Pleroma API based GNU social WebApp frontend and client – GSoC 2020

Project Repository:	notabug.org/dadada/gnusocial-fe
Name:	Susanna Di Vita
E-mail:	susanna.divita.2@gmail.com
Location:	Genova, Italy (GMT+2 Rome Timezone)
Website:	github.com/SusannaDiV | notabug.org/susdiv
IRC & XMPP:	susannadiv
Proof of Competence:	notabug.org/susdiv/gnusocial.network

Mentor’s page

Abstract

This project aims at implementing a fully featured Pleroma API WebApp client and developing a Pleroma FE based Frontend for GNU social's v3 - a communication software used in federated social networks.
It comprehends both a completely new Pleroma FE-like frontend for improved accessibility and user interaction and a fully functional endpoint calling implementation for each and every Pleroma API functionality, developed through pleroma.site's Mastodon and Pleroma-based backend.

This new Pleroma FE-like design's improved accessibility will now hopefully help spreading the word of Free Software by making it more attractive to those familiar to traditional mainstream social network platforms - expectantly helping to build a good first impression for potential newcomers to the federated network. All but the emoji reactions, emoji counters, reply counter, status search and direct message timeline endpoints are currently working – this is because these functionalities are not currently working in pleroma.site's Pleroma API backend. I have implemented them anyways so that both the emoji reacting, direct timeline viewing and reply counting will be working as soon the Pleroma API backend they depend will be fixed – this additional implementation was not part of my original project plan and will aid future contributors to the project by saving them the time-consuming hassle of endpoint implementation for the Pleroma API.

Table of Contents

Preface

My mentors' attentive yet autonomy-respectful approach has been found to be a very effective guiding force during both the first tentative slants of the community bonding period and the 3-month long successful completion of this project. Diogo's passionate, informative and detail-oriented explanations managed to let me gain deeper knowledge in GNU social's inner workings and open-source contribution policies, while Phabluto's savvy and targeted feedback helped me realize the right direction to which to take the project in its initial stages. I'll be sure to treasure their teachings in all my future open-source contributions.

Introduction

This project is a fully-functioning Pleroma FE and API-based WebApp made for GNU social - a communication software used in federated social networks. It comprehends both a completely new Pleroma FE-like frontend and a fully functional endpoint calling implementation for each and every Pleroma API functionality made through pleroma.site's mastodon-pleroma-based backend.

This project encompasses all Pleroma API's offered functionalities, as I have personally implemented from scratch all the endpoints offered by the API – both Mastodon's generic and Pleroma.site's specific subset. Here is a list of all the employed endpoints (the relative explanations can be found in the State of The Art's "Pleroma API client WebApp endpoints implementation" subsection):

- User Actions: /api/v1/accounts/verify_credentials, /api/v1/accounts, /api/v1/apps, /oauth/token, /oauth-callback, /api/pleroma/captcha, /api/v1/accounts/${id}, /api/v1/accounts/update_credentials, /api/v1/notifications, /api/v1/notifications/clear

- ProfileTile Actions: /api/v1/accounts/${this.props.profileId}/follow, /api/v1/accounts/${this.props.profileId}/unfollow, /api/v1/accounts/${this.props.profileId}/block, /api/v1/accounts/${this.props.profileId}/unblock, /api/v1/accounts/${this.props.profileId}/mute, /api/v1/accounts/${this.props.profileId}/unmute, /api/v1/accounts/${this.props.profileId}/followers, /api/v1/accounts/${this.props.profileId}/following

- Timelines: /api/v1/timelines/public?local=true&only_media=false&count=20&with_muted=true', /api/v1/timelines/home?count=20&with_muted=true, /api/v1/favourites, /api/v1/timelines/public?local=true&only_media=false&count=20&with_muted=true&visibility=direct, /api/v1/statuses/${userId}/context (for the In conversation, parent status-specific timeline)

- Status-related actions: POST/DELETE /api/v1/statuses/(${id}) with either status or commentData, /api/v1/media, /api/v1/statuses/${screamId}/favourite, /api/v1/statuses/${screamId}/unfavourite, /api/v1/statuses/${id}/reblog, /api/v1/statuses/${id}/unreblog

- Emoji counter and reactions + Status search: /api/v1/pleroma/statuses/${this.props.scream.id}/reactions/${emojiObject.unified}, /api/v2/search

All but the emoji reactions, emoji counters, reply counter, status search and direct message timeline endpoints are currently working – this is because these functionalities are not currently working in pleroma.site's Pleroma API backend. I have implemented them anyways so that both the emoji reacting, direct timeline viewing and reply counting will be working as soon the Pleroma API backend they depend will be fixed – this additional implementation was not part of my original project plan and will aid future contributors to the project by saving them the time-consuming hassle of endpoint implementation for the Pleroma API.

