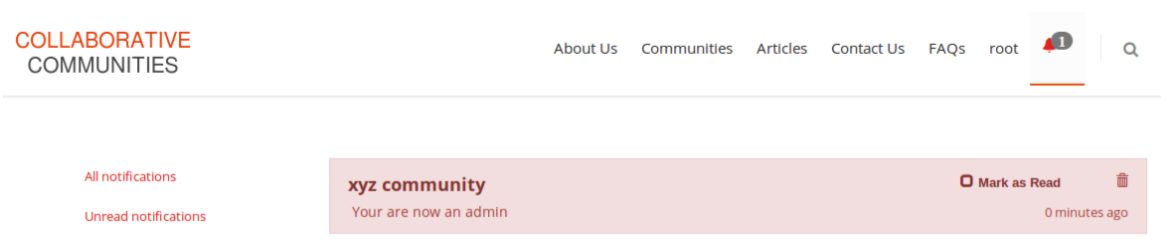


Figure 6: After clicking on mark as Unread

- **Delete** : It deletes a specific notification.

Before clicking on "Delete(Trash) icon" the notifications page appears as shown below.

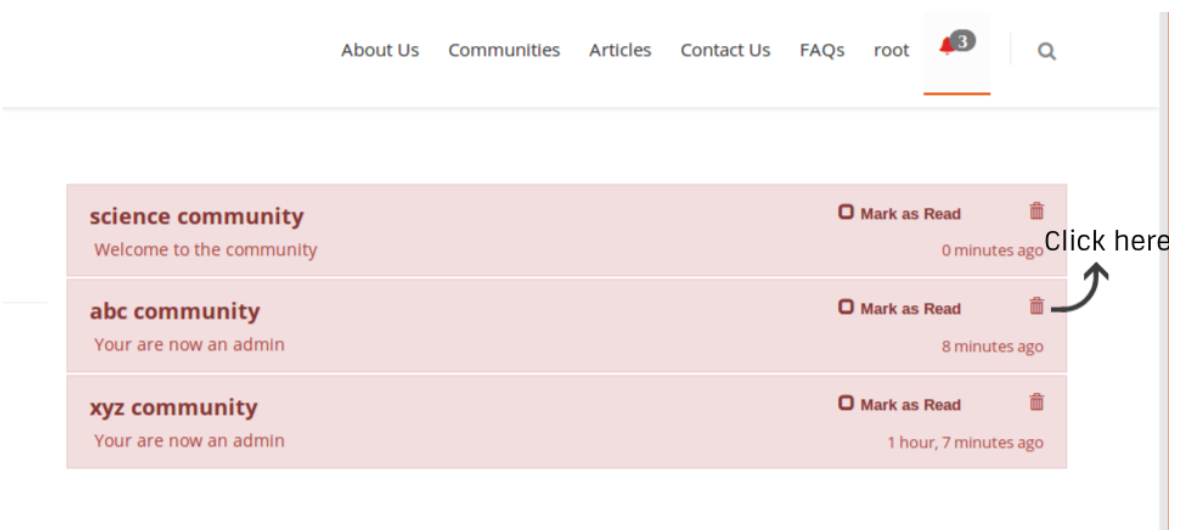


Figure 7: Before deleting a notification

After deleting the notification, the notifications page appears as shown below.

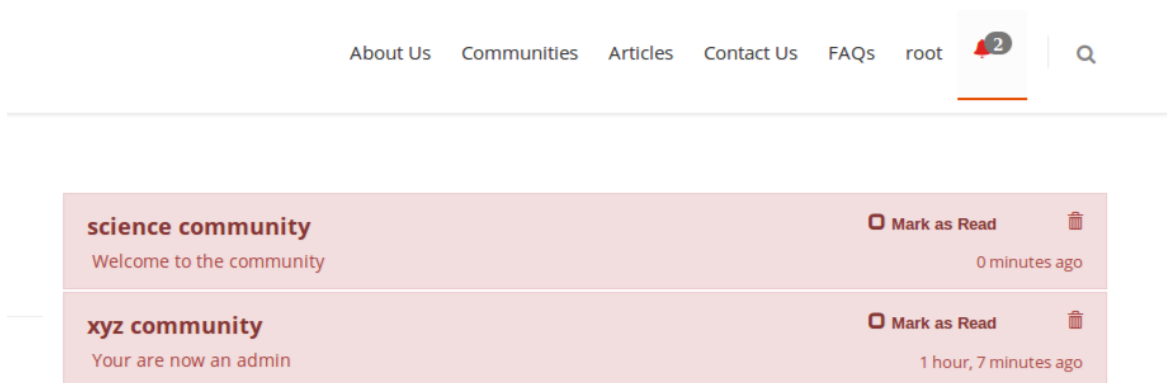


Figure 8: After deleting a notification

- **Mark all as Read** : It marks all the notifications as read.

Before clicking on "*Mark all as read*" button the notifications page appears as shown below.

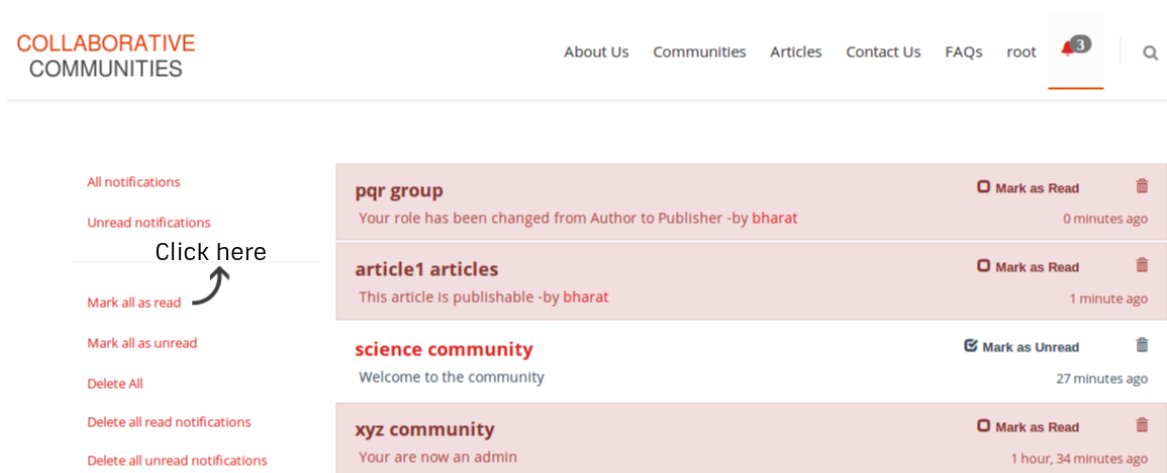


Figure 9: Before clicking on "Mark all as read"

After clicking on "*Mark all as read*" button the notifications page appears as shown below.

