**Chapter X Approach 2 – Integrating other third party open source text editors into OpenEdx platform** **(needs appropriate title)**

**X.1 Integration Of The other third party Editors**

Just like CodeMirror and TinyMCE there are other third party open source text editors which meet all the prerequisites of our enhancement critera. Some of them are Notepad++, Quill, Brackets, ContentTools, Grapejs,etc. As an alternative to our first approach we thought of another possibility of reaching our objective. Just like how the OpenEdx has merged two text editors for their HTML editor, similarly even we can try to integrate WYSWYG editor using the same methodology. In this approach we develop an entire new editor itself. Moreover since all these editors are open source their github repositories is easily accessible. So we also thought of trying to import the logic they used in their source code to add features such as code indentation ,code folding etc. We put this approach as our backup for our approach1 because, approach2 is a bit hacky and complicated to implement compared to approach1

**X.2 Other similar editors (title required)**

Before integrating here are some of the questions that we asked ourselves:

* What kind of editor do we want...??
* What features does that editor should have..??
* How does these editors help with our motive..??

So here are few points that helped us clear our doubts:

* We need an editor that makes our work easy, simple, and has got much better working properties as compared to the editor that is currently being used in the OpenEdx platform.
* The editor should have inbuilt features(code folding, pretty indentation,languages,etc) that could just wipe out the issues that are being faced by the openEdx html editor.
* Since our motive is to add features such as indendation , cold folding, internal css,etc, so considering an editor that has these features inbuilt, and integrating it by making some required changes might help us in achieving our motive, and would even resolve in eliminating the issues faced by the current HTML editor that is being used by the OpenEdx platform.

After a bit of research work what we found out is some pretty good and well built editors that support html language. All these editors are open source and all their source code is available in their respective git repositories. The list of editors that we had researched on are as follows:

* Notepad++
* Brackets
* ContentTools

The detailed explanation about each editor follows .

**X.3 Notepad++**

**X.3.1 About**

Notepad++ is a free (as in “free speech” and also as in “free beer”) open source code editor and Notepad replacement that supports several languages. Running in the operating system its use is governed by the GPL license.

Based on the powerful editing component SCINTILLA, Notepad++ is written in c++ and uses pure WIN32 API and STL which ensures a higher execution speed and smaller program size.

As of now the current version of the Notepad++ available in the market is v7.5.6.

**X.3.2. Features**

1. Syntax Highlightning
2. Syntax Folding
3. User Desfined Syntax Highligting and Syntax Folding. (images (n++1,2,3,4))
4. PCRE (Perl Compatible Regular Expreession) Search and Replace.
5. GUI entirely customizable.
6. Auto completion: Word completion, Function completion and Function parameter hint.
7. Multi document (Tab interface).
8. Multi view
9. WYSIWYG (printing).
10. Zoom in and Zoom out.
11. Multi language environment supported.
12. Macro recording and playback.
13. Bookmark.

**X.3.3 observations made**

Here are a list of things that we have found about notepad++ :

1. Scintilla is the main component of Notepad++ which is very powerful.
2. This editor is written in c++
3. All the syntax styling part is done using the Scintilla component.
4. Scintilla edits and even is used for debugging the source code of the Notepad++
5. Some files where the required changes are to be made and importing those files so they can be used in building an editor for OpenEdx platform.

**X.3.4 Links**

Here are some links that would be helpful in knowing where the required code snippet regarding styling, language, etc, are present related to notepad++

* *Source code:*

the source code of notepad++ is available on github repository :-

[**https://github.com/notepad-plus-plus/notepad-plus-plus**](https://github.com/notepad-plus-plus/notepad-plus-plus)

* *Indendation code:*

the code regarding where the logic of notepad++ indentation is available is below:

[**https://github.com/notepad-plus-plus/notepad-plus-plus/blob/master/scintilla/lexlib/Accessor.cxx**](https://github.com/notepad-plus-plus/notepad-plus-plus/blob/master/scintilla/lexlib/Accessor.cxx)

the code regarding where the logic of Syntax Folding is present is below:

