perplexity

Comprehensive Text Editor Components and Technical Implementation Guide

Based on your current implementation using standard HTML inputs and textareas, here's a detailed technical guide for upgrading to modern text editor components that will significantly enhance your AI Product Leader portfolio admin dashboard.

Current Implementation Analysis

Your existing setup with standard HTML `<input>` and `<textarea>` elements provides basic functionality but lacks the rich text editing capabilities essential for professional content management [1]. While this approach offers simplicity and reliability, it limits your ability to create compelling, formatted content that showcases your AI expertise effectively [2].

Recommended Text Editor Components

1. TinyMCE - Feature-Rich WYSIWYG Editor

TinyMCE stands out as one of the most comprehensive rich text editors available, offering extensive formatting capabilities and seamless React integration [3]. This editor excels in providing familiar Microsoft Word-like functionality that makes content creation intuitive for users [4].

Technical Implementation:

```
import { Editor } from '@tinymce/tinymce-react';

const TinyMCEEditor = ({ value, onChange }) => {
  return (
    <Editor
      apiKey="your-api-key-here"
      value={value}
      onEditorChange={onChange}
      init={{
        height: 400,
        menubar: false,
        plugins: [
          'advlist', 'autolink', 'lists', 'link', 'image', 'charmap',
          'anchor', 'searchreplace', 'visualblocks', 'code', 'fullscreen',
          'insertdatetime', 'media', 'table', 'preview', 'help', 'wordcount'
        ],
        toolbar: 'undo redo | blocks | bold italic forecolor | alignleft aligncenter alig
      }}
    />
```

```
  );
};
```

2. Quill - Lightweight and Modular

Quill offers a perfect balance between functionality and performance, making it ideal for applications requiring fast, responsive text editing [5]. Its modular architecture allows you to include only necessary features, reducing bundle size and improving loading times [6].

Technical Implementation:

```
import ReactQuill from 'react-quill';
import 'react-quill/dist/quill.snow.css';

const QuillEditor = ({ value, onChange }) => {
  const modules = {
    toolbar: [
      [{ 'header': [1, 2, false] }],
      ['bold', 'italic', 'underline', 'strike', 'blockquote'],
      [{'list': 'ordered'}, {'list': 'bullet'}, {'indent': '-1'}, {'indent': '+1'}],
      ['link', 'image'],
      ['clean']
    ],
  };

  return (
    <ReactQuill
      theme="snow"
      value={value}
      onChange={onChange}
      modules={modules}
    />
  );
};
```

3. Tiptap - Modern and Extensible

Tiptap represents the cutting edge of rich text editors, offering exceptional customization capabilities and a developer-friendly API [7]. Industry experts consistently rank Tiptap as the most well-rounded choice for 2025, striking an optimal balance between features and flexibility [2].

Technical Implementation:

```
import { useEditor, EditorContent } from '@tiptap/react';
import StarterKit from '@tiptap/starter-kit';
import { Bold, Italic } from '@tiptap/extension-bold';

const TiptapEditor = ({ content, onChange }) => {
  const editor = useEditor({
    extensions: [StarterKit],
```

```
    content: content,
    onUpdate: ({ editor }) => {
      onChange(editor.getHTML());
    },
  });

  return (
    <div className="tiptap-editor">
      <div className="toolbar">
        <button
          onClick={() => editor.chain().focus().toggleBold().run()}
          className={editor.isActive('bold') ? 'is-active' : ''}
        >
          Bold
        </button>
        <button
          onClick={() => editor.chain().focus().toggleItalic().run()}
          className={editor.isActive('italic') ? 'is-active' : ''}
        >
          Italic
        </button>
      </div>
      <EditorContent editor={editor} />
    </div>
  );
};
```

4. Lexical - Meta's Advanced Framework

Lexical, developed by Meta (Facebook), powers some of the world's most widely used applications and offers exceptional performance for complex editing scenarios [8]. This framework excels in handling large documents and provides advanced features like collaborative editing and real-time synchronization [9].