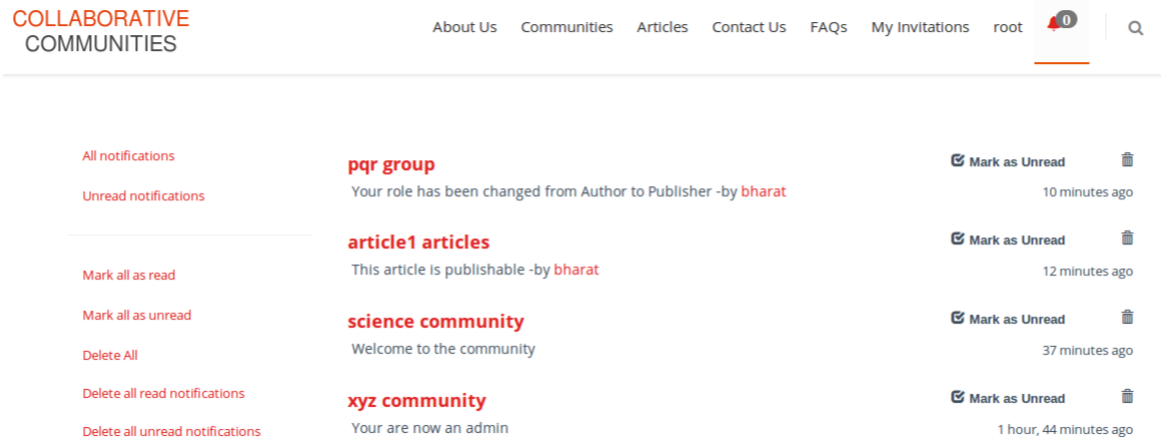


Figure 10: After clicking on "Mark all as read"

- **Mark all as Unread** : It marks all the notifications as unread.

Before clicking on "*Mark all as unread*" button the notifications page appears as shown below.

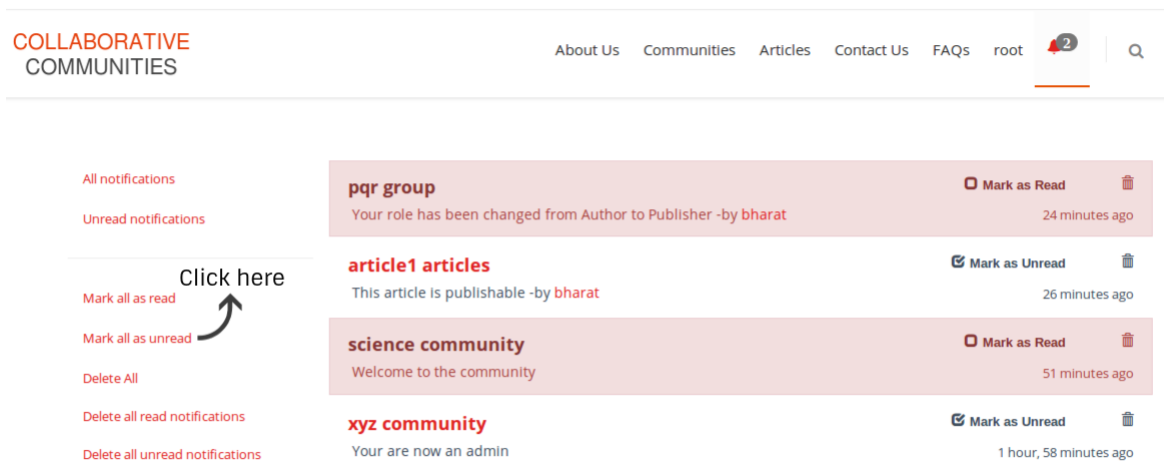


Figure 11: Before clicking on "Mark all as unread"

After clicking on "*Mark all as unread*" button the notifications page appears as shown below.

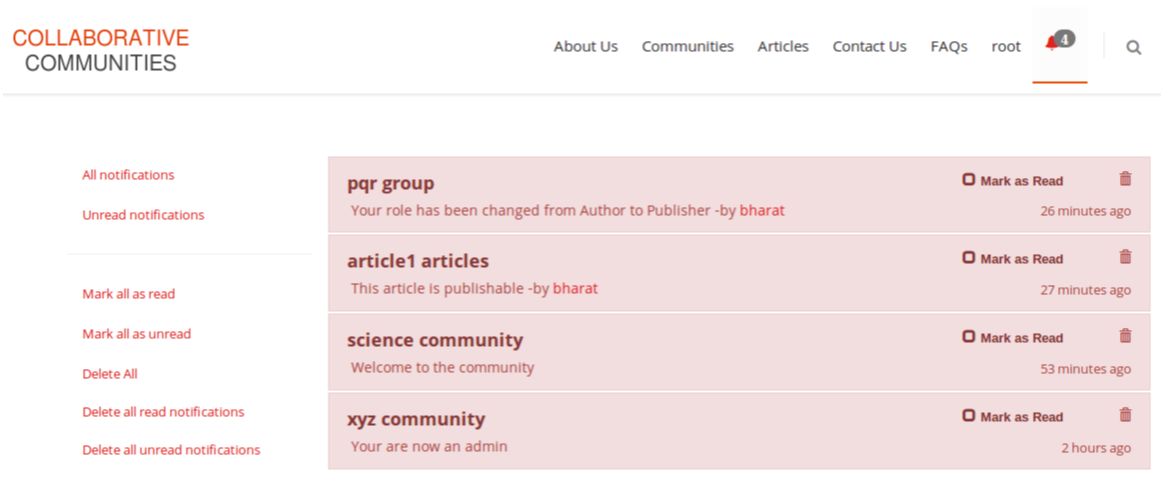


Figure 12: After clicking on "Mark all as unread"

- **View unread notifications** : It displays all the unread notifications to the user.

Before clicking on "*Unread notifications*" button the notifications page appears as shown below.

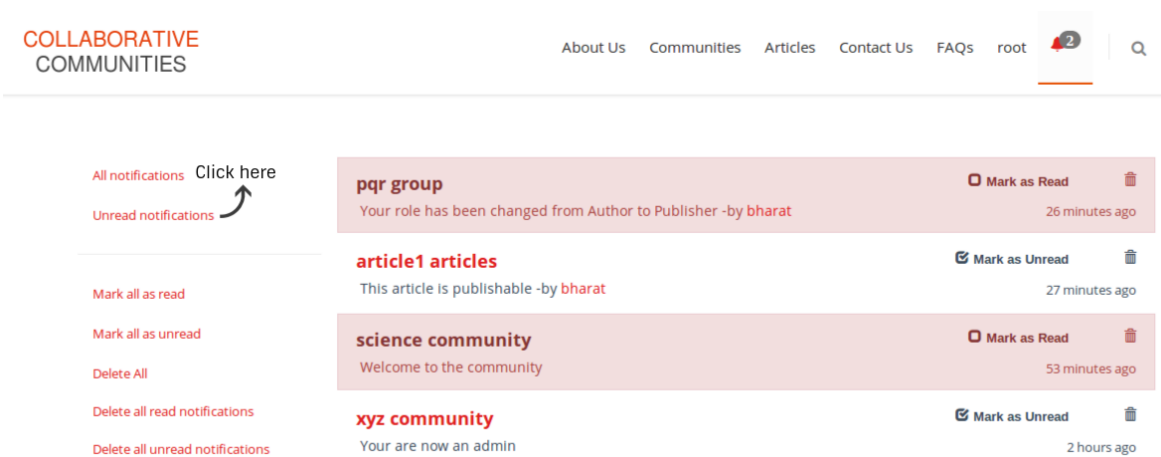


Figure 13: Before clicking on "Unread notifications"

After clicking on "*Unread notifications*" button the notifications page appears as shown below.