[**https://github.com/notepad-plus-plus/notepad-plus-plus/blob/master/scintilla/lexers/LexVerilog.cxx**](https://github.com/notepad-plus-plus/notepad-plus-plus/blob/master/scintilla/lexers/LexVerilog.cxx)

the code regarding where the logic of Syntax Higlighting , Detecting Comments(both single & multi line comments) , Initialization of indentation level is present is below :

[**https://github.com/notepad-plus-plus/notepad-plus-plus/blob/master/scintilla/lexers/LexKVIrc.cxx**](https://github.com/notepad-plus-plus/notepad-plus-plus/blob/master/scintilla/lexers/LexKVIrc.cxx)

the below link provides reference to the code where the functions regarding all the above functions of indendation are included:

[**https://github.com/notepad-plus-plus/notepad-plus-plus/tree/master/scintilla/lexlib**](https://github.com/notepad-plus-plus/notepad-plus-plus/tree/master/scintilla/lexlib)

* *Lex files:*

The code for regeneration of the files of the source code based on the comments is found in the below link:

[**https://github.com/notepad-plus-plus/notepad-plus-plus/blob/master/scintilla/scripts/FileGenerator.py**](https://github.com/notepad-plus-plus/notepad-plus-plus/blob/master/scintilla/scripts/FileGenerator.py)

code relating to all the Lex filesis present in the below link:

[**https://github.com/notepad-plus-plus/notepad-plus-plus/tree/master/scintilla/scripts**](https://github.com/notepad-plus-plus/notepad-plus-plus/tree/master/scintilla/scripts)

* *Fonts and colour styling:*

Code relating the available fonts in the editor and scolour styling for each font is present in the below link:

[**https://github.com/notepad-plus-plus/notepad-plus-plus/blob/master/scintilla/src/Style.cxx**](https://github.com/notepad-plus-plus/notepad-plus-plus/blob/master/scintilla/src/Style.cxx)

* *Autoc Completion:*

code relating the Auto Completion feature is present in the below link:

[**https://github.com/notepad-plus-plus/notepad-plus-plus/blob/master/scintilla/src/AutoComplete.cxx**](https://github.com/notepad-plus-plus/notepad-plus-plus/blob/master/scintilla/src/AutoComplete.cxx)

* *Sparsing of styles:*

the code for storing the sparsed fonts and colour styling is done in the below link:

[**https://github.com/notepad-plus-plus/notepad-plus-plus/blob/master/scintilla/src/RunStyles.cxx**](https://github.com/notepad-plus-plus/notepad-plus-plus/blob/master/scintilla/src/RunStyles.cxx)

* *Line marker:*

code relating the numbering the lines of code is present in the below link:

[**https://github.com/notepad-plus-plus/notepad-plus-plus/blob/master/scintilla/src/LineMarker.cxx**](https://github.com/notepad-plus-plus/notepad-plus-plus/blob/master/scintilla/src/LineMarker.cxx)

* *Decoration:*

Code relating the Decoration style and the visual elements that are added over text is present in the below link:

[**https://github.com/notepad-plus-plus/notepad-plus-plus/blob/master/scintilla/src/Decoration.cxx**](https://github.com/notepad-plus-plus/notepad-plus-plus/blob/master/scintilla/src/Decoration.cxx)

* *Languages:*

the code regarding the identification of various languages such as Perl, Ruby, HTML, CSS , JS ,etc, is found in the below link:

[**https://github.com/notepad-plus-plus/notepad-plus-plus/tree/master/scintilla/lexers**](https://github.com/notepad-plus-plus/notepad-plus-plus/tree/master/scintilla/lexers)

**X.3.5. Idea of Implementation**

In a similar way in which the Open edX platform uses an xmodule(html\_module.py) that handles the operation of its current html editor, similarly, using the same working principle we can replace our editor and load the html editor and configure it for proper working.

**X.4. Adobe Brackets**

**X.4.1 About**

With focused visual tools and preprocessor support, Brackets is a modern text editor that makes it easy to design in the browser. It's crafted from the ground up for web designers and front-end developers

Brackets is an open source editor written in HTML, CSS, and JavaScript with a primary focus on web development. It was created by Adobe Systems, licensed under the MIT License, and is currently maintained on GitHub by Adobe and other open-sourced developers. Brackets is available for cross-platform download on Mac, Windows, and is compatible with most linux distros. The main purpose of brackets is its live html, css and js editing functionality.

