Improve stream management and message viewing in web app

Google Summer of Code proposal 2021 - Zulip

Basic Information

Name	**Purushottam Tiwari**
Email	**purushottam.tiwari.cd.cse19@itbhu.ac.in**
Github handle	**m-e-l-u-h-a-n**
Nationality	Indian
University	Indian Institute of Technology (BHU), Varanasi
Degree	IDD(B.Tech + M.Tech), (2019-2024)
Contact	(+91)8955316471
Timezone	Indian Standard Time (UTC +5:30)

About Me

I am a second year student enrolled in Computer science and Engineering department of Indian Institute of Technology, (BHU) Varanasi. I have a keen interest in web development and it has increased since I started contributing to open source. I am always open to learning new technologies in any field related to computer science and try to apply my learnings in efficient ways to achieve good results.

I have worked on several projects, and working on them gave me practical experience and knowledge in frontend development frameworks like Vue.js, React.js, and backend frameworks like Django, Django Rest Framework, Express, Nodejs. I have experience in programming with various languages like Python, C, CPP, Javascript, Dart, and Bash.

Why Zulip?

Zulip is where I have learned true disciplines of open source contribution and I want to learn more by getting involved in bigger projects in order to make Zulip even better. I have been part of the community for more than 5 months from now and have found amazing help from the community for whatever problems I faced. I never felt myself confused about where to ask for my doubts(thanks to the awesome chat.zulip.org community). Zulip also has a great documentation that helped me to get started flawlessly. So Zulip is the only organization, I am applying for in this GSOC.

Contributions to Zulip

- **All pull requests opened by me**
- **All pull requests by me (status: open)**
- **All issues opened by me**

Merged/Closed Pull requests		
Sr. No.	**Title**	**Status**
1	Add error codes in validate_account_and_subdomain	Merged
2	markdown: Refactor backend logic for handling user mention	Merged
3	documentation(api): minor type fix	Merged
4	markdown mention bug	Merged
5	Alignment of presence circles in typeahead	Merged
6	Extend mention syntax	Merged
7	Clean force argument in tooling	Merged
8	Increase spacing for lists after paragraph in message view	Merged
9	Add user presence circle in mention typeahead	Merged
10	users: Clarify readability issues related to access_user_by_id	Merged
11	Add api documentation for /display	Merged
12	Remove vertical scroll bar for alert box	Closed
13	Migrate various backend tests to use assert Logs	Merged
14	Refactor mock.patch to use assertLogs	Closed (both commits merged)

Open Pull requests		
Sr. No.	Title	Status
1	Video call providers documentation	Open
2	Add user groups as permission option for posting	Open
3	[WIP] Make user profile picture adjustable	Open
4	Clean up realm_bots remove and delete events	Open
5	Add administrators bot type	Open
6	Attempt to translate search suggestions	Open
7	stream-settings: Add support for subscribing all members of a user group to stream.	Open

Issues Opened by me		
Sr. No.	Issue No.	Title
1	#17535	Inline mention for users, groups and streams fail if they come after an invalid mention
2	#17111	Clarify readability issues over access_user_by_id
3	#16828	Add an assertNoLogs() context manager to the ZulipTestCase

Proposal Description

Overview

My overall aim while working as a GSoC student for zulip, would be to improve the zulip web app so that its stream management, message viewing becomes easier and efficient which would be beneficial for both small and large organisations using Zulip. For this I would like to work on various issues like refining left-sidebar, improving stream management and enhance message viewing.

1. Finish up implementation of default stream groups

This feature aims at providing organizations with options to enable users to subscribe to different groups of streams, based on their interest when they are joining the organization.

Issue: #13670

This issue is explained in great detail by Tim on the issue thread itself. As mentioned in the issue description, various works in the backend side have been done for this issue. I think I can proceed with this issue in the following steps:

- Working on UI for creating and managing ***DefaultStreamGroup***.
- Merging ***DefaultStream*** and ***DefaultStreamGroup*** to decide a consistent design for the signup page.
- Working on UI to provide users with options for selecting different stream groups.
- Then there are scopes of extensions to these features about making it usable for users to subscribe to different groups even after joining the organization.