Technical Implementation:

```
import { $getRoot, $getSelection } from 'lexical';
import { LexicalComposer } from '@lexical/react/LexicalComposer';
import { PlainTextPlugin } from '@lexical/react/LexicalPlainTextPlugin';
import { ContentEditable } from '@lexical/react/LexicalContentEditable';
import { HistoryPlugin } from '@lexical/react/LexicalHistoryPlugin';

const LexicalEditor = ({ onChange }) => {
  const initialConfig = {
    namespace: 'MyEditor',
    onError: (error) => console.error(error),
  };

  return (
    <LexicalComposer initialConfig={initialConfig}>
      <PlainTextPlugin
        contentEditable={<ContentEditable />}
        placeholder={<div>Enter some text...</div>}
      />
```

```
      <HistoryPlugin />
    </LexicalComposer>
  );
};
```

Technical Architecture Considerations

Database Schema Enhancement

To support rich text content, you'll need to modify your database schema to accommodate HTML content and metadata [10]:

```
-- Enhanced content table structure
CREATE TABLE portfolio_content (
  id SERIAL PRIMARY KEY,
  section VARCHAR(100) NOT NULL,
  content_type VARCHAR(50) DEFAULT 'rich_text',
  content_html TEXT,
  content_markdown TEXT,
  meta_data JSONB,
  version_number INTEGER DEFAULT 1,
  created_at TIMESTAMP DEFAULT NOW(),
  updated_at TIMESTAMP DEFAULT NOW()
);

-- Version history table
CREATE TABLE content_versions (
  id SERIAL PRIMARY KEY,
  content_id INTEGER REFERENCES portfolio_content(id),
  html_content TEXT,
  change_summary VARCHAR(255),
  created_at TIMESTAMP DEFAULT NOW()
);
```

Content Management API Implementation

Your backend API should handle rich text content with proper sanitization and version control [11]:

```
// Content management API endpoint
app.post('/api/content/update', async (req, res) => {
  const { section, htmlContent, markdownContent } = req.body;

  // Sanitize HTML content to prevent XSS attacks
  const sanitizedHTML = DOMPurify.sanitize(htmlContent);

  try {
    // Save current version to history
    await saveContentVersion(section, sanitizedHTML);

    // Update main content
```

```
    const result = await db.portfolio_content.update({
      content_html: sanitizedHTML,
      content_markdown: markdownContent,
      updated_at: new Date()
    }, {
      where: { section: section }
    });

    res.json({ success: true, message: 'Content updated successfully' });
  } catch (error) {
    res.status(500).json({ error: 'Failed to update content' });
  }
});
```

Auto-Save Implementation

Implement intelligent auto-save functionality to prevent content loss [12]:

```
const useAutoSave = (content, onSave, delay = 2000) => {
  const [isSaving, setIsSaving] = useState(false);
  const [lastSaved, setLastSaved] = useState(null);

  useEffect(() => {
    const timeoutId = setTimeout(async () => {
      if (content && content.trim() !== '') {
        setIsSaving(true);
        try {
          await onSave(content);
          setLastSaved(new Date());
        } catch (error) {
          console.error('Auto-save failed:', error);
        } finally {
          setIsSaving(false);
        }
      }
    }, delay);

    return () => clearTimeout(timeoutId);
  }, [content, onSave, delay]);

  return { isSaving, lastSaved };
};
```