**X.4.2. Features**

1. Quick Edit
2. Quick Docs
3. Live element debugging
4. Live preview
5. Inline Editors
6. Split View
7. Thesus Integration
8. LESS support
9. W3C Validation
10. Drag and Drop
11. Auto Prefixer
12. Git Integration for Brackets
13. JS Lint

**X.4.3. Observations made**

Here are a list of things that we have found out about Brackets:

1. Brackets is written in JavaScript
2. Its an automated WYSIWYG editor
3. Mainly designed for web developers and front end developers
4. Applies quick edit for HTML elements which helps in displaying corresponding CSS properties for that particular element.
5. When a color is typed it shows an inline color picker for our better convienience in searching.
6. While writing the code it enables live preview , that works only for Google Chrome, and if there is any html syntaxtical error then it forbiddens opening of the live prview.
7. Using this Thesus integration feature that enables inspecting an element in the real time and debug any extension in brackets.
8. This uses CodeMirror text raw html editor for Code Formatting.
9. Brackets has got more than 20 Dependencies .
10. It consists of a bracket (which is a root module) that pulls in other modules as dependencies.
11. Some files where the required changes are to be made and importing those files so they can be used in building an editor for OpenEdx platform

**X.4.4. Links**

Here are some links that would be helpful in knowing where the required code snippet regarding styling, languages, code formatting,etc, are present in Brackets:

* *Source Code:*

the source code of brackets is available on github repository :-

[***https://github.com/adobe/brackets***](https://github.com/adobe/brackets)

* *Code Formatting:*

The code regarding code Formatting is found in the below link:

[***https://github.com/adobe/brackets/blob/master/src/styles/brackets\_codemirror\_override.less***](https://github.com/adobe/brackets/blob/master/src/styles/brackets_codemirror_override.less)

the code regarding all the styles used in brackets are found in the below link:

[***https://github.com/adobe/brackets/tree/master/src/styles***](https://github.com/adobe/brackets/tree/master/src/styles)

code related to handling the color matching functionality is found in the below link:

[***https://github.com/adobe/brackets/blob/master/src/utils/ColorUtils.js***](https://github.com/adobe/brackets/blob/master/src/utils/ColorUtils.js)

* *JS Code Hints:*

The code related to JavaScript code hints while writing the code in this editor is found in the below link:

[***https://github.com/adobe/brackets/tree/master/src/JSUtils***](https://github.com/adobe/brackets/tree/master/src/JSUtils)

* *Live Development:*

all the codes related to the Live Development Feature is found in the below link:

[***https://github.com/adobe/brackets/tree/master/src/LiveDevelopment***](https://github.com/adobe/brackets/tree/master/src/LiveDevelopment)

*the below link refers to the code that loads and reloads the inline stylesheets using CSS:*

[***https://github.com/adobe/brackets/blob/master/src/LiveDevelopment/Agents/CSSAgent.js***](https://github.com/adobe/brackets/blob/master/src/LiveDevelopment/Agents/CSSAgent.js)

the code related to the Interaction of the debugger with the editor interface is found in the below link:

[***https://github.com/adobe/brackets/blob/master/src/LiveDevelopment/Agents/ScriptAgent.js***](https://github.com/adobe/brackets/blob/master/src/LiveDevelopment/Agents/ScriptAgent.js)

the code related to launching the live preview is found in the below 2 links:

[***https://github.com/adobe/brackets/blob/master/src/LiveDevelopment/LiveDevMultiBrowser.js***](https://github.com/adobe/brackets/blob/master/src/LiveDevelopment/LiveDevMultiBrowser.js)

[***https://github.com/adobe/brackets/blob/master/src/LiveDevelopment/LiveDevelopment.js***](https://github.com/adobe/brackets/blob/master/src/LiveDevelopment/LiveDevelopment.js)

the code related to the connection management of live Preview to Chrome is found in the below link:

[***https://github.com/adobe/brackets/blob/master/src/LiveDevelopment/Inspector/Inspector.js***](https://github.com/adobe/brackets/blob/master/src/LiveDevelopment/Inspector/Inspector.js)

the code related to the live Development integration into brackets is found in the below link:

[***https://github.com/adobe/brackets/blob/master/src/LiveDevelopment/main.js***](https://github.com/adobe/brackets/blob/master/src/LiveDevelopment/main.js)

* *Editor:*

all the links related to the features of editors are included here.

