Ronak Raj
codeitronak226277@gmail.com
India
Timezone: Indian Standard Time (UTC+5:30)

Implement a prototype of a Rich Text Editor

GSoC Project proposal for Kolibri

Personal Details and Contact Information

Full Name-	**Ronak Raj**
github username -	https://github.com/RONAK-AI647
Email **-**	codeitronak226277@gmail.com
University (optional)-	
Time-zone -	Indian Standard Time (UTC+5:30)

TABLE OF CONTENTS

Synopsis

Kolibri Studio is web application designed to deliver ,materials to Kolibri .It helps in organizing and publishing content channels on the format suitable for import from Kolibri, manages the creation of pathways and assessments .It uses the Django framework for backend and Vue.js for the frontend.
No Doubt, without Studio , creating , organizing and managing educational content before it gets imported to Kolibri would not be possible . Users are able to create content channels , arrange lessons , add videos , PDFs, and exercise all with the help of Studio.

Right Now , Kolibri Studio uses an old outdated Text editor which is complex and not flexible .The cons of the current Text editor are well mentioned in the proposal which introduces a new text prototype Rich Text Editor which is latest , flexible and most important Kolibri friendly which will definitely support Leaning Equality .

Special Thanks to Marcella and Jacob , who will mentor the entire journey of creating this gem for Kolibri Studio and upbringing the change for the good.

Benefits to the Community

- **Light weight ,Faster, Offline Friendly(most important)**
- **Works with the version that we are on(especially vue,Future-proof for Kolibri's Vue 3 migration)**
- **Performant on low cost android device(battery friendly)**
- **Accessible**
- **Can build Internationalization support**
- **Gonna bring something new for impaired people and screen readers**

Current Status of the Project

Currently Studio is using Toast UI Editor v2 as the foundation for both the Markdown editor and viewer Components. It uses @toast-ui/editor for editing and @toast-ui-editor.css for styling.While the editor is capable of handling both WYSIWYG and Markdown modes, but currently the studio uses WYSIWYG mode only .
Markdown [a lightweight markup language for formatting text , similar to HTML but easier] is stored on the backend, so conversions in both directions (Markdown ←→ HTML) is possible , we can notice that the actual Markdown mode in the UI is hidden from users.

The Toast OI editor uses the following core technologies:

- ***Markdown Editor.vue [for writing markdown with Text editor]***
- ***Markdown Viewer.vue[for displaying formatted markdown content]***
- ***WYSIWYG mode uses SQUIRE for content editing which is accessed via "editor.getSquire()"***
- ***Markdown Mode uses CodeMirror, but I find it's not active for users, can be accessed via "editor.getCodeMirror()"***
- ***The conversion of Markdown → HTML uses TOASTMARK and conversion of HTML→ Markdown uses TO-MARK.***

Now most features in the editor can be bought using PLUGINs (all plugins are located in the plugin directory {📁 `shared/views/MarkdownEditor/plugins/`):

- ***Formulas Plugin i.e Mathquill [for editing formula, adding new formula]***
- ***Image Upload Plugin[allows image insertion via drag and drop and uploading from a menu]***
- ***Customization plugin***

Cons of Current Editor:

- Heavier
- **Markdown Based**
- **No real time support**
- **Limited Customization**
- **Not Ideal for Kolibri**

INTRODUCING

Tip-Tap

Tip-Tap is a modern Text Editor which provides real time collaboration support , is flexible and lightweight , uses **ProseMirror** as core Technology , highly customizable and ideal for Kolibri. It uses version vue2 and is open with vue3 as well if Vue updates take place in future.

Why Tip Tap over any other Text Editor or Toast-UI Editor ?