Integration with Your Admin Dashboard

Component Architecture

Integrate the text editor components into your existing dashboard structure [13]:

```
const ContentEditor = ({ section, initialContent }) => {
  const [content, setContent] = useState(initialContent);
  const [isDirty, setIsDirty] = useState(false);
```

```
  const { isSaving, lastSaved } = useAutoSave(
    content,
    async (content) => {
      await updateContent(section, content);
      setIsDirty(false);
    }
  );

  const handleContentChange = (newContent) => {
    setContent(newContent);
    setIsDirty(true);
  };

  return (
    <div className="content-editor">
      <div className="editor-header">
        <h3>Edit {section}</h3>
        <div className="save-status">
          {isSaving && <span>Saving...</span>}
          {lastSaved && <span>Last saved: {formatTime(lastSaved)}</span>}
          {isDirty && <span className="unsaved">Unsaved changes</span>}
        </div>
      </div>

      <TinyMCEEditor
        value={content}
        onChange={handleContentChange}
      />

      <div className="editor-actions">
        <button onClick={() => saveContent(section, content)}>
          Save Now
        </button>
        <button onClick={() => previewContent(content)}>
          Preview
        </button>
      </div>
    </div>
  );
};
```

Performance Optimization

Implement lazy loading and code splitting for better performance [14]:

```
// Lazy load text editor components
const TinyMCEEditor = lazy(() => import('./TinyMCEEditor'));
const QuillEditor = lazy(() => import('./QuillEditor'));
const TiptapEditor = lazy(() => import('./TiptapEditor'));

const EditorSelector = ({ editorType, ...props }) => {
  return (
    <Suspense fallback={<div>Loading editor...</div>}>
      {editorType === 'tinymce' && <TinyMCEEditor {...props} />}
```

```
      {editorType === 'quill' && <QuillEditor {...props} />}
      {editorType === 'tiptap' && <TiptapEditor {...props} />}
    </Suspense>
  );
};
```

Recommendation for Your Portfolio

For your AI Product Leader portfolio admin dashboard, I recommend starting with **TinyMCE** due to its comprehensive feature set and enterprise-grade reliability [15]. TinyMCE provides the professional editing experience necessary for creating compelling case studies and technical content while maintaining the clean HTML output essential for SEO and performance [16].

The implementation should include auto-save functionality, version control, and proper content sanitization to ensure your portfolio content remains secure and professionally presented [17]. This upgrade will transform your basic text inputs into a sophisticated content management system worthy of your AI product leadership expertise [18].

Consider implementing a hybrid approach where simple fields (titles, headlines) remain as standard inputs for performance, while complex content areas (case study descriptions, technical explanations) utilize the rich text editor for enhanced formatting capabilities [19]. This strategy optimizes both user experience and system performance while providing the flexibility needed for professional content creation [20].

⁂

1. https://wedevs.com/blog/334733/best-text-editors/
2. https://liveblocks.io/blog/which-rich-text-editor-framework-should-you-choose-in-2025
3. https://www.npmjs.com/package/react-draft-wysiwyg
4. https://www.linkedin.com/pulse/top-text-editing-software-options-content-creators-digital-ambekar-sp0vf
5. https://www.npmjs.com/package/react-simple-wysiwyg
6. https://ckeditor.com/solutions/content-management/
7. https://froala.com/blog/general/10-essential-tools-for-visual-html-editor-workflows-in-react-on-2025/
8. https://dev.to/joodi/10-top-rich-text-editors-for-react-developers-in-2025-5a2m
9. https://strapi.io/blog/best-wysiwyg-editors
10. https://www.dhiwise.com/post/tiptap-vs-lexical-choosing-the-best-web-text-editor
11. https://www.syncfusion.com/react-components/react-rich-text-editor
12. https://basicutils.com/learn/quilljs/top-7-react-rich-text-editors
13. https://www.reddit.com/r/vuejs/comments/1c2al6u/which_vue_editor_library_quill_tiptap_or_lexical/
14. https://ej2.syncfusion.com/react/documentation/rich-text-editor/getting-started
15. https://www.tiny.cloud/blog/upgrade-react-textarea-with-rich-text-editor-and-vite/
16. https://dev.to/just_ritik/mastering-react-quill-a-step-by-step-guide-to-implement-a-rich-text-editor-in-your-react-app-5edn

17. https://ckeditor.com