[***https://github.com/adobe/brackets/blob/master/src/editor/***](https://github.com/adobe/brackets/blob/master/src/editor/CodeHintList.js)

*Css inline editor:*

The code related to the CSS inline editor is found in the below link:

[***https://github.com/adobe/brackets/blob/master/src/LiveDevelopment/main.js***](https://github.com/adobe/brackets/blob/master/src/LiveDevelopment/main.js)

*Code hint :*

The codes related to the Quick Edit feature of the Brackets is found in the below link:

[***https://github.com/adobe/brackets/blob/master/src/editor/CodeHintList.js***](https://github.com/adobe/brackets/blob/master/src/editor/CodeHintList.js)

*image viewer:*

the code realted to the uploaded image link given while writing the code is found in the below link:

[***https://github.com/adobe/brackets/blob/master/src/editor/ImageViewer.js***](https://github.com/adobe/brackets/blob/master/src/editor/ImageViewer.js)

* *Html contents:*

All the code related to the html contents is present down in the below link:

[***https://github.com/adobe/brackets/tree/master/src/htmlContent***](https://github.com/adobe/brackets/tree/master/src/htmlContent)

* *Animation:*

The code related to the Utilities dealing with animation in the user interface is found here in the below link:

[***https://github.com/adobe/brackets/blob/master/src/utils/AnimationUtils.js***](https://github.com/adobe/brackets/blob/master/src/utils/AnimationUtils.js)

* *Drag and Drop:*

the code related to the Drag and Drop feature is found here in the below link:

[***https://github.com/adobe/brackets/blob/master/src/utils/DragAndDrop.js***](https://github.com/adobe/brackets/blob/master/src/utils/DragAndDrop.js)

* *Highlighting codes:*

the code related to the Code Highlighting feature is found here in the below link:

[***https://github.com/adobe/brackets/blob/master/src/LiveDevelopment/Agents/HighlightAgent.js***](https://github.com/adobe/brackets/blob/master/src/LiveDevelopment/Agents/HighlightAgent.js)

* *Languages supported:*

the code regarding the identification of various languages such as Perl, Ruby, HTML, CSS , JS ,etc, is found in the below link:

[***https://github.com/adobe/brackets/tree/master/src/language***](https://github.com/adobe/brackets/tree/master/src/language)

using the above reference we can include only those languages that we require and eliminate the others.

**X.4.5. idea of Implementation**

In a similar way in which the Open edX platform uses an xmodule(html\_module.py) that handles the operation of its current html editor, similarly, using the same working principle we can replace our editor and load the html editor and configure it for proper working.

**X.5. ContentTools**

**X.5.1. About**

**ContentTools** is a JavaScript/CoffeeScript library aimed at building WYSIWYG editors for HTML content. ContentTools aims to provide both a fully-functional editor that can be used out of box and a toolkit of classes, a set of tools for performing common editing tasks, and a history stack for managing undo/redo. Whilst the components provided by the toolkit work well together, they can also be used or replaced as required.

**X.5.2. Features**

1. Easily integrated with any HTML document.
2. Drag and Drop directly within the page.
3. Floating context-sensitive toolbar.
4. Compatible with all major web browsers and operating systems.
5. The javaScript library can transform any HTML page into an WYSIWYG editor.
6. Countless possibilities for building wondrous interactive apps and services.
7. Media Resizing.