State of the Art

The employed technology for this FE & client project has been the ReactJs framework, as thanks to its component-based frontend designing approach most objects and functions have been made reusable, without having to worry about redundancy due to JS' innate library import and redux-based re-routing.

Frontend UI Design and Implementation for improved UX and User Interaction

Pleroma.site's offered UI results in a bubble-tiled, old-timey two-column structure; the left side is cluttered with less than relevant, non-user specific sections that could be easily omitted in most of the timelines (namely the Home, Public, Favorites and Direct Messages), while the timeline's main content is relegated to the admittedly less attention grabbing column. This layout goes against all UI visual hierarchy best practices, and therefore will be completely replaced by the content-centric three-column design. This results in an indistinctive, non-content-centered design that focuses more on customizable tweaks than on post readability and user interaction.

Moreover, in order to comply with the F-pattern reading techniques unconscionably followed by most left-to-right user readers and therefore ameliorating the overall UX, all call-to-action elements like the ProfileTile and Timeline Navigation tiles will be placed at the top left and right margins of the screen, instead than being regrouped in a single clump as the previous implementation did.

Additionally, in order to bring a sense of prioritized order within the elements themselves, the background will be neutrally-colored and separated from the user's sight within the framework. This will provide a blank canvas on which to arrange the sections, now perceived as easily-recognizable and eye-catching instead of too-heavily loaded.

Finally, Pleroma FE's suboptimal placement of call-to-action elements like buttons and drop-down menus will be corrected as follows in accordance with the Z-pattern design standard - in order to redirect the user's focus on the timeline's main content after the first-contact horizontal-search scanning phase.

1. Added a completely new Navigation Bar – containing the Home, Favorites, Messages, hidden Search bar, Profile/Logout fields and the unified Mentions/Favorites/Retweets/Followings/Replies Notification drop-down menu with an option to mark all the present notification as read. The presence of a navbar will help streamline the interface by creating a starter point for user's browsing, thanks to newly introduced universally recognizable icons.

   Moreover, space management will be improved as such addition would eliminate the need for the user-specific section in the left column, making the following ProfileTile and Statistics sectors more visible.

   Not to mention, in the current Pleroma FE implementation the purpose of a set-to-visible Notification cloud gets defied by making users scroll to see the earlier notifications in the list.

2. Like/Comment/Retweet/In conversation redesign – obtained through CUD-

friendly redundant color/darkness/saturation coding. Such distinct and colorful gradient separation of status-specific user-employable actions will result in a more space efficient relevance-ranking system.

The currently employed dimension-importance ratio technique clogs a good portion of the left status-side, forcing the In conversation functionality to be relegated in the less visible three-dotted top-right status menu.

3. UI Timeline redesign - In order to lead users down the content page organically, this project had to implement new efficient strategy for elemental hierarchy; essential navigation page elements will be stored in the left column, with different grayscale shades to highlight their hover state; the former Notifications section, now hidden, has been moved in the navbar to space saving and improved readability.

   The right column now stores the ProfileTile section instead – containing different buttons in regards of being in some other User's Timeline or in one of the logged in user's own (Favorite and Edit if in Home/Public/Favorite/Direct Messages, Follow/Unfollow, Direct Message, Block/Unblock and Mute/Unmute if in the User's), followed by the Statistics sector. This new separation technique will also bring out the heavily text-based content of the Public Timeline, not to mention offer ameliorated clarity in regard to following/followers/statuses counters + clickable lists of users.

4. Follower and Following list Redesign – in order both to maintain consistency throughout the design and to reduce first-approach fatigue, the standard unordered list common to both the Followers and Following clickable user list has been replaced with an easily readable and catchier tile design, which users listed in a hidden, open-when-clicked tile that now appears right under the ProfileTile & right before the Statistic tile (as frequently occurs due to lack of proper CSS nesting). Such hidden section contains either the list of the users that are currently following or are current being followed by the logged in user's account; it is hidden so to avoid wasting user-focusing space (that used to occupy more two entire timelines in the user's page) that is now assigned instead to the main content.