These steps are already mentioned in the issue description by Tim in the same or in some modified form. I have mentioned them here in an organized form in which I wish to complete the work on this feature. I will discuss it thoroughly enough when I start working on this.

2. Clean Up stream edit modal

Zulip's current stream editing view looks very confusing and cluttered. So the main aim for this task would be to improve the stream edit experience so that frequently used features like checking/adding subscribers are easily accessible.

Issue: #9382

Zulip's current stream settings page looks like a cluster of various settings just somehow arranged together. I will try to clean up the settings modal and redesign it so that important settings are quickly visible. There has been a lot of discussion about this previous summer in #design > add user to stream which somewhat points that stream settings need to have tabbed view with ***Personal settings***, ***Stream settings***, and ***Subscribers*** split in their individual tabs.

I will discuss with the community about any other feedback about design needed for this as I go for implementing this.

3. Unmuting certain topics within a muted streams

This task aims at adding a feature for having some topics unmuted in muted streams.

Issue: #2517

This has been pending for a long time mainly because of the lack of design/method that should be followed for solving this, and also to have feedback on various possible use cases for this. Some initial thoughts could be to have a ***Topic*** table which facilitates a lot of this kind of features. Another method that could be used for this is to have all the topics except for selected ones to be muted but stream is not marked muted in this case and the left sidebar shows the unread count for only those messages that are from unmuted topic.

I think there would be more discussion needed for this. I will surely put it up on chat.zulip.org to have more ideas about it.

4. Stream level wildcard mention permission

Aim of this task would be to introduce a new stream level permission set, that deals with clarification of fact that who is allowed to use wildcard mention in a stream.

Discussion thread: #feedback > Stream-level control over wildcard mentions

There has been some feedback about the current way in which we allow limiting wildcard mention in large streams. The reason for confusion about this was missing clarification about how actually a large stream is decided for this feature.

After some discussion the plan was to introduce a new stream level permission set with name as ***wildcard_mention_policy***. The default could be kept as ***None*** in which case the stream is expected to inherit organization level policy value. I will surely discuss it in detail while going into implementations for this.

5. Allow selected user groups to post in announce-only streams

Aim for this task would be to have a setting that allows members of a selected user group to post in announce-only streams.

Issue: #9951

I have already done some work in PR #17300 for creating streams with ***stream_post_policy*** that allows only admins and members of selected user groups to post messages. So most of the work would be to complete all the refactors needed for its review so that after merging #17300 we can achieve it by setting the appropriate ***stream_post_policy***.

6. Improve private message view

Aim for this task is to solve the problem of unintentionally marking the first unread message as read in a private message narrow, by making use of the Recent Topics view for this.

Issue: #13510
Related Issue: #10886
Discussion thread: #design > Private message view design

There had been several feedbacks that opening private messages unknowingly marks some of the messages as read and requests to improve private message view in general. I have proposed about utilizing recent topics to address this feature request. From the discussion I had earlier in the discussion thread following is a rough stepwise plan for this.

- Extend recent topics to support listing pms.
- Make a new filter to list only private messages in the list.
- Working on left sidebar to improve its design for private message

I have done some prototype work on the first step, which enables private messages to get listed in the recent topic. There is some polishing work that needs to be done after which it will be ready for opening a pull request.

For the second step we can have an attribute on the table row as ***data-message-type*** with values as ***private*** *or* ***stream*** which could be utilised for filtering. This is just a rough first hand approach towards solving this. I will surely put it up for discussion on an appropriate thread to agree upon a better design.

Work on the third step mostly deals with redesigning the left-sidebar to support listing private messages in a better way. From the current discussion on #design > Private message view design about it the idea was to open Recent Topic view with a private message filter(made in second step) activated.

7. Support storing multiple contiguous blocks of history

Aim for this task would be to support multiple ***message_list_data*** data structures to support caching multiple contiguous blocks of message history for performance reasons.

Issue: #16697

This issue is a part of efforts required to enhance the performance of web-app when switching the view to different narrow filters. Issue #15131(Cache multiple MessageListData objects) is the task preceding this and there is already a PR #16746 from **@ryanreh99** regarding this. So in case it's not merged by that time I would coordinate with him to finish up PR #16746 and any follow up needed for this.