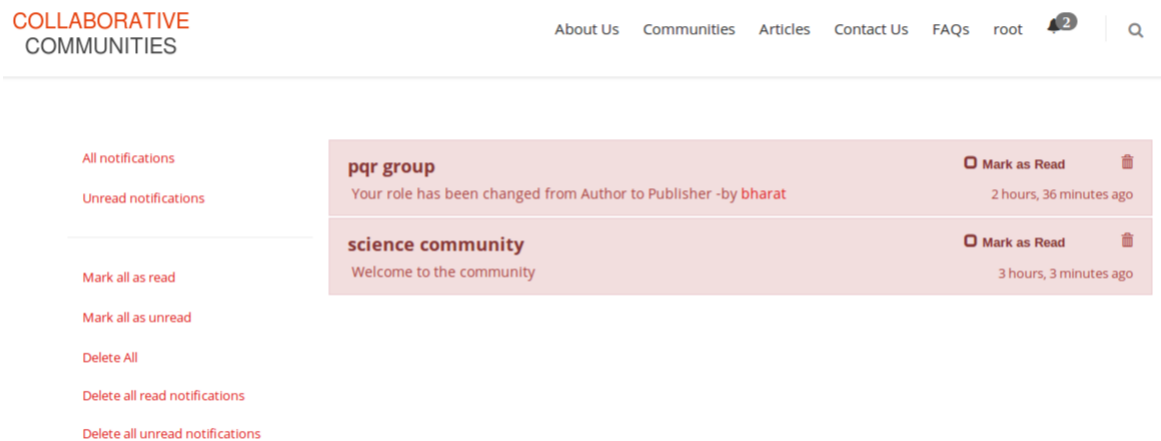


Figure 14: After clicking on "Unread notifications"

- **Delete all read notifications** : It deletes all the notifications which have been marked as read.

Before clicking on "*Delete all read notifications*" button the notifications page appears as shown below.

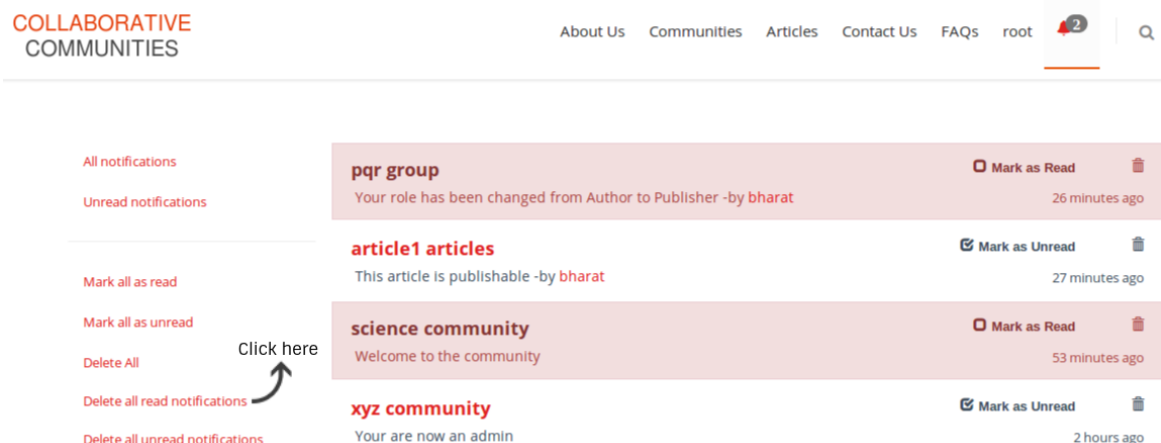


Figure 15: Before clicking on "Delete all read notifications"

After clicking on "*Delete all read notifications*" button the notifications page appears as shown below.



All notifications	pqr group Your role has been changed from Author to Publisher -by bharat 2 hours, 36 minutes ago	<input type="checkbox"/> Mark as Read 
Unread notifications	science community Welcome to the community 3 hours, 3 minutes ago	<input type="checkbox"/> Mark as Read 
Mark all as read		
Mark all as unread		
Delete All		
Delete all read notifications		
Delete all unread notifications		

Figure 16: After clicking on "Delete all read notifications"

- **Delete all unread notifications** : It deletes all the notifications which have been marked as unread.

Before clicking on "*Delete all unread notifications*" button the notifications page appears as shown below.

COLLABORATIVE
COMMUNITIES

About Us

Communities

Articles

Contact Us

FAQs

root

2

All notifications

Unread notifications

Mark all as read

Mark all as unread

Delete All

Delete all read notifications

Delete all unread notifications

pqr group

Your role has been changed from Author to Publisher -by bharat

2 hours, 36 minutes ago

Mark as Read

article1 articles

This article is publishable -by bharat

2 hours, 38 minutes ago

Mark as Unread

science community

Welcome to the community

3 hours, 3 minutes ago

Mark as Read

xyz community

Your are now an admin

4 hours, 10 minutes ago

Mark as Unread

Click here

Figure 17: Before clicking on "Delete all unread notifications"

After clicking on "*Delete all unread notifications*" button the notifications page appears as shown below.

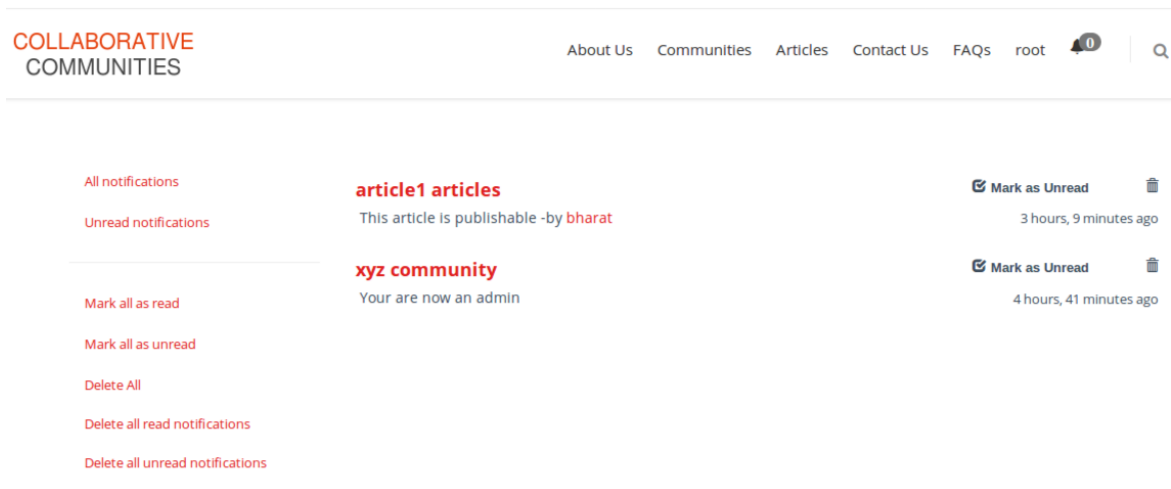


Figure 18: After clicking on "Delete all unread notifications"

- **Delete all notifications** : It deletes all the notifications.

Before clicking on "*Delete all notifications*" button the notifications page appears as shown below.