5. Button Redesign – they now have more recognizable icons and text descriptions, changed to raise the UI's recognition potential and to minimize recall fatigue on the user's end. The added color-coded contrast helps action buttons to stand out and have their function immediately recognized by users – while breaking the content's grey scale monotones and allowing a catchier design implementation.

6. Added hover background focus and improved general responsiveness of the UI - for a modernized, accessible look. This provides users with immediate feedback on their actions, while also allowing for device-optimized layout changes thanks to the extensively documented CSS Grid usage. Mobile-friendly features - such as horizontal-scrolling avoidance, consistent zoom-free text readability and adequate spacing for tap targets - have been implicitly implemented.

7. Added Home, Favorites, Direct Messages and Profile shortcuts in the Timeline navigation sections – to optimize interaction flow and to endorse user-side exploration. The aim was to improve design immediacy and to make

both content and features available for discovery; in turn, this helps avoiding disengagement once a user has begun working with maximum efficiency. Integrating the Profile and Logout options within the Avatar drop-down menu button in the top right corner helps achieving both a UI free from friction-inducing clutter and a better free space management in the left column.

8. Drop-down Notices Menu implementation – it contains notifications coming both from the Favorites and from the Replies Timelines, with the correct corresponding conjugation. Having all notifications stored in one place improves time responses to other user's action and therefore increase user interaction. Users are able to recognize the favorited/retweeted/replied to/mentioned/followed content from the first line of displayed text – just as in pleroma.site. A second drop down menu is be present in the ride side of the navbar for Account-related options – namely, the Profile and Logout buttons.

9. Login and Signup pages implementation – the current login and registration can only be accessed through a barely visible, non-intuitively placed all-in-one sector. This not only makes it difficult for new users to locate the starting point of their registration process, but it also takes them to an unmodified, old-fashioned timeline layout. My separate Login and Signup pages (the latter equipped with captcha verification) complete with a more directly approachable form containing all login options.

10. Introduced a simplified Edit Profile page by regrouping related sectors. The Avatar, Name and Bio fields have been all made editable through the Edit Button present in the logged in user's ProfileTile – such preferences, when set, will be shown in addition to the unmodifiable and clickable user handle, and the followers/following/status count.

Pleroma API client WebApp endpoints implementation

This part of the project – completed during the final month of August – aimed at implementing all the Pleroma API endpoints used in pleroma.site, succeeded in completing this full-sack project by making all the previously introduced functionalities from the newly created frontend work.

Timeline endpoints

Adding the parameter **with_muted=true** to all timeline queries will also return activities by muted (not by blocked) users. This will be employed in the Public and Direct Messages Timelines, as they contain either conversations from a wide range of instances – both local and remote - not directly connected to the user that might still be of interest or posts where the logged in user has been explicitly specified as a recipient. The Favorite Timeline will only contain posts from muted users that have been favorited by the logged-in user's, while the Home Timeline will be deprived of them. This is seen as a protectory measure against timeline flooding from particularly active followed users. All posts from all timelines are presented in order.

By adding the parameter **visibility** to the timeline queries will instead exclude the statuses with the given visibility types specified in the input array. This will be employed in all but the Home and Profile Timelines, as to exclude private and direct conversations the user's not part of (e.g. **visibility=direct**). In all Timelines but the User ones, the post making component is visible.

Home Timeline (also known as "Personal Timeline of <name of user>")

The Home Timeline contains both the logged in user's direct and public messages and all the posts that have been made, retweeted or replied to by any of the followed users. This is the first Timeline that the users see once logged in; it is possible to access it by either clicking on the GNU social logo in the navbar's top-left corner or the following Home button.

All code pertaining to the Home Timeline can be found in the /src/pages/home.js file, the implemented endpoint being **/api/v1/timelines/home/ only_media=false &count=20&with_muted=true**.