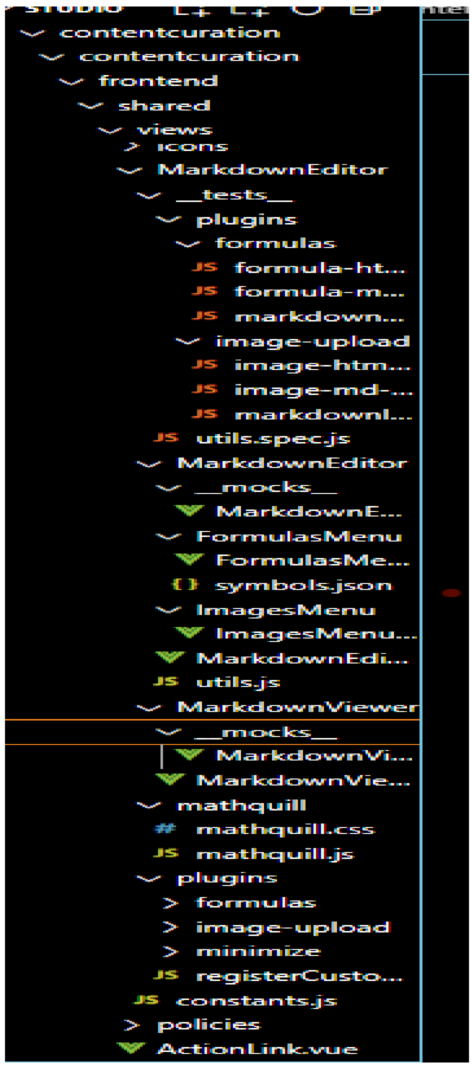
The most important criterion I am looking for in a Text Editor is *LightWeight* and *Faster* performant. As Kolibri is an International/Global app which supports global learning .It should work in regions where the internet is inaccessible , where android devices are difficult to afford. I found Tip-Tap to be most flexible , offline and Kolibri friendly which is not found in other editors like Quill,State.js,Toast UI.

Goals:

The goal of my project is to integrate Tip-Tap into Kolibri Studio.We try to understand how Tip-Tap is beneficial for Studio.Will a fully Tip-Tap integrated Text Editor work or should we integrate TipTap with Markdown for Markdown users. Hence , there are few more approaches discussed later in the the approach section(must read).

Goal 1: Replace Toast UI with Tip-Tap

- [ ] **For implementing TipTap as our primary text editor , our goal 1 will include understanding the use of the current editor in Studio repo .What kind of files , features , imports and plugins are used.For example : the image below shows all the files which includes the text editor's data.**



```
STUDIO
⌄ contentcuration
  ⌄ contentcuration
    ⌄ frontend
      ⌄ shared
        ⌄ views
          > icons
          ⌄ MarkdownEditor
            ⌄ __tests__
              ⌄ plugins
                ⌄ formulas
                  JS formula-ht...
                  JS formula-m...
                  JS markdown...
                ⌄ image-upload
                  JS image-htm...
                  JS image-md-...
                  JS markdownl...
              JS utils.spec.js
            ⌄ MarkdownEditor
              ⌄ __mocks__
                MarkdownE...
              ⌄ FormulasMenu
                FormulasMe...
                { } symbols.json
              ⌄ ImagesMenu
                ImagesMenu...
              MarkdownEdi...
              JS utils.js
            ⌄ MarkdownViewer
              ⌄ __mocks__
                MarkdownVi...
              MarkdownVie...
            ⌄ mathquill
              # mathquill.css
              JS mathquill.js
            ⌄ plugins
              > formulas
              > image-upload
              > minimize
              JS registerCusto...
            JS constants.js
          > policies
          ActionLink.vue
```

- ☐ Making sure the feature parity with the existing Toast UI Editor . For this we can initialize the Tip tap editor instance using STARTKIT for core markdown features (e.g., support for headings, bold, italics, lists, links).
- ☐ Extensions for markdown Support will be created using @tiptap/markdown. Here we will also create custom extensions for formulas and image uploads to match existing Studio plugins .
- ☐ Create a new TIpTapMarkdownEditor.vue which will replace MarkdownEditor.vue .
- ☐ Replace MarkdownViewer.vue with Tip tap read only rendering mode as well as we will try to leverage EditorContent with a non-editable Tip Tap instance to render markdown as HTML.