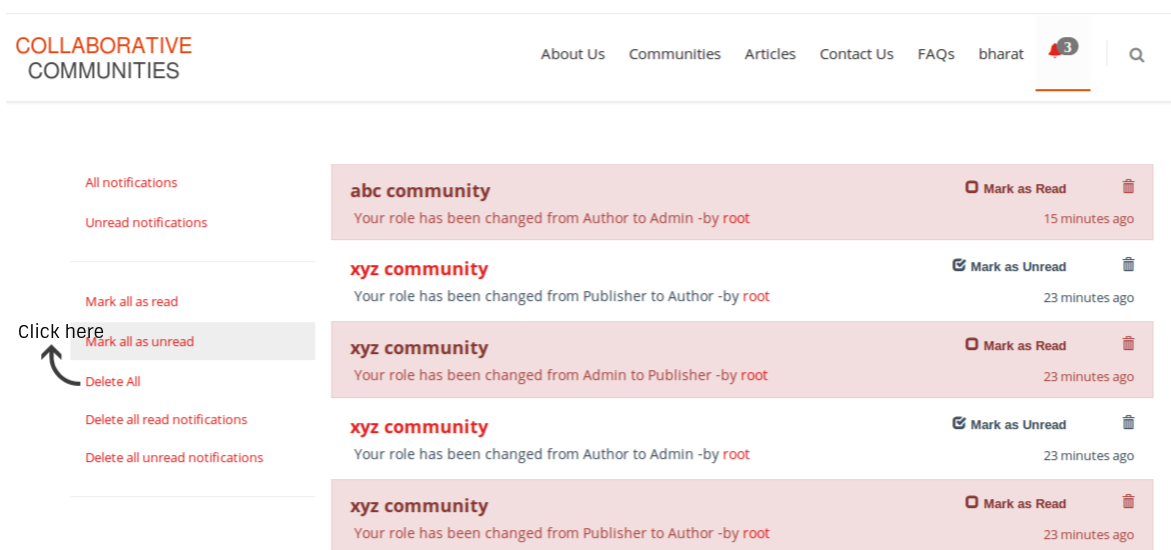


Figure 19: Before clicking on "Delete all notifications"

After clicking on "*Delete all notifications*" button the notifications page appears as shown below.

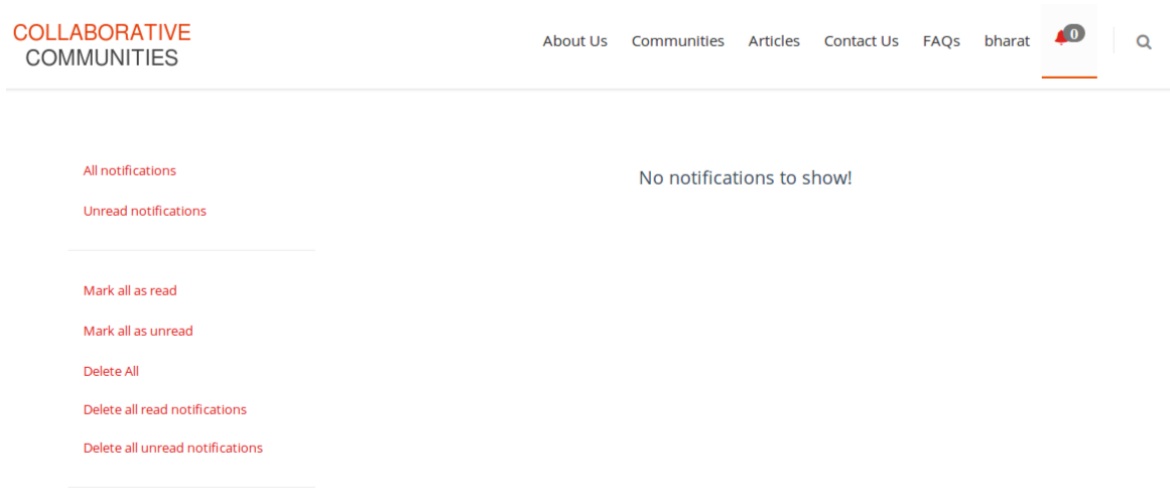


Figure 20: After clicking on "Delete unread notifications"

- **Unread Count** : It shows the number of unread notifications.

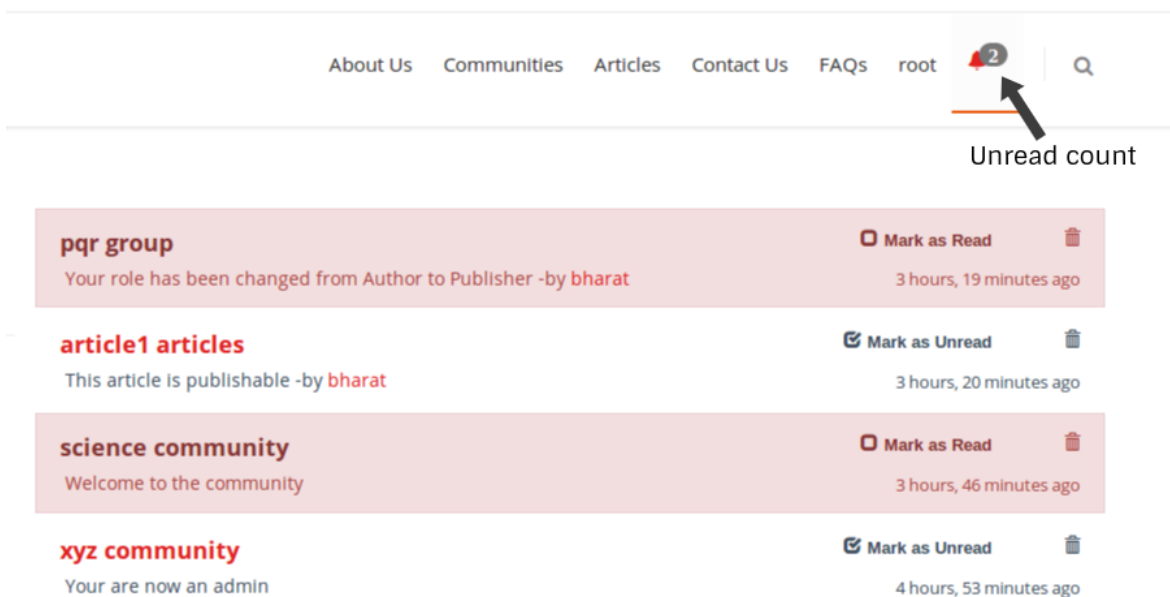


Figure 21: Unread Count

- **Linking** : When a user clicks on any notification he is redirected to corresponding community/group/article page and that particular notification is marked as read automatically.

Click here

All notifications

Unread notifications

Mark all as read

Mark all as unread

Delete All

Delete all read notifications

Delete all unread notifications

pqr group
Your role has been changed from Author to Publisher -by bharat
3 hours, 29 minutes ago
Mark as Read

article1 articles
This article is publishable -by bharat
3 hours, 30 minutes ago
Mark as Unread

science community
Welcome to the community
3 hours, 56 minutes ago
Mark as Read

xyz community
Your are now an admin
5 hours, 3 minutes ago
Mark as Unread


Figure 22: Clicking on a notification

After clicking on the notification he/she is redirected to a particular page.