Public Timeline

The Public Timeline contains all the posts that have been published with the highest accessible scope visibility (public) from all around the world in the Pleroma community. It is the first and only accessible page for logged out users and first-time visitors – as all the other left-menu options point to the Login page, and only a restricted version of the navbar is visible (containing only the Public and Login buttons). When logged in, the Public page is still accessible by clicking on the homonym left-menu option. Posts made by muted users are also visible.

The implemented endpoint for the Public Timelines is the following: **/api/v1/timelines/public?local=true&only_media=false&count=20&with_muted=true'**

User Timelines

User Timelines hold all the posts that have been made by the given user and that either have their accessibility scope set to public or that are direct messages with the visiting logged in user as one of the recipients. The contents of the ProfileTile will change to contain their avatar, name, bio, following/followers/status count, the list of all the explorable following/followers' profiles (when clicking respectively on the following/followers icon) and 4 different buttons (3 of whom double-edged): the follow/unfollow, the direct message, the block/unblock and the mute/unmute. The Statistics tile will also self-update to show the given user's ID and registration date. Code pertaining to the User Timeline can be found at **/src/pages/user.js**, and all user data (both status and ProfileTile wise) is received by the endpoint **/api/v1/accounts/${id}/statuses**, with the id parameter being the user's id.

Favorites Timeline

The Favorite Timeline, accessible from both the navbar, the left side-menu option, and the shortcut button in the logged in user's ProfileTile as stated priorly, encompasses all the posts that have been favorited by the logged in user. The statuses are returned as an array and the timeline is only visible for logged in users.

The implemented endpoint for the Favorites Timelines can be found at /src/pages/favorites.js and is the following: **/api/v1/favourites**

Direct Message Timeline

In addition to the user's id parameter, I have also specified two additional parameters: **visibility** and **with_muted.** The prior excludes the notifications for activities with the visibilities present in the input array (which may contain one or more of the following options: public and direct). This parameter will be set accordingly, whether it will be used in the logged-in user's Direct message Timeline's implementation or in the generic user's Profile Timeline (which contains both public and direct mentions; the latter only if the recipient matches the logged in user's id). It is important to note that Direct Messages cannot be retweeted, as otherwise the retweeted status would have public visibility on the retweeter's timeline. Since Direct Messages can only be observed on the sender's and recipients' timelines, they can only be replied to by the tagged recipients – leaving them out of the Public Timeline as they are completely confined to the concerned user's Home Timelines.

In my endpoint implementation I have also decided to include all the Direct Messages coming from muted – by the logged in user's – accounts; this is because of the high number of topic-targeted bot accounts that are however led by active human users in the Pleroma community. The maximum number of results to return had to again be explicitly set to the default value, same as with the only media option, in order not to incur in lacking timelines if the default current settings ever get updated.

Current endpoint implementation for Direct Message Timeline includes prior explained options with the following values:
pleroma.site/api/v1/timelines/public?local=true&only_media=false&count=20&with_muted=true&visibility=direct.

As stated before in the previous State of the Art section regarding the frontend, the logged in user's Direct Message Timeline can be visible through clicking on the correspondent navbar/left-menu button. In order to change the scope of an in-the-making post, it is instead necessary to click on the Direct Message toggle – that will change its color from burgundy to dark grey to signal its activation, briefly followed by the visible-when-hovered-upon text tip.

User Actions endpoints

Show Notifications and mark them as read

These endpoints both mark all the present notifications as read and show all the possible types of notifications (favorites/replies/retweets/mentions/followings) for the logged in user, respectively. The Notification button present in the navbar includes the notifications for activities with the types reckoned by the input array (which may contain one or more from the following: mention, follow, reblog, reply and like). All types of notices are displayed and categorized in the navbar drop-down, and the easily recognizable and labeled checkbox on top allows to mark all notification as read. The response contains the notification entity – or the array of notification entities - that were read.

The implemented endpoint for both the Notification list and the Mark Notification as read can be found at /src/components/layout/Notifications.js and /src/actions/userActions.js; they are the following: **/api/v1/notifications** and **/api/v1/notifications/clear**.

Update Profile

This endpoint (**/api/v1/accounts/update_credentials**) has been by me employed to make the editing of the user's publicly visible name, biography and avatar possible. All this user specific information is stored within the ProfileTile for both the logged in and external user, and can be easily modifiable by clicking on the EditProfile button present in the ProfileTile (only the logged in user's, as only the user itself has the necessary token permissions to modify their account). By clicking on it, a pop-up editable form containing name & bio text boxes and a button for file-picking the avatar are present. The avatar, once set, will be visible both in each of the made/repeated/replied to statuses by the user, their ProfileTile and the top right corner of the logged in user's navbar (which will open the Profile/Logout menu if hovered and redirect directly to the Profile page when clicked).

