FeatureList-GSoC2024 - superseded by GSoc-deliverables document which builds on this

1. **Major feature of interest:** We have implemented communities where in any user can submit an article and get it reviewed and published.Each article has new comment section under it(which can be seen publicly). On a different note, we want to create private communities and people can add any article they want into their community and discuss it (but the discussions done on this private communities must not be visible to public and should be private only to particular community).

   This is the primary major feature we are looking to add right now. To give more detail, communities on the site at present are all public. The possibility of private communities means that the site can be used as a journal club or discussion group or reading group, without any risk of comments being read by non-members. This is a high-priority major update, since the site can be used in this form in order to iron out bugs before its use as an open portal.

   Anybody can set up a community and then invite people. For now, you can have all communities be hidden, unless you get an invite… but one could also have 2 versions - one where you can find community info and request to join even without an invite, and another where community is hidden completly. All this can be setup in community options by community owner/moderator. Similarly, the names/IDs of the articles being discussed in a community can either be public or private – it is fine to keep it private for now, if simpler.

2. **Major feature of interest:** Add/improve the ability to pull up articles by DOI and/or arXiv id… the goal here is that in add an article, right now, Pubmed title is the only option. Pubmed ID should be possible, and a nice display of the pubmed article (using the pubmed api) should follow. Similarly, the same can be done for arXiv papers (using the arXiv webaddress or id as input, and the api to pull the data), and for DOIs as well.

   Zotero's open-source code-base may help in this regard. In general, in future, we are targeting a Zotero plugin to the site as well, so that comments can be made from Zotero. We respect, like and look up to Zotero a lot. We don't want to recreate Zotero here, but aspects of how it pulls data automatically would be very useful !

3. **Major:** Create a unit-testing framework for the site, as it grows to have more than 1 contributor. This is becoming very important, because changes are likely to break the site.
4. **Major front end -** articles: Both a facebook feed-style interface, and a gmail style interface (with text below articles list on browser) are good designs for the “older” generation, and others that are more familiar to the current generation could work too. The present interface (for the list of articles one gets when you click on articles, for for that matter communities etc) is not workable when there are a large number of articles one is following.

   (Note that for the front-end, the goal has always been that anybody can make their own front-end to our site, we provide the API and back-end. So you are welcome to implement other front-ends using other frameworks. These can be implemented as mobile and desktop apps, and also as web-served frontends ! We are not trying to compete (or restrict) in terms of front-end designs, of which there can be many.)

5. **Frontend - mobile app.** Check out if something like Ionic Capacitor can be used to convert the React-based webpage into a mobile app. If not, is there another solution ? A mobile-app based front-end would be extremely useful. It would interact with the API provided by the back-end. **Partially done. Link to be added to webpage.**
6. **Frontend sidenav – minor (done):** The frontend design has a sidenav which is clickable, We want a fixed side navigation similar to instagram and must be responsive on mobile devices.
7. **Frontend – article page:** he article page below has comments written the design is not upto the mark, If you have any better designs.Try to enhance them.
8. **Asynchronous tasks (e.g. emails, notifications):** Reduce burden on server using an asynchronous worker. *Use Celery ?*
9. **Article chat page:** Each article has a unique group chat page where any user can share their views. Create a Group chat with join and leave feature from that group. And user must be only notified of new messages if they have joined the group.

   Think of this as a mix of Reddit and Wikipedia’s article chat. The discussion page of an article can/will be frequently moderated, re-organized etc, like a Wikipedia article, so that someone reading it is not lost in conversations of only transient value,but actually gets a sense of serious community discussion, almost like via a comment in a peer-reviewed journal. The chat/discussion then is like a Wikipedia chat… where people can discuss the content of their contributions via comments, perhaps raise isues, more of a free-for-all, anytihng of value here should be copied over to the main discussion page - this is the idea.

10. **Comment URLS -** Make a url which can be shared using comment ID, when accessing url it must show the comment which it is associated to.

11. **Pages for Article Reviewers, Article Moderators:** Design Pages for reviewers, moderators to make it easier for them to track on which articles they are added as moderators.
12. **Push Notifications for normal events on website:** Create Push Notification alert for events like when a user is added as reviewer,moderator,admin to particular community.