Communities > abc > pqr


View Feeds Group Content

Redirected to corresponding group



pqr
Since: July 1, 2018 2 Members 0 Published articles
this is unique style

Top Contributors

Publisher 

Unsubscribe Create Article

Published Articles Other Groups Members

bharat root

Figure 23: Redirecting to a page

That particular notification is automatically marked as read.

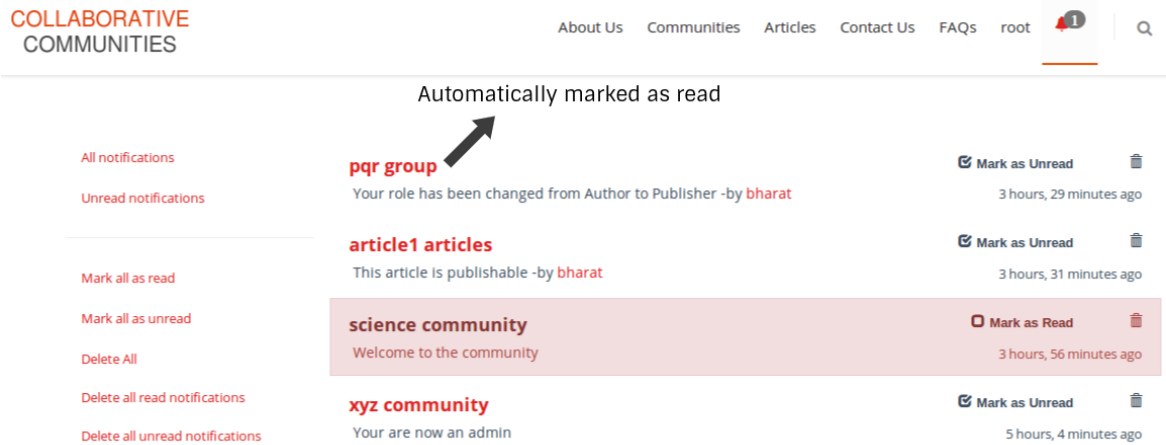


Figure 24: Automatically marked as read

3.4 Implementation

3.4.1 Events

Corresponding to an event, there is a notification and/or feed generated. So, it becomes mandatory to list down all the events occurring in *Collaborative Communities*.

We have divided the events into two broad categories i.e. events related to community and events related to group.

1. **Events related to Community:** The following table majorly includes events related to community articles and role management.

- **Community Articles:** A community article undergoes the following state changes before getting published.

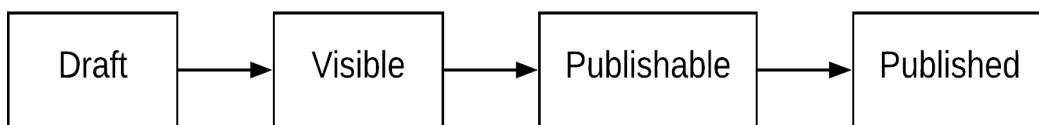


Figure 25: Different states for community article

- **Role Management:** A community user can either be an author, a publisher or a community admin.

In the following table, N* and CF* maps to the respective entries in the Notification (3), Community Feeds (4) tables.

Sno	Event	Notifications	Feeds
1	Article goes from draft to visible state	-	CF1
2	Article goes from visible to publishable state	N1	CF2
3	Article goes from publishable to published state	N2, N3	CF3
4	Article is edited by an author	N4	-
5	Article is edited by a publisher	N5	-
6	Article is edited by an admin	N6	-
7	User subscribes to a community	N16	-
8	User unsubscribes to a community	N17	-
9	Publisher unsubscribes to a community	N18	CF12
10	Admin unsubscribes to a community	N19	CF13
11	Role changed from author to publisher	N7	CF4
12	Role changed from author to admin	N8	CF5
13	Role changed from publisher to author	N9	CF6
14	Role changed from publisher to admin	N10	CF7
15	Role changed from admin to author	N11	CF8
16	Role changed from admin to publisher	N12	CF9
17	Author is removed	N13	-
18	Publisher is removed	N14	CF10
19	Admin is removed	N15	CF11

Table 1: Events related to Community

2. **Events related to Group:** The following table majorly includes events related to group articles and role management.

- **Group Articles:** A group article undergoes the following state changes before getting published.

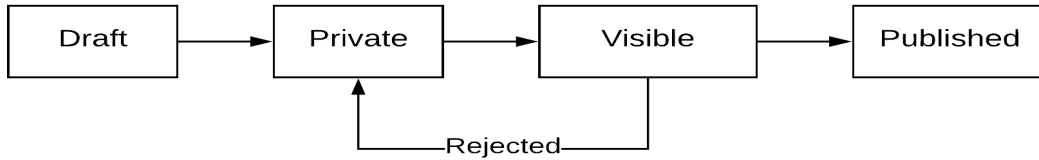


Figure 26: Different states for group article

- **Role Management:** A user can either be an author, a publisher or a group admin.

In the following table, N* and GF* maps to the respective entries in the Notification (3), Group Feeds (5) tables.

Sno	Event	Notifications	Feeds
1	Article goes from draft to private state	N20	GF1
2	Article goes from private to visible state	N21	GF2
3	Article goes from visible to published state	N22, N23	CF14
4	Article goes from visible to private state (Rejected)	N24, N25	GF1
5	Private state group Article is edited by a group author	N26	-
6	Private state group Article is edited by a group publisher	N27	-
7	Private state group Article is edited by a group admin	N28	-
8	Visible state group Article is edited by a community publisher	N29	-
9	Visible state group Article is edited by a community admin	N30	-
10	User unsubscribes to a group	N31	-
11	Publisher unsubscribes to a group	N32	GF12
12	Admin unsubscribes to a group	N33	GF13
13	Role changed from group author to group publisher	N34	GF4
14	Role changed from group author to group admin	N35	GF5
15	Role changed from group publisher to group author	N36	GF6
16	Role changed from group publisher to group admin	N37	GF7
17	Role changed from group admin to group author	N38	GF8
18	Role changed from group admin to group publisher	N39	GF9
19	Group Author is removed	N40	-
20	Group Publisher is removed	N41	GF10
21	Group Admin is removed	N42	GF11

Table 2: Events related to Groups

3.4.2 Types

The following tables show various types of notifications and community/group feeds implemented in *Collaborative Communities*.