Profile/Account – Login & Registration
/api/v1/accounts/verify_credentials, /api/v1/accounts, /api/v1/apps, /oauth/token, /oauth-callback, /api/v1/accounts/${id}, /api/pleroma/captcha

The api/v1/accounts/:id endpoint was by me employed to retrieve the logged in user's or visiting user's profile information, that immediately gets stored in the corresponding ProfileTile. The statuses posted to said given account are instead returned by the api/v1/accounts/${id} endpoint, The api/v1/accounts endpoint is used to create a user and account records; it returns an account access token for the app that initiated the request. The api/v1/accounts/verify_credentials is a test endpoint I've employed to make sure the token works. The captcha (whose original Pleroma FE component I had to completely re-write to allow for better UX) endpoint returns a new captcha. It requires no parameters and the response is a provider specific JSON, in which the only guaranteed parameter is type. The oauth/token endpoint was employed to obtain an access token, which corresponds to the token endpoint (for more information view section 3.2 of the OAuth 2 RFC), while the oauth-callback represent what has to get invoked after OAuth authorization for the connected app (see /src/pages/signup.js and login.js for implementation).

Status-related action endpoints

Post Status – with Media Attachment and Direct Message option

These endpoints allow the logged in user to post a new status – either pure text or with a media attachment – that will either be present in the Public Timeline and the timeline of the followed by users (if the Direct Message toggle has been left on its default value false) or only in the tagged recipient's timelines and notifications (if the Direct Message toggle has been set to true by clicking on it – even signaled by the changing of both the title and the toggle color, to dark grey). Submitted posts are always visible in the Public Timeline and the Home Timeline of the logged in user; additionally, if the logged in user posting is being followed by other users, the Home page of those who follow it will encompass it as well; same goes for who favorites them (as they will appear in their Favorite timeline). The post making component is made visible in every timeline but the user specific ones, and encompasses a text both with character count, a submit button with a loading circular progress component, the direct message button and both the file-picker attachment button and the emoji picker. The ability to add emoji to text or media post is currently hindered by the API's errorful implementation, but I added it for posterity as well as the emoji reactions and counters. The endpoints that have been implemented to employ the previously described functionalities are: **POST /api/v1/statuses** with the statuses variable set and **/api/v1/media**.

Delete Status

The delete status endpoint allows the logged in user to delete their own-made status. When a status has been made by the user, the orange outlined Delete button appears on the top-right part of the status, right under the date. When clicked, a pop-up component asking whether you are sure to delete the post will appear; when clicked on the positive answer, the delete request described in this endpoint will be sent: **DELETE /api/v1/statuses/${id}**.

Favorite/Unfavorite a status

These endpoints let the logged in user mark a given status as either favorite (when the button is burgundy) or unfavorite (when the button has changed its color to dark grey). It also encompasses a like counter, that keeps track of the number of all the Pleroma users that have favorited the given status – and that therefore comprehend it in their Favorites Timeline. All the statuses visible to the logged in user can be favorited (and therefore also unfavorited), as the statuses only show on their timeline when their visibility setting allows it. These endpoints' (**/api/v1/statuses/${screamId}/favourite** and **/api/v1/statuses/${screamId}/unfavourite**) complete implementation can be found in the /src/components/scream/LikeButton.js file.

Retweet/Unretweet a status

These endpoints allow the logged in user to reblog/unreblog a given status on their timeline without adding any additional information if not the name of the user they reblogged it from (if the user desires to add a personal commentary, it qualifies as a reply – which is accessible through the In conversation button). When retweeting, a new status with the same visibility scope (public) will be created on your personal timeline (and therefore be visible in both the Public and logged in user's or followers' Home Timeline) with the red caption "Retweeted form <name of user>". Not all the statuses that are visible on the logged in user's profile are retweetable; I disable the Direct Message's retweet button and adding an apposite warning title, as if that weren't the case the originally direct message would then become visible to the public/the followers of the reblogging user, and not only the original recipients. The endpoints I've implemented for the retweeting and unretweeting of status are, respectably, **/api/v1/statuses/${id}/reblog** and **/api/v1/statuses/${id}/unreblog**.