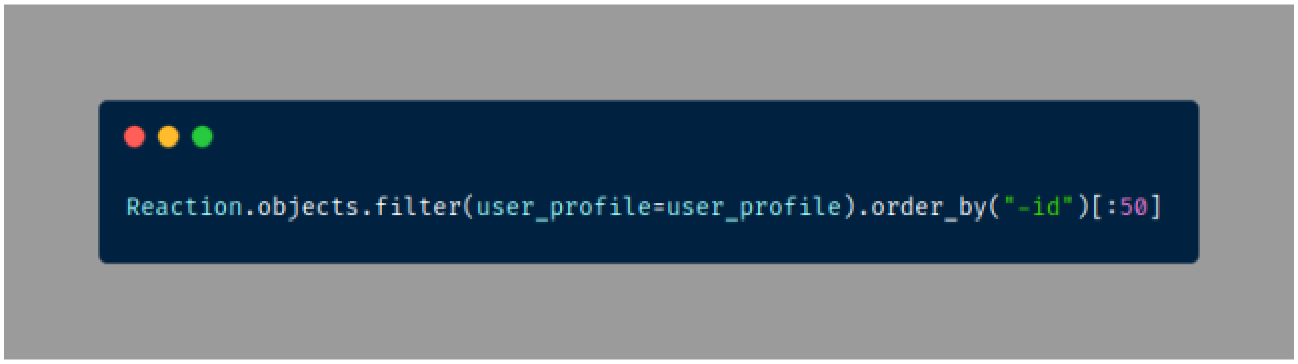
Then for this specific task, Tim has mentioned some initial thoughts about supporting multiple ***message_list_data*** objects for this, but it may require more discussion before going into implementations. So in next steps I would discuss the approach mentioned in the issue description and also any other ideas for this and then I would go with implementation for this issue.

8. Recent Emoji feature

Emoji pop-over should show recently used emoji by a user.

Issue: #13240

This issue has been discussed on github issue thread and on #general > Recent reactions #13240 and the approach for this as suggested by Tim is to use id of reaction for getting recent reactions of a user. Initial discussion had concerns about database queries required for supporting it, but we can fetch it in a single query in following way:



```
Reaction.objects.filter(user_profile=user_profile).order_by("-id")[:50]
```

For the UI part we can utilize zulip's current emoji popover with an extra tab for recent emojis.

9. Improve model for supporting more number of image previews

Aim for this task would be to modify the way we show image previews in messages, and as a result probably increase the number of allowed image previews.

Issue: #17925

We basically want to have a scrollbar kind of arrangement in the message where we could show previews of the images. Issue description has clear thoughts on why this is required. The approach here would be to utilize ***rendered_markdown.js*** as messages are passed through it before rendering to update any dynamic content in them so it would be easy to start for first hand prototyping of the idea.

Using ***rendered_markdown.js*** we can wrap the row of images in a div which supports horizontal scrolling to allow the user to see all images. Based on the feedback and discussion on the prototype, further outline for solving this can be planned.

Project Timeline

Period	**Timeline**
Community Bonding Period (17^{th} May- 6^{th} June)	
17^{th} May- 6^{th} June	1. Coordinate to get pending PRs merged and complete any immediate follow up needed for them. 2. Discuss with my mentor and community about my approach for Improve private message view and try to finish parts of it. 3. Coordinate to complete PR [#16746(Cache multiple MessageListData objects.)] and discuss ideas for Support storing multiple contiguous blocks of history. 4. Work on Improving private message view.
Phase 1 (7^{th} June - 16^{th} July)	
7^{th} - 14^{th} June	1. Finish implementation for Improve private message view. 2. Work on “Support storing multiple contiguous blocks of history” based on previous discussions. 3 Start discussions about a better design for stream settings modal and start working on it based on the ideas.
15^{th} June- 30^{th} June	1. Try to complete most of the design work for stream settings design work in this period so that new settings can be implemented on changed design. 2. Finish the implementation of “Support storing multiple contiguous blocks of history”. 3. Work on recent emoji feature after some confirmations about the approach I mentioned above.