1. Notifications

Sno	Actor	Recipient	Message	Target
N1	Community member	All publishers and admins of Community	This article is publishable.	Article Edit
N2	A Publisher or Admin of Community	Author of Article	Your article is Published	Article View
N3	A Publisher or Admin of Community	All publishers and admins of Community	This article is Published	Article View
N4	Author of Community	Author of Article	Your Article got edited	Article Edit
N5	Publisher of Community	Author of Article	Publisher edited your article	Article Edit
N6	Admin of Community	Author of Article	Admin edited your article	Article Edit
N7	Admin of Community	Author whose role was changed to publisher	Your role has been changed from author to publisher	Community
N8	Admin of Community	Author whose role was changed to admin	Your role has been changed from author to admin	Community
N9	Admin of Community	Publisher whose role was changed to author	Your role has been changed from publisher to author	Community
N10	Admin of Community	Publisher whose role was changed to admin	Your role has been changed from publisher to admin	Community
N11	Admin of Community	Admin whose role was changed to author	Your role has been changed from admin to author	Community
N12	Admin of Community	Admin whose role was changed to publisher	Your role has been changed from admin to publisher	Community
N13	Admin of Community	Author who was removed	You have been removed from the community	Community
N14	Admin of Community	Publisher who was removed	You have been removed as a Publisher	Community
N15	Admin of Community	Admin who was removed	You have been removed as an admin	Community
N16	User who Subscribes	User who Subscribed	Welcome to the Community	Community

N17	Author of Community	Author who left the community	You left the community	Community
N18	Publisher of Community	Publisher who left the community	You left the community as a publisher	Community
N19	Admin of Community	Admin who left the community	You left the community as an admin	Community
N20	Author of article	All publishers and admins of group	This group article is in private state, can be changed to visible	Article Edit
N21	Publisher or admin of group	All publishers and admins of Community	This group article can be published	Article Edit
N22	Publisher or Admin of community	Author of Article	Your article is published	Article View
N23	Publisher or Admin of community	All publishers and admins of Community	This article is published	Article View
N24	Publisher or Admin of community	Author of Article	Your article got rejected	Article View
N25	Publisher or Admin of community	All publishers and admins of Group	This article got rejected	Article View
N26	Group Author	Author of article	Your Article got edited	Article View
N27	Group Publisher	Author of article	Group publisher edited your article	Article View
N28	Group Admin	Author of article	Group Admin edited your article	Article View
N29	Community Publisher	Author of article	Community publisher edited your article	Article View
N30	Community Admin	Author of article	Community admin edited your article	Article View
N31	Author of Group	Author who left the group	You left the group	Group
N32	Publisher of Group	Publisher who left the group	You left the group as a publisher	Group
N33	Admin of Group	Admin who left the group	You left the group as an admin	Group
N34	Admin of Group	Author whose role was changed to publisher	Your role has been changed from author to publisher	Group
N35	Admin of Group	Author whose role was changed to admin	Your role has been changed from author to admin	Group
N36	Admin of Group	Publisher whose role was changed to author	Your role has been changed from publisher to author	Group
N37	Admin of Group	Publisher whose role was changed to admin	Your role has been changed from publisher to admin	Group
N38	Admin of Group	Admin whose role was changed to author	Your role has been changed from admin to author	Group
N39	Admin of Group	Admin whose role was changed to publisher	Your role has been changed from admin to publisher	Group

N40	Admin of Group	Author who was removed	You have been removed from the group	Group
N41	Admin of Group	Publisher who was removed	You have been removed as a Publisher	Group
N42	Admin of Group	Admin who was removed	You have been removed as an admin	Group

Table 3: Notifications

2. Community Feeds

Sno	Actor	Verb	Action Object
CF1	Article	Article is available for editing	Author of the Article
CF2	Article	This article is no more available for editing	Community member
CF3	Article	Article has been published	Publisher or Admin of Community
CF4	Author whose role is changed	Role changed from Author to Publisher	-
CF5	Author whose role is changed	Role changed from Author to Admin	-
CF6	Publisher whose role is changed	Role changed from Publisher to Author	-
CF7	Publisher whose role is changed	Role changed from Publisher to Admin	-
CF8	Admin whose role is changed	Role changed from Admin to Author	-
CF9	Admin whose role is changed	Role changed from Admin to Publisher	-
CF10	Community Publisher	Publisher has been removed	-
CF11	Community Admin	Admin has been removed	-
CF12	Community Publisher	Publisher has left	-
CF13	Community Admin	Admin has left	-
CF14	Article	Group article has been published	Author of Article

Table 4: Community Feeds

3. Group Feeds

Sno	Actor	Verb	Action Object
GF1	Article	Article is available for editing	Author of the article
GF2	Article	This article is no more available for editing	Publisher or Admin of Group
GF3	Article	Article has been published	Publisher or Admin of Community
GF4	Author whose role is changed	Role changed from Author to Publisher	-
GF5	Author whose role is changed	Role changed from Author to Admin	-
GF6	Publisher whose role is changed	Role changed from Publisher to Author	-
GF7	Publisher whose role is changed	Role changed from Publisher to Admin	-
GF8	Admin whose role is changed	Role changed from Admin to Author	-
GF9	Admin whose role is changed	Role changed from Admin to Publisher	-
GF10	Group Publisher	Publisher has been removed	-
GF11	Group Admin	Admin has been removed	-
GF12	Group Publisher	Publisher has left	-
GF13	Group Admin	Admin has left	-

Table 5: Group Feeds

3.5 Technical Specifications

1. **Sending signals:** Whenever an event occurs `notify.send()` and/or `action.send()` signal is sent along with required arguments. Given below is a detailed explanation of usage of these signals.

(a) **notify signal:** The arguments for `notify.send()` signal are as follows:

- sender - A user who triggers the event.
- verb - The verb phrase that identifies the occurred event.
- recipient - A user or list of users who will receive the notification.
- target - The object on which the activity was performed.
- target_url - Django named url of target. (JSON data)
- sender_url - Django named url of sender. (JSON data)
- sender_url_name - The username of sender. (JSON data)

Sender, Recipient are generic foreign keys to 'User' whereas Target is foreign key to any arbitrary object. In order to provide links in a particular notification, extra data is sent in JSON format.

target_url will change according to the target provided, sender_url will always be the same i.e. 'user_profile_view' and sender_url_name will always be the sender's username.

For example, for generating N1 notification, sender will be any community member, verb will be 'Welcome to

the Community', recipient will be list of all the community publishers, target will be a publishable article, target_url will be 'article_edit', sender_url will be 'user_profile_view' and sender_url_name will be username of sender.

(b) **action signal:** The arguments for action.send() signal are as follows:

- actor - The object that performed the activity.
- verb - The verb phrase that identifies the action of the activity.
- target - The object to which the activity was performed.
- action_object (Optional) - The object linked to the action itself.
- actor_href - Django named url to which the actor should be linked.
- actor_href_id - Id required by actor_href url.
- action_object_href (Optional) - Django named url to which the action object should be linked.
- action_object_href_id (Optional) - Id required by action_object_href url.

Actor, Action Object and Target are Generic Foreign Keys to any arbitrary object and so can represent any Django model. An action is a description of an event that was performed (Verb) at some instant by some Actor on some optional Target that results in an Action Object getting created/updated/deleted.

For this project, Target for community/group feeds is always a Foreign Key to the respective community/group. And actor_href, actor_href_id, action_object_href and

action_object_href_id are sent as **JSON** object by the framework.

For example, for generating CF1 community feed, actor is a Foreign Key to Articles model in which the Article is the one whose state has been changed from draft to visible, verb is a string "Article is available for editing", action_object is a Foreign Key to Users model in which the User is the author of the article, target is a Foreign Key to Community model in which the Community is the one whose feeds are being displayed, actor_href is "article_edit", actor_href_id is the id of article, action_object_href is "display_user_profile" and action_object_href_id is the username of the author.