Reply to and Show full status Conversation

These endpoints allow users to reply to a status and see the entire conversation (aka all the replies organized in timely ascending order). Replies are both viewable in order and newly writable and submittable by clicking on the In conversation button, where both the text box and the previous posts are stored. Every submitted reply appears on both the Public Timeline and the Home Timeline of the logged in user's and the followers' as singular statuses with the red caption "Reply to <name of user>"; in order to both add a comment or see the previous statuses which make the conversation, the user can click on the In conversation button for all their needs. Replies to Direct Messages will have the same scope of visibility (direct) as the original poster's and will therefore only be viewable and interacted with by the originally tagged-in recipients. All the statuses pertaining to the conversation – which are visible in the In Conversation closable pop-up timeline – can be liked, retweeted and even directly replied upon to both fuel the original conversation or start new ones. The employed endpoints to both show the previous posts in the conversation and submit a new reply are: **/api/v1/statuses/${userId}/context** (for the In conversation, parent-child status-specific timeline) and **POST /api/v1/statuses** with the commentData variable set. Their implementation can be found in the /src/components/scream/ScreamDialog.js file.

ProfileTile action endpoints

Follow/Unfollow and Following/Followers count & user list

The follow/unfollow endpoints allow the logged in user to subscribe and

unsubscribe to a given user's timeline, respectively. When following a user, all of the posts they have personally made, retweeted or replied to will be visible on the logged in user's Home Timeline and made possible to interact with (if the visibility scope has been set to public upon creation). All information regarding the number of accounts that are following/being followed by a given user and the clickable list of said following/follower accounts is visible for all users (both logged in and external); however, the double-edged, double-titled and double-iconed burgundy button is only visible on the external user's ProfileTile. The clickable list of following/followers user, presented in ascendant time-based order, is hidden by default and only visible – between the Statistics and ProfileTile - when clicking on the respective following/followers counter icon.
The employed endpoints for the priorly explained functionalities are:
/api/v1/accounts/${this.props.profileId}/follow,
/api/v1/accounts/${this.props.profileId}/unfollow,
/api/v1/accounts/${this.props.profileId}/following,
/api/v1/accounts/${this.props.profileId}/followers.

Block/Unblock
These endpoints – accessed by either single-clicking or double-clicking the burgundy "Block"/"Unblock" button in a not Home/Public/Favorite/Direct Message ProfileTile, allow for a user to completely block the recipient. This means that both the blocked and the logged in users won't be able to neither mention nor see (and therefore neither interact by liking, retweeting or replying) any of the other user's statuses. Both the icon and the title of the button change when the corresponding request has been sent. Their corresponding endpoints are
/api/v1/accounts/${this.props.profileId}/block and
/api/v1/accounts/${this.props.profileId}/unblock.

Mute/Unmute
These endpoints, that are also visible in a double-edged button while not in a Home/Public/Favorite/Direct Message ProfileTile, can be accessed by either single-clicking or double-clicking the dark grey "Mute"/"Unmute". They allow the logged in user to mute the recipient – meaning that none of their post will be visible in the Home Timeline, while still being able to follow, mention or interact with the given user and their post. This functionality has been deemed as popular in the previous pleroma.site implementation, as there are many targeted bot profiles that are led by human users. Both the icon and the title of the button change when the corresponding request has been sent. Their corresponding endpoints are
/api/v1/accounts/${this.props.profileId}/mute and
/api/v1/accounts/${this.props.profileId}/unmute.

Implementation of Emoji Reactions and Counter – out-of-the-box ready for when Pleroma API's implementation gets fixed

Due to problems related to the implementation of the Pleroma API itself, it was not possible for me to add a functioning emoji reacting system, complete with counters to track which type of emoji has been used by users to react to the post. However, I have already implemented all the endpoints needed to integrate these functionalities in this project at a further time as out-of-the-box, once the parent Pleroma API functionality itself will be fixed. In particular, I have added a minimizable (and hidden by default) Emoji Choice panel – accessible by clicking either on the post status' component (to add the desired emojis to the new status) or on every already-posted scream (as reactions with each emoji's respective counters). The site-wide emoji pack

I've featured correspond to the default one (therefore not forcing every user to download an instance-personalized one), with its employed endpoint being **/api/v1/pleroma/statuses/${this.props.scream.id}/reactions/${emojiObject.unified}**. Similarly, I have also already implemented the search page, hidden navbar search component and relative endpoint for status search site-wide: **/api/v2/search**.