Goal 2:Improve Accessibility & Screen Reader Support and Extension Replacement

- ☐ Ensure Tip Tap provides better accessibility for visually impaired users,people with vision problems, using WCAG guidelines.
- ☐ Add keyboard navigation and ARIA roles to improve the experience for screen reader users.
- ☐ We will create a custom TipTap extension for formulas using MathQuill . Here taking mentors' help will be helpful whether to use the old logic(used with TOI) or any other logic can be used.
- ☐ For the image upload plugin part , we will try to implement a custom Tiptap NodeView to mount Vie image components (reusing existing image resizing and edit logic). We can also use Studio's existing file upload system to upload and persist images or even concert images to markdown or html on save .

Goal 3:Enhance Customizability & Extensibility With Bidirectional support Markdown <-> HTML conversions.

- ☐ Provide enhancement for collaborative editing which can be Done by replacing TOAST UI’s `ToastMark` and `to-mark` with Tiptap’s markdown parsing pipeline.
- ☐ We will integrate @tiptap/markdown to serialize editor content to markdown on save and on load , it should deserialize save markdown into tiptap-compatible HTML or JSON.

Goal 4 :Internationalization and Further Enhancement

- ☐ To internationalize Tip Tap in Kolibri Studio, we can integrate Vue I18n for UI translations and support RTL languages like Arabic which is written left to right.. Language-specific spell checking and dynamic language switching will improve usability.
- ☐ The backend should handle multilingual content storage effectively. As a stretch goal, features like automatic translation suggestions and voice input can enhance accessibility.

Goal 5:Create a custom UI for managing editor settings(if time period allows)

We can create a customization like toolbar customization. Now going a little deeper .The current TOAST UI editor provides very limited control over personalization or feature toggling. But with the flexibility of TipaTap and vue , we can build a dedicated settings menu , where any user can :

- Enable or disable editing features (e.g., math formulas, image uploads, tables)
  Choose a default font size or theme (e.g., light/dark mode, larger preview font)
- Toggle auto-formatting or live preview
- Set default alt-text or image sizes for uploads
  Enable accessibility enhancements (e.g., keyboard shortcuts, high contrast mode)
- For this , technically we will need to add a new vue component : EditorSettingsPanel.vue. We will mount it as a sidebar or a modal or maybe a dropdown accessible from the toolbar. At last we will integrate settings with Tiptap's extension system to dynamically change behavior.

  Example : "*if (this.setting.enableFormulas) { extensions.push(FormulaExtension);}* "

- *This makes it easier to introduce future enhancements (like plugin toggles or themes) without touching the core editor code*

Deliverables

The possible Deliverables are as follows:.

- **Deliverable 1**

The first Deliverable should include setting up TipTap in Kolibri Studio alongside the existing Toast UI editor.WE will implement a basic text editor component with core editing features like underline , bold etc.

- **Deliverable 2**

  Adding Markdown support to Tip-Tap and ensuring a smooth transition between markdown and smooth Text. Then implementation of internationalization (i18n) will be our next task, so that accessibility to multiple languages can be achieved.Using accessibility features for screen readers , those with visual problems , hearing impairments etc and screen reading compatibility will be introduced.

- **Deliverable 3**

We will try to migrate to tip tap fully from toast UI , if it turns out to be a feasible action. Next collaborative editing , data persistence, backend integration can be achieved in the final deliberation and if time persists we will work on Goal 5 as well.