2. **Retrieval:** Notifications are automatically filtered on the basis of recipient, whereas feeds are explicitly filtered on the basis of Target in this project.

```
community = Community.objects.get(pk=pk)
feeds = community.target_actions.all()
```

In the above code snippet, the filtering is done on the basis of target 'community'. So the feeds will be community specific.

3. Front End:

(a) **Notifications:** The following is the structure of a notification.

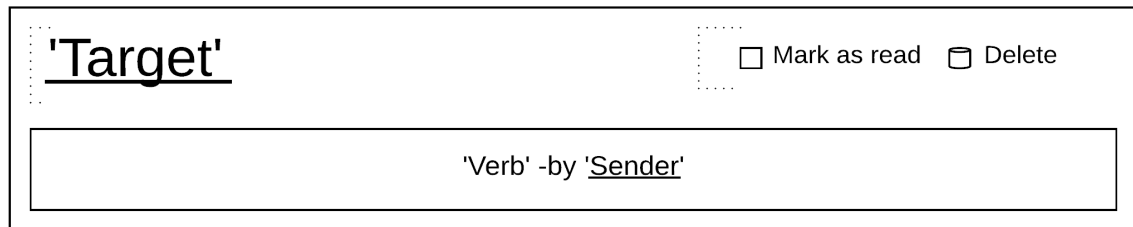


Figure 27: Structure of a notification

An example of notifications page is as follows:

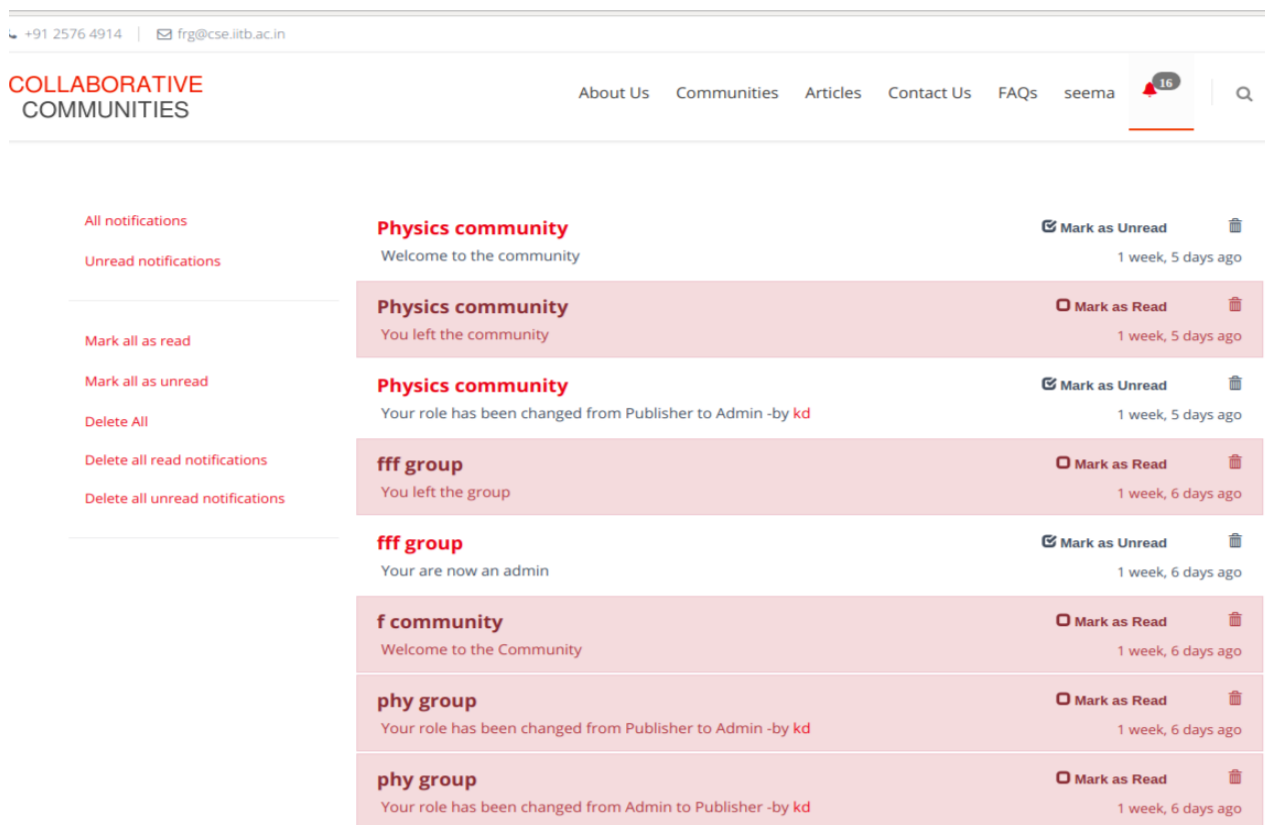


Figure 28: Example of notification page

Pagination has been added to the notification page with a count of 10 i.e. there will be 10 notifications per page.

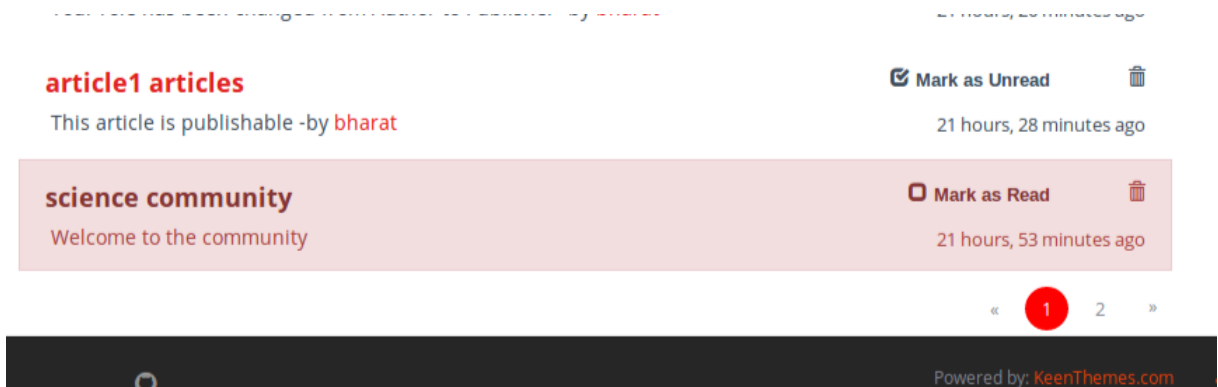


Figure 29: Paginator on notification page

(b) **Feeds:** The following is the structure of a feed.