Description of the done work

May 4 – June 1 | Bonding Period

- Setting up the environment: Set up of a GS-dedicated virtual machine+ Installation of docker and docker-compose + Acquired the domain susannadiv.tk and set up /bin/bootstrap_certificates + Registered for a GnuDIP dynamic DNS solution + Researched systemd Debian documentation page
- Self-study of UNIX/Bash-based tools + Finished studying all the material regarding git usage + Studied additional resources regarding file manipulation and shell scripting + Short self-training period to freshen-up previous knowledge
- Debugged initial GS-fowl setup: Configured CNAME profile record from the DNS provider + Rewrote GunDIP dynamic DNS hosting setup + Installed necessary php-packages
- Attended briefing on ActivityPub, OStatus, Core and Plugins functionalities + Finished CSS styling on the GNU social landing page + Perfected readability by tuning down rgb brighteness for a less vivacious dark background + modified font palette into lighter tones
- Continued personal GS-fowl instance debugging: Searched for port-forwarding isp-related problems and analyzed possible solutions + Setting up of AWS VPS service EC2 + Setting up of a fully functioning EC-2 instance with docker and docker-compose + Debugging of DNS problems + Installation of another CS v2 instance on said EC-2 machine + set up net-based instance and started 'get to know the federation handling' tests

Total hours during the coding period: 36.3

June 1 – June 29

June 1-28 (Week 1, 2, 3 ,4)

- Debugged ActivityPub Following and Follower endpoint handling for empty list + Set up remote debugging environment
- Located and half-way through fixing Following and Follower endpoint "empty list for a specific actor does not return correct json report" bug for the ActivityPub plugin; said bug has been observed and reproduced in a GS v2 Postgres db-based instance
- Studied and reviewed previous knowledge on ActivityPub Protocol + Familiarized with the entirety of the AP plugin implementation + Brief testing of actor's profile functionality
- Fixed DB syntax error and gathered information on image uploading +

codebase image-handling familiarization + MR writing for previous debug session

- Fixed the Postgres incompatible query in both the AP and Autocomplete plugins and the clean_profile script + Brief testing of image uploading functionality + Fixed db setup
- Tested and attempted at physically optimizing tuned left-deep pipeline of notice-getting timeline query + Explained and learned about DBMS' abstract structures and postgres-specific physical time-sensitive performance optimization through usage of scalar subqueries as opposed to insiemistic operators + Tested thorughly image (and specifically avatar) upload handling in AP
- First implementation of postman's DELETE http request function for cache refresh in AP + Testing of avatar fetching in AP + Testing of OStatus' queue handling for remote likes and writing of plan of action for AP's queue handler's implementation

Total hours these weeks: 136.81

July 3 - July 27

July 3 – 18 (Week 1, 2)

- Fixed commits + indentation
- Re-read React rules, best practices and node-modules + finished CSS styling
- Added Home page + Navbar, Profile component + adjusted CSS to SCSS standard
- Added Notifications, Navbar, ProfileTile components + redux routing
- Added ProfileTile component + Network and Public pages + upgraded redux
- Upgraded user page routing & user specific ProfileTile component
- Acquainted with Bootstrap framework + studied routing for Status/post-specific model

Total hours this week: 44.5

July 19 – 27 (Week 3)

- Added & tested Like functionality + Notification Panel
- Added Login scroll down menu panel
- Notification panel touchup + designed scroll bar
- Added functioning base for full settings page
- Modified reply/comment structure
- Added spinner component

- Added input form handling

Total hours this week: 45.9

July 19 – 27 (Week 4)