Expected Results

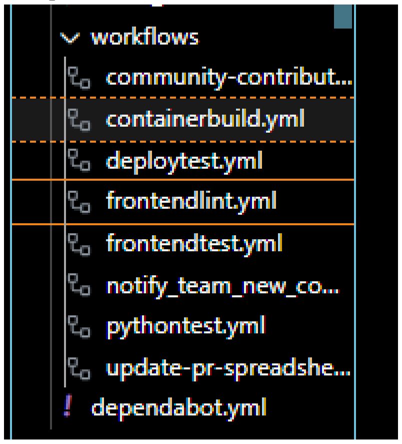
By the end of the project, Kolibri Studio will have a fully integrated Tiptap-based editor, offering a

- Modern,
- extensible,
- accessible

alternative to the existing Toast UI Editor. This will provide better Markdown support, improved formatting options, and seamless user experience. Additionally, the new editor will enhance accessibility for screen readers, support internationalization, and lay the foundation for future collaborative editing features.

Will integrating the new Tiptap-based Studio Editor require any new workflow check?

Currently we have 6-7 workflow checks in Studio. My research says possibly , but not necessarily.It depends on how we implement and test the new editor .See below:

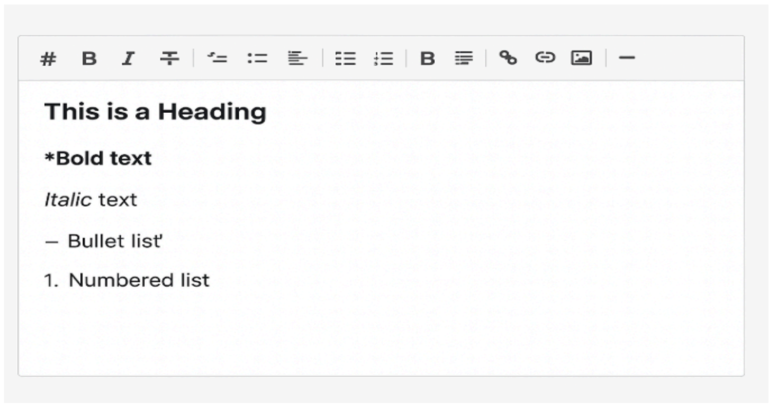


```
workflows
  community-contribut...
  containerbuild.yml
  deploytest.yml
  frontendlint.yml
  frontendtest.yml
  notify_team_new_co...
  pythontest.yml
  update-pr-spreadshe...
dependabot.yml
```

OR possibly we can add a new test folder (/test/editor-e2e) with new types of tests and we can name the check to be like “studioeditor-e2e.yml”. Else we can use Percy

Visual testing in Studio as well which already exists in KDS.This requires mentors approval .

1) Text Editing with Toast UI [old editor]


This is a Heading

*Bold text

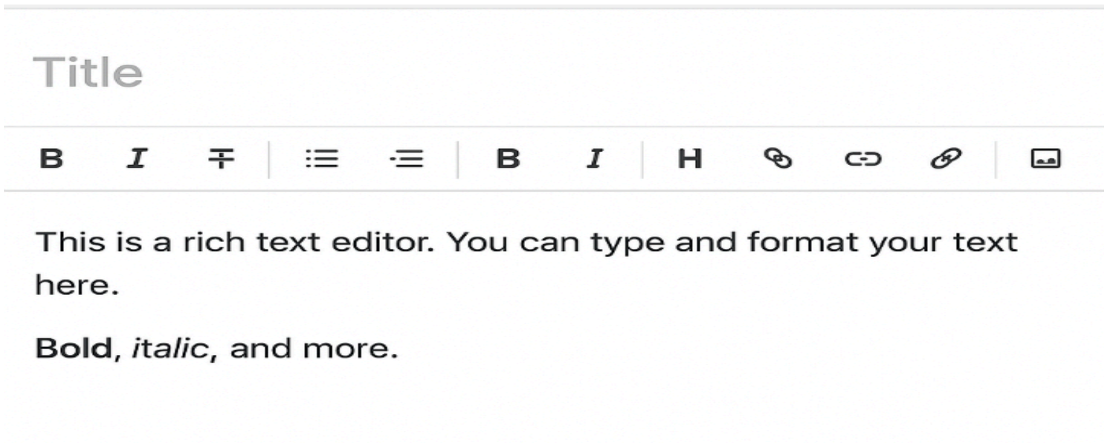
Italic text

– Bullet list'

1. Numbered list

This image represents Toast UI Editor – it has a Markdown-based interface with a side-by-side preview.

2) Text Editing with TipTap [Modern text Editor]


Title

This is a rich text editor. You can type and format your text here.

Bold, *italic*, and more.

This image represents Tip Tap – it features a rich text editing interface with a modern toolbar.

Approach:

There are 2 possible approaches I have researched:

Standalone Tip Tap

- **Pros**: It is modern , fully featured , supports rich text formatting and collaborative editing.
- **Cons**: Markdown users might need to adjust .

Hybrid(TipTap + Markdown + Toast UI Editor)

- **Pros**: Supports both ,Markdown and TipTap .
- **Cons** : Requires conversions and is much more complex

My Opinion:

The fully Tip Tap based text editor is awesome and we can see rich text features .The Hybrid approach is good , but for markdown users which supports markdown Based editing. We will try something which will successfully transition from the current Toast UI to TipTap while ensuring continuity in Kolibri Stiudio's editor functionality. I plan to follow a modular replacement strategy which allows us to maintain editor stability.

PHASE 1: COMMUNITY BONDING AND UNDERSTANDING ALL THE USAGE POINTS OF THE CURRENT TOAST UI IN STUDIO REPO:

- **I see the "studio\docs\markdown_editor_viewer.md"provides great documentation of the editor(toast, markdown, its parsing and conversion, about plugins, and other components like viewer etc . used in Studio all round) . We will keep editing this file as our goals are accomplished.**

- **Audit current implementation. The major files and directories related to the current editor are mostly located here:** "`contentcuration/frontend/shared/views/MarkdownEditor/`:"

Editor Component Structure

1)MarkdownEditor.vue: This file includes all the imports,and the core Vue component mounting the TOAST UI editor instance. Here is a summary of what will change in MarkdownEditor.vue after Integrating TipTap.

- **WE WILL CHANGE THE IMPORTS: We will import TipTap's Editor class and initialize a new instance, probably using `useEditor()` if we are framing it as a Vue component.eg: *this.editor = new Editor({....})***

- **MAKE THE TEMPLATE CHANGE: EditorContent is the Vue2 compatible rendering component from @tiptap/vue-2., so we will change the current "<div ref="editor" class="editor"></div>" with " <EditorContent :editor="editor" />".**

- **REMOVE TOAST UI ASSETS: All Toast Ui css/ js imports and assets will be removed and we will import the required styles from Tioptap and prosemirror if needed.**

- **PLUGIN INTEGRATION FORMULAS AND IMAGE UPLOAD: We will change the plugins to Extensions in Tiptap: We will change these current plugins as TipTap extensions .**

- **TWO WAY BINDING FOR MARKDOWN PROP : Tip Tap provides .getHTML() or .getJSON() method from the editor instance to emit updates manually.**

- **FORMULA MENU AND IMAGE MENU OVERLAY POSITIONING: These components will stay but the selection coordinates, positioning logic will change . We will tiptap's `editor.view.coordsAtPos(pos)` or `editor.view.domAtPos()` to calculate overlay positions.**

- **UTILS USAGE:Functions like `generateCustomConverter`, `getExtensionMenuPosition`, `clearNodeFormat` will be rewritten to work with Tiptap's state and view APIs and instead of parsing markdown directly , we will be using ProseMirror's document node traversal.**

2)MarkdownViewer.vue - It helps in viewing the markdown content and is used in read only views like preview panela and detailed pages.

3) Utils.js - Here we will make updates according to the way we are going to handle TipTap with markdown.
It contains helper methods like formatting , parsing and processing markdown or html .

4) __test__ & __mocks__: These folders as we know help for testing . especially when we are working with a VUE + JEST setup like in Studio . The __tests__ files in Studio are the test cases written using Vue Test Utils to ensure that components , plugins or utilities work as expected. When we integrate TipTap ,we will update both these files , we will need to rewrite __tests__ so it reflects the new implementation.
If we update any component , we will need to mention[it in the __mocks__ files to match the new structure and new names.Example: If our new `MarkdownViewer.vue` (Tip Tap version) now supports rendering math formulas using Tip Tap extensions, but you still only want a mock for `markdown`, just ensure the mock has the same `props.`

5)

- **plugins/formulas:**

These subdirectories include files like formula-html.js, formula-md.js, and markdown-it-plugin.js which are rendered in HTML and markdown .WE WILL REPLACE IT WITH CUSTOM TIPTAP INLINE NODE+ MATHQUILL NODEVIEW.

- **plugins/Image-upload**

These subdirectories include image-html.js, image-md.js, and registerCustomHTMLRenderer.js. WE WILL REPLACE THEM WITH A TIP TAP EXTENSION WHILE PRESERVING STUDIO'S UPLOAD WORKFLOW.

6)

- **FormulasMenu.vue & ImagesMenu.vue:**

These are likely to be reused in Tiptap NodeVIews for rich customisation.

- **symbols.json**

It includes JSON file listing LateX symbols used in the formula editor.WE will preserve and use in the TIPtap formula picker .

- **Mathquill/**

It contains two files mathquill.js and mathquill .css . Mathquill is a great and consistent option , we will use it for formula extension for sure.

PHASE 2: FEATURE TESTING AND COMPATIBILITY

- **Write integration tests for:**
- ☐ **Editor initialization**
- ☐ **Rendering formula and images**
- ☐ **We will validate that markdown stored by tip tap is compatible with previously stored markdown content to maintain backward compatibility in this phase.**

PHASE 3: GRADUAL REPLACEMENT AND CLEANUP

- ☐ **Our last phase which will include the replacement of MarkdownEditor.vue across the app with TipTapMarldownEditor.vue. The complete removal of Toast UI and its dependencies from the project . We will try to migrate any shared logic(toolbar , modals, etc) from old editor to reusable components which means while replacing old editor with Tip tap , there will be several UI features that the old editor ,managed well like the formula insertion menu , image upload menu , toolbar interaction , utilities.**
- ☐ **So instead of rewriting them all from start , we will extract and refactor these into modular and reusable Vue components .Right now, much of the logic is tightly coupled to TOAST UI's APIs and markdown structure.**
- ☐ **To build a clean and modern architecture, we want Tiptap to handle content, but move the rest of the functionality (menus, image uploads, etc.) into Vue components or utilities that are reusable and testable.**

Questions which may arise:

- **Do Tip tap support the version of vue or other dependencies?**

  TIp Tap (the new text editor) is built to work natively with vue.js only (including Vue 2 and vue 3). It will work with the current vue version which Kolibri Studio uses version 2.6.12. In future if Vue is updated to Vue 2.7.x or Vue 3. , Tip Tap will be compatible to work with both.The other dependencies like ….

  - [ ] ***Yarn 1.22.22***
  - [ ] ***Node 16.20.2***
  - [ ] ***Vue 2.6.12***
  - [ ] ***Npm >=8***
  - [ ] ***etc aso support***

- **Can we drag and drop image uploads or using Advanced Embeds like Youtube , Twitter possible?**

  Yes , such features were not available on the Toast UI editor which studio uses Currently .The Tip Tap text editor will provide us with all these features.

- **Kolibri keeps in mind that people with disabilities also could study using Kolibri .What Tip Tap brings for physically challenged people?**

  Kolibri prioritizes accessibility, ensuring that visually impaired users and people with disabilities can use the platform effectively. Tiptap does not provide built-in accessibility features , but we can add it with ARIA attribute , keyboard navigation and screen reader support..

  - ***Screen Readers:***
    Tip tap does not have any built-in screen reader optimization but we can add them manually, We can add ARIA role attribute which will make text editing more accessible.

Example: ***role="textbox" aria-multiline="true***

- ***Audio Feedback for formatting:***

  We can add a feature where screen readers announce text formatting changes (e.g., "Bold text applied").Use `aria-live="polite"` to Users ensure hear updates . This is beneficial for those who are having Impairments.

- ***Keyboard Navigation and focus Management***

  We can easily implement Keyboard navigation (ctrl+B , ctrl+I) etc for text formatting.

Timeline:

Period	Task
After proposal submission **April 8 - June 1**	**Community Bonding Period**
Week 1 **Week 2** **Week 3** **Week 4**	- **Implementing Goal 1** - **Working on Phase 1 of approach** - **Implementing Goal 2** - **Implementing Goal 3 and working on Phase 2 of approach** - **Evaluation 1**
Week 4 **Week5** **Week 6**	**.- Implementing Goal 3** **- Working on approach phase 3 of approach** **- Evaluation 2**

Week 7 Week 8 Week 9	- Working on Goal 4 - analyzing phase 4 and working on it -working on phase 5 - Implementing Goal 5 if possible

Additional Information

About Me:

Hey ! This is Ronak Raj , a High SchoolGraduate (class 12)Completed in 2024 under [CBSE], India. I am a frontend web developer and a software enthusiast and a self learner .I love spending my time on coding and learning new stuff .I have experience with HTML, CSS, Git/Github, Javascript(React, Vue), Python (django) and a little bit of C++.
Apart from this, I am a professional speaker , motivator , calligraphist and dancer .

Skills(1-5)

• Html	-	4.5
• Css	-	4.5
• javascript(react, vue)	-	4
• Python(django)	-	4
• Git/Github	-	4
• C++	-	3

Previous Contributions in Learning Equality:

Issue	Link to Issue /PR	Additional notes
• Updating KDS icon with the brand new logo	https://github.com/learningequality/kolibri-design-system/pull/809	Merged
• Ensure Visibility of Focus Ring in Firefox	https://github.com/learningequality/kolibri-design-system/pull/827	Merged
• Update docs of z-index and drop-shadow for better clarity	https://github.com/learningequality/kolibri-design-system/pull/819	Merged
• Adding Title attribute in KBreadcrumbs for Truncated Texts	https://github.com/learningequality/kolibri-design-system/pull/872	Merged
• Adding Husky for yarn lint-fix	https://github.com/learningequality/kolibri-design-system/pull/880	In progress
• Updating changelog.yml/workflows to make it more robust.	https://github.com/learningequality/kolibri-design-system/pull/885	In progress
• Updating browsing History Integration with the navigation Table of content (Toc) filter.	https://github.com/learningequality/kolibri-design-system/pull/847	Merged
• Visual Test for KBreadcrumbs	https://github.com/learningequality/kolibri-design-system/pull/936	In progress
• Using the new ariaLabelledBy prop in KSwitch	https://github.com/learningequality/kolibri/pull/12918–	In progress

Why learning Equality:

A year ago , I got my first laptop. I was confused what to do with a desktop when every use of mine is fulfilled with my smartphone . For 2 months ,I left it unused . One very bright day , I thought of doing something , I thought of coding , and heard about open source . Everyone told , I am wasting my time , it's too early to code and I shou;ld wait to enter a University . But I kept going , I faced many hurdles , but as someone just turned 18 and enthusiastic to learn more and more brought me to Learning Equality , the idea of offline learning and hel[ping people worldwide to reach education is an insane idea which striked my head . Since then I decided , I will keep contributing to LE till I am an open source contributor . I learned everything I have today due to LE and the maintainers . I am Thankful to them.

Plans after GSOc

GSoC or not, I would love to continue contributing to Learning Equality and give my little contribution to anything that I can do to help bridge the digital gap that exists around us.

Other Commitments:

I don’t have any other plan or commitment except GSoC this summer . I am completely devoted to contributions for the right cause.

Thank You!!!