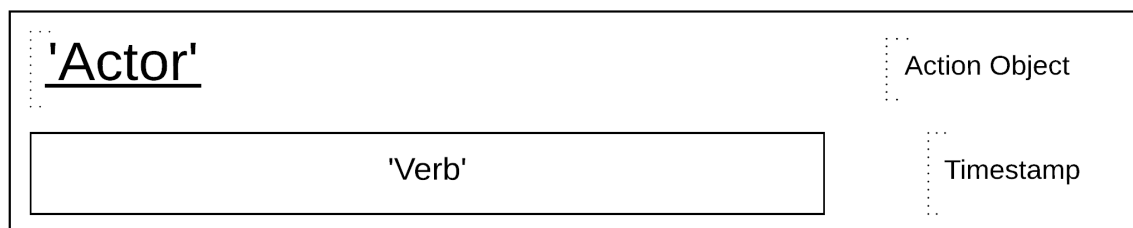


Figure 30: Structure of a feed

An example of community feeds is as follows:

COLLABORATIVE
COMMUNITIES

About UsCommunitiesArticlesContact UsFAQsroot

Communities > xyz

View

Manage Community


Update community Info

Community Content

Feeds


Group Content

Forums

bharat


Role changed from Admin to Publisher

0 minutes ago

root


Role changed from Publisher to Admin

0 minutes ago


root

Role changed from Admin to Publisher


2 minutes ago

IIT Bombay


Article has been published

root

3 minutes ago

IIT Bombay

This article is no more available for editing

root

4 minutes ago

Figure 31: Example of Community Feeds

An example of group feeds is as follows:

COLLABORATIVE
COMMUNITIES

About UsCommunitiesArticlesContact UsFAQskomal

View


Manage group

Update group Info


Feeds

Group Content


Group Feeds

FRG

This article is no more available for editing


komal

0 minutes ago

bharat


Admin has been removed

0 minutes ago


root

Publisher has left

2 minutes ago

FRG

Article is available for editing

bharat

5 minutes ago

Figure 32: Example of Group Feeds

Pagination has been added to the group feed page with a count of 5 i.e. there will be 5 group feeds per page.



Figure 33: Paginator on group feeds page

CHAPTER 4

4.1 Selenium Testing

Selenium is an open source automated testing suite for web applications across different browsers and platforms. It has been used for testing event logging system.

The common requirement for all test cases was logging in into system that required username, password which is taken as input, and other inputs as per the event requirements.

In order to test event logging system, data is to be extracted from the REST API endpoints which requires authentication. So, a token is to be generated which is used by selenium to call the REST API.

The approach used for testing all the test cases is to check count of total hits i.e. while testing an event, the total hits should be increased by 1, comparison of event name was made and the id of user/community/group/article was compared.

The link for detailed description of all the test cases is given below:

https://docs.google.com/spreadsheets/d/1kz9f4N6v0kvd_yiiWwsjBsFkH7ewgM4sC0IfmWg53Uw/edit#gid=0

The following is status of all the test cases:

Sno	Event Name	Status
1	event.article.create	Pass
2	event.article.view	Pass
3	event.article.edit	Pass
4	event.article.statusChanged	Pass
5	event.article.published	Pass
6	event.community.view	Pass
7	event.community.subscribe	Pass
8	event.community.unsubscribe	Pass
9	event.user.login	Pass
10	event.user.logout	Pass
11	event.profile.view	Pass
12	event.group.view	Pass
13	event.group.unsubscribe	Pass
14	event.community.create	Pass
15	event.group.create	Pass
16	event.group.manage	Pass
17	event.content.view	Pass
18	event.course.create	Pass
19	event.course.view	Pass
20	event.course.edit	Pass
21	even.course.manage	Pass
22	event.course.update	Pass
23	event.comment.post	Pending

Table 6: Status of test cases

CHAPTER 5

5.1 Problems and Solutions

- **Problem 1:** The major problem we faced during this project was making the notification system real-time which means a user gets updated about the new notifications without going to the notification page.

Solution: We made live updater notification badge which shows the unread notification count. That means, without refreshing the system, badge count will increase by the number of new notifications since the last refresh. For its implementation, django-notification's live updater API has been used.

```
{% load notifications_tags %}
{% register_notify_callbacks badge_class='badge'
    refresh_period='5' callbacks='fill_notification_badge' %}
<li>
    <a class="glyphicon glyphicon-bell"
        href="{% url 'notifications:all' %}">
        {% live_notify_badge badge_class='badge' %}
    </a>
</li>
```

- **Problem 2:** In Django-notifications:
 1. In file **list.html**, the default layout of displaying notifications was not suitable according to our requirement.
 2. In file **notification-tags.py**, the default layout of badge was square in shape, not properly aligned according to the bell icon.
 3. In file **models.py**, the default alert-class for both read and unread notifications were same, not as per our theme.

4. In file **views.py**, few functionalities i.e. mark all as unread, delete all read/unread were missing.
5. In file **urls.py**, notification page required authentication.

Solution: All of these files were changed accordingly which are present in temp\notifications folder of Collaboration System, to be replaced in python packages. Note: file **settings.py** was missing while installing the django-notifications and to be copied from temp\notifications folder.

5.2 Future Scope

1. Implementing real time notifications using Django Channels:

- Currently, the badge which displays the unread notification count shows real time updates.
- This is implemented using a javascript API which updates specific fields within django template by making HTTP Requests to server at regular intervals of time.
- This technique is not very feasible as it consumes lot of resources by making many unnecessary requests to server.
- This problem can be overcome by use of WebSockets Communication Protocol.
- By default django supports only HTTP protocol, but it can be extended to handle WebSockets protocol with the help of Django Channels.
- The current system can be extended using similar technique to deliver real time notifications.

2. Generating Notifications based on reputation System:

- Presently the system does not provide notifications to users about events which occur as a result of his/her reputation.

3. Paginator while using "*Delete*" functionality:

- While using delete functionality, paginator for notifications page had an issue.

For example, let's say we have a total of 13 notifications, 10 of which are on first page and 3 are on second. When we delete a notification from first page it is expected that we will have 10 notifications on first page and 2 on second. But currently unless we refresh the page it's showing 9 notifications on first page and 3 on second.

5.3 Results

In this project, Notification System has been successfully implemented for *Collaborative Communities*.

Notification System along with other projects has been merged and is being maintained under 'develop' branch of Collaborative Communities repository. Link is given below:

`"https://github.com/fresearchgroup/Collaboration-System/tree/develop"`

The source code of Collaborative Communities along with Notification System is given below:

`"https://github.com/kanikadhiman17/Collaboration-System"`

Event Logging System System has been successfully tested using Selenium. Source code for the test cases can be found below:

`"https://github.com/Bharat-Reddy/Collaboration-System-Selenium/tree/eventlogs"`

CHAPTER 6

6.1 References

- [1] Simple is better than complex, **Django Tutorial**,
"<https://simpleisbetterthancomplex.com/series/beginners-guide/1.11/>", Accessed on 20th may 2018.
- [2] **Django documentation**, "<https://www.djangoproject.com/>", Accessed on 23rd may 2018.
- [3] **django-notifications** framework, "<https://github.com/django-notifications/django-notifications>".
- [4] **django-activity-stream** framework, "<https://github.com/justquick/django-activity-stream>".
- [5] Hashnode blog post, **Facebook feed architecture**,
"<https://hashnode.com/post/architecture-how-would-you-go-about-building-an-activity-feed-like-facebook-cioe6ea7q017aru53phul68t1>",
Accessed on 24th may 2018.
- [6] **Pagination in Django**, "<https://simpleisbetterthancomplex.com/tutorial/2016/08/03/how-to-paginate-with-django.html>",
Accessed on 1st june 2018.
- [7] **AJAX**, "https://www.w3schools.com/xml/ajax_intro.asp", Accessed on 15th june 2018.
- [8] **Selenium Testing**, "<http://selenium-python.readthedocs.io/>", Accessed on 23rd june 2018.