- Added retractable search bar + button redesign + attachment handling + fixed routing + new post maker + added sub-notification panel
- Finished post styling & added functioning comments & added like counter & added button shading as per design & added functioning user page & re-routed login/logout pages & changed status card & added emoji button & added editProfile component
- Finished settings page + added avatar uploading + redesigned authentication system + additional routing for logged in/out functionalities
- Fixed login routing + added edit avatar component + settings routing in navbar + redux for handles
- Added placeholder backend for demonstrating functionalities + fixed settings menu
- Finishing profileTile fixes for logged in users + fixed routing + fixed public folder's contents and primary color for components
- Finishing component tweaking - deleted personal-use comments
- Fixed indentation + final testing

Total hours this week: 61

July 31 - August 24

July 31 – August 7 (Week 1)

- Studied Pleroma API + Mastodon endpoints in relation to the implemented frontend + started FavouriteList retrieval call
- Replaced Public page's API call
- Workaround on Pleroma login documentation issue + started implementation of API calls
- Finished login & signup API calls
- Studied authentication token (placement + state management) + getCaptcha implementation for registering
- Registration Code Setup & working with OAuth tokens + Added API call for home timeline data calling + Added Login, Signup, tokens & captcha reload code

Total hours this week: 36.65

August 8 – 15 (Week 2)

- Finished status object for timelines
- Post display on all timelines
- Added html parser for tag in text posts removal + adding make post component endpoint
- Post-making component with media upload + retweet/unretweet endpoint
- Showing profile data on Navbar and logout feature working + Implementing retweets mechanism
- Getting statuses for the selected users on timeline + Deleting a post
- Reacting with emoji + emoji counter endpoint implementation + beginning of reply with comments

Total hours this week: 69

August 16 – 23 (week 3)

- Finished working on In-conversation dialog box for replies history + creating comments
- Show notifications + adding mark all notifications as read
- Added ProfileTile followers/following design + fetch user's favorite timeline
- ProfileTile endpoint implementation + followers/following counter
- ProfileTile + follow + avatar implementation
- Working on follow/unfollow and followers/followings list + Showing avatar in statuses & bio in profile tile
- Finishing EditProfile + Uploading media in post and show media in statuses + Showing followings/followers list in profile

Total hours this week: 73.5

August 24 – 31 (Week 4)

- Block/unblock users + Mute/Unmute users + Favorite timeline + Direct Messages timeline + all necessary routing and buttons
- Fix onClick function for followers/following tile + refine routing for user's page
- Fixing profile routes on multiple places and hiding profile tile when logout + assuring profiles links working from anywhere including followers/followings
- Mark all notifications as done + Showing post content snippet in notification bar
- Sending direct message (post) to user + not allowing retweet for direct messages

- Final In conversation + overall design touchups
- Tech-report write up

Total hours this week: 62.5

Conclusion

During the GSoC duration I have managed to fully complete this Pleroma API based GNU social WebApp frontend and client, delivering additional endpoints implementations to be used out-of-the-box once the Pleroma API-based backend it was made to rely on will return to being completely operative. My project can now be easily cloned and installed on any machine that runs npm/yarn. Thanks to the external guidance of attentive and passionate mentors and my own self-resilience, I was able get to know both GNU social's inner workings and Pleroma's API. My Pleroma FE-like design's improved accessibility will now hopefully help spreading the word of Free Software by making it more attractive to those familiar to traditional mainstream social network platforms - expectantly helping to build a good first impression for potential newcomers to the federated network.

Already implemented, for posterity: all endpoints of both Emoji Reactions and their relative counters have been implemented in the Status component, for when Pleroma API's emoji-reacting functionality will again be declared operative. Same can be told for the direct message timeline, status search and reply counter components.

Appendix - Biography

I'm currently a second-year BSc Computer Science 4.0 GPA student at the University of Genova (UNIGE), Italy. I am also a member of the IANUA-ISSUGE Excellence Academic Program (admission requires a perfect GPA score). In addition to the above, I also have 4 autonomous JavaScript, Bootstrap, HTML5, CSS3, Node.js, Vue.Js, React, GraphQL, Express.Js and Angular frontend projects on my GitHub account – one of which secured my team first place in the Innovation 4.0 SMAU Award for a responsive React Native & JS mobile app for flood alert display.

The inspiration to code for the Federated Social Network came from the words of Richard Stallman himself, whom I had the honor of meeting during a conference on the importance of Free Software at my University – the idea of offering a privacy-focused alternative to mainstream social media platforms immediately resonated with me. I started to get involved in the movement first through contact with the community and then through taking interest in the software that powered such federation.