    When someone likes your comment under post or comments on your post. When a official reviewer,official moderator comments on your article

13. **Redesign Pages related to Community Admin**
14. **Documentation:** Add link to SciCommons developer page (make one with link on Scicommons front page).
15. **To be discussed:** potentially using nextjs and/or Typescript; ci/cd using GitHub actions and dockerization; using ShadCn instead of Material UI

Clarifications

1. This is a discussion of the relationship between articles and communities. Note that what i write below is conceptual. it is not related to how jyothiswaroop has implemented it.

   main forum - each article is its own object. many communities can “take up” an article, so an article has a property related to which communities are discussing/have discussed it. unless a community is private, all its discussions are available on the article page as well, but separated by community (and displayed as such if the front-end chooses to). the article can also be available via the community page, but the object is the same. so there is some overlap between communities and articles - meaning, a community will have many articles, but almost always not all, and each article may have 1 or more communities discussing it, but almost always not all communities. makes sense ?

   private communities are really islands. they can pick up articles and then their discussion of that stays private. if they want to have public visibility that they exist, and perhaps also which articles they are discussing, perhaps that option can be set in settings. but if their discussion of the article is also public, then they are now a public community (yes, there is also the difference that public community ratings transfer to main forum, but we dont want to have public communities with isolated ratings). so what distinguishes private communities is that their article discussion is not public and any ratings from there do not go to main forum.

2. You can introduce fresh concepts and not just be limited to the features listed here - we strongly encourage this. We suggest though, that before you spend significant time coding something, unless it is for a front-end, you discuss the concept with us after you have written up the concept, so that it is not time wasted if we were not going to integrate it in the first place. However, for the front-end, as you can read above, we welcome an ecosystem, and you are welcome to use our API to build an app as you wish.
3. Please keep pulling/rebasing latest changes from main/master into your dev branch to ensure that they are always up to date with the main branch. This workflow ensures that you can make sure your pull requests do not conflict and it is easy for the mentor to merge them after testing.

Other apps we like

1. Zotero. We really like and admire Zotero and would like to integrate with them via a plugin for commenting, making comments visible within Zotero etc. But at a later stage. Zotero also gives us interesting code for scraping and handling a wide variety of article sources very smooth - cutting-edge, as far as I know.
2. Zulip. Zulip works very very well as a Slack alternative, and we use it exclusively for our lab discussions. They are Python-based and use Django and Tornado. They do not use websockets (for, I think, legacy reasons), but use a different Tornado-based system that works very well. Perhaps we can learn something from their use of Tornado as well for certain aspects of our site.

General notes (this is text from a document I wrote previously)

We will build a web portal that facilitates manuscript submission as well as an automated, free, community-based, open-access, peer-review system where manuscripts, reviews and reviewers will all receive evaluations from the community itself. Our open-source portal will allow a) non-author controlled, double or single-blinded, pseudonymous pre- and post-publication peer review, b) reviewers to get credit in various ways for reviewing, and c) quality ratings for individual articles, reviews, authors and reviewers to emerge from collective community opinion. The portal will consist of both a public space, as well as communities that consist of a group of people interested in an area, with many, but not all of them approved to act as moderators and official reviewers. This approval can be based on qualifications, a StackOverflow-like reputation on the portal based on previous reviewing/moderating/comments, nomination by an existing moderators/reviewer, reputation of previously authored work etc. However, any registered participant can comment on any article or on any review/comment. Communities will perform a role that overlaps somewhat with that of a journal in the traditional sense, though their reviewing will be open and transparent, and their reviews can be commented upon by anyone, allowing for open, robust post-publication discussion tied to the article unlike the closed, gatekeeping-based traditional journal system of today. At submission, authors submit their manuscripts and some meta-data (like field of research, related papers, etc) at the portal. As soon as they submit, the manuscript will become visible. They can also provide a link to data/code submitted to

repositories like OSF, OpenNeuro, Dandi, C-BIG at McGill etc. At this point, the submitter has the option to pick a list of communities to “submit” their manuscript to. Once a moderator/reviewer of a community accepts the article, the system will randomly allocate another moderator and reviewers from that community to post reviews and ratings of the article, thus making sure that these identified reviewers are not hand-picked by the authors (though everyone with privileges can add reviews and ratings). It is possible that multiple communities may take up an article – all these reviews and comments will appear alongside the article, which will continue to be associated with its original public record. Authors can respond to feedback, post updated versions of their article, and a moderator or a team may manage the discussion, summarizing and collating threads as needed. Readers and viewers can filter feedback based on different criteria in order to only see reviews, ratings and article reputation from some selected group of interest. Importantly, though all participants have to identify themselves to the portal (at the time of registration via a verification process), all activity on the site can be done pseudonymously with the system assigning random handles for a given reviewer for each article, but keeping track of their activity and earned reputation. This way, credit (and potentially, discredit) is earned for activity on the portal, but reviewing can be done ethically, objectively and fearlessly. Finally, we note that if an article is not picked up by any community, it continues to live in the public space and can be reviewed/commented on by anyone – being endorsed by a community may only be needed for indexing in portals like Pubmed under the current rules. Our system also eliminates the current wasteful and frustrating process of sequentially preparing and submitting a manuscript to different journals/publishers until it finds a home.