1st July- 16th July	1. Discuss and complete the work on the default stream group feature. 2. Finish implementing recent emoji feature. 3. Discuss approaches to be followed for “Unmuting certain topics within a muted streams” 4. Iterate on reviews or any follow ups needed for work done till this time, and polish up pending works for first phase evaluations.
First Evaluation (12th - 16th July)	
Work Period (16th July- 16th August)	
16th - 31st July	1. Work on “Unmuting certain topics within a muted stream”. 2. Work on “Allow selected user groups to post in announce-only streams”. 3. Discuss and prototype “Improve model for supporting more number of image previews”.
1st - 23th August	1. Finish implementations of “Unmuting certain topics within a muted stream” 2. Work on “Improve model for supporting more number of image previews” based on discussions and feedback on the prototyping work on it. 3. Work on “Stream level wildcard mention permission”. 4. Iterate on reviews or any follow ups needed for work done till this time, and polish up pending works for final evaluations. 5. Discuss and start working on the future aspect of the default stream group about making it usable in general for any user to join multiple streams.
Final Evaluation 16th - 23rd August	

Past Experience

I am a member of Club of Programmers (COPS), IIT(BHU). COPS is a group of students in our college where students share their knowledge about different technologies in order to learn from each other, it also organizes various hackathons and coding events to enrich programming culture. I joined the club in my first year and learned a lot from my seniors by working and contributing to various projects of the club. It is through the seniors of this club that I came to know about open source contributions as well as about Google Summer of Code.

Working with various personal as well as projects of the club (some of which I maintain now), I have got very good experience in working with Django, Express, and frontend frameworks like Reactjs and Vuejs. I currently maintain the projects in the club for which I was a core developer, and try to pass whatever I learn to my juniors in the club while continuing my learnings at the same time.

Projects

I have worked on various projects some of which are made individually and others are made in groups.

1. Shopify (Source code)

This project uses Django at its core. It was developed as part of CSE-205N (ITW-2) course taught in third semester at our college. It is an e-commerce website that uses django in backend and Django templates in frontend. In order to address responsiveness and design issues it uses MD Bootstrap. The purpose of this application was to demonstrate efficient database querying using Django ORM. At the time of final submission it supported following noticeable features:

- Cart and order summary before checkout.
- Saving both delivery and billing information for future use.
- Adding coupons for special purchases.
- Suggestions for items of similar category.
- Payment simulation using stripe test cards
  (As stripe does not offer these services in India.)

It is this project that gave me a broad understanding of how Django ORM works, how templating systems can be utilised efficiently for dry code. Also it was praised to be one of the best submissions for the lab of the course, both in terms of design and implementation.

2. Peer IO (Source code)

This project heavily uses Javascript as it is built using (MongoDb + Express) at backend and Vuejs at frontend. It was built during a 24 hour hackathon I recently participated into. The idea behind developing the application was to have some way of increasing peer learning among students even during online semesters.

It does so by enabling students to anonymously review and provide feedback on work of their peers. Reviewing can be done only after submissions deadlines are over. Reviews remain anonymous meaning that neither reviewer comes to know whose work he/she is reviewing nor the reviewee comes to know about who has reviewed his/her work. This is done to prevent students from putting comments based on personal grudges and abuse cases.

It was an amazing experience working in such a project with really tight deadlines. We managed to finish as second runner ups for the hackathon.

3. Hackalog

- **Frontend source code**
- **Backend source code**

I developed this project as a member of COPS, IIT(BHU). It is a web application that supports hosting of hackathons and dev sprints. Actually COPS organizes such events with an aim of testing whatever is taught to the new members of the club. For such an event we required a platform to organize events in a fair manner. So we decided to build our own platform for this.

This project gave me a huge learning opportunity and practical experience on working with Django Rest Framework for backend and Reactjs for frontend. We also successfully organized a dev sprint for new members of the club after launching it in February. In the next release for this we are planning to improve the admin interface so that anyone can host such events.

Availability

I will be having my end semester exams in the first week of May and will be over by 12th of May well before community bonding period. So I will be completely free during the entire GSOC period without any other commitments. Therefore I can easily give around 35-40 hrs per week.

I would also inform about any sudden engagements(if any) and can easily cover up by working more in preceding and succeeding weeks.

Post GSoC

I wish to contribute to Zulip with the same eagerness even after GSoC. I recently learned about Github actions and various automation related things. So I'll also try to expand my area of contribution from Zulip web to other projects, like Zulip archive and python zulip api.