**X.5.3. Observations made**

1. Allows text content, images, embedded videos, tables and other page content to be edited, resized, or moved using the Drag and Drop property within the page itself.
2. Requires bower or npm for installation.
3. For building library’s and project, requires grunt node modules and SASS.
4. Uses an HTML string for formatting rather than using an HTML parser written in JavaScript.
5. Its library uses a minimal finite state machine(FSM) for JavaScript.
6. It consists of an JS library that provides cross-browser support for content selection.
7. It has a javascript library that provides a set of classes for building content editable HTML elements.
8. The ContentTools editor has already been implemented into the content management systems of a number of websites.
9. This editor is a recent editor which has got wonderful, simplified and an unique UserInterface.
10. Its Framework integration includes “Django REST Framework” and “Image Uploads with Cloudinary”.
11. ContentTools is part of a collection of JavaScript libraries (ContentTools, ContentEdit, ContentSelect, FSM, HTMLParser) which were developed to aid in the creation of HTML WYSIWYG editors.
12. It has an ability to configure styles for your content.
13. Contenttools can be easily integrated into the CMS.
14. It is an opensource web-based HTML editor whose working is quite similar to CodeMirror and TinyMCE, so integrating with the OpenEdx platform would be quite simpler as all of then belong to the same working methodology.

**X.5.4. Links**

Here are some reference links that would get in much detail about the working of ContentTools and how to include it in your project, how to enable some required essential plugins for having much more extra features.

1. Setting up:

<https://developer.telerik.com/featured/a-review-of-contenttools-a-rich-content-editor/>

1. Source code:

<https://github.com/GetmeUK/ContentTools>

1. Saving Straegies:

<http://getcontenttools.com/tutorials/saving-strategies>

1. Demo:

<http://getcontenttools.com/demo>

1. Getting Started:

<http://getcontenttools.com/getting-started>

1. Handling image uploads:

<http://getcontenttools.com/tutorials/handling-image-uploads>

1. Adding and manging new tools:

<http://getcontenttools.com/tutorials/adding-new-tools>

1. Multilingual support:

<http://getcontenttools.com/tutorials/multilingual-support>

1. Integration of ContentTools with CMS:

<http://getcontenttools.com/tutorials/content-tools-plus-cms>

1. Django integration :

<https://github.com/Cotidia/django-contenttools-demo>

1. Image uploads using Cloudinary: <http://getcontenttools.com/tutorials/image-uploads-with-cloudinary>
2. HTML string for formatting:

<http://getcontenttools.com/api/html-string>

1. Content-selection:

<http://getcontenttools.com/api/content-select>

1. Content-Edit:

<http://getcontenttools.com/api/content-edit>

**X.5.5. Idea of Implementation**

In a similar way in which the Open edX platform uses an xmodule(html\_module.py) that handles the operation of its current html editor, similarly, using the same working principle we can replace our editor and load the html editor and configure it for proper working.

Integrating this would be much easier compared to Brackets and Notepad++ as its working methodology is just as similar to the CodeMirror and TinyMCE which are being used as of now in the OpenEdx platform.

**X.5.6.INSTALLATION ISSUES**

Installing Notepad++ and Brackets easy was error free but with ContentTools the only issue that occurred while installing was related to the npm package installer.

1. The packages were not completely installed and hence npm threw an error while installing.

Here is the log file that shows the errors that would be occurring .

[npm issues.docx](npm%20issues.docx)

*Solution:* to resolve this issue here are the things that are to be followed:

1. Open your terminal and perform the following operations

***sudo apt-get update***

***synapti***c

2. A dailog box opensSearch for npm and node modules

3. Delete those folders manually

4. Save the changes and close the dialog.

5. Open the terminal

***npm install --save ContentTools***

**X.6.Advantages of Approach1 compared to Approach2**

1. Upgrading something is always better than integrating something new altogether. As per our research and observation the versions of CodeMirror and TinyMCE in Open edX are outdated. Since the latest versions of these two editors fullfill our requirements updating the editors is much straighforward and sensible than incorporating two completedly new ediitors into the Open edX platform, since it can lead to a lot of dependency issues.

2. The iframe approach also allows the course content creator to include third party CSS such as bootstrap, w3css which will make the course even more interactive and boost student experience. Course creators can style their courses with a variety of such third party CSS features and make it a lot more interactive for the students.

3. Wrapping data inside the iframes containerizes the data ensuring no mixing of CSS between two HTML blocks in LMS. This is very important in order to provide an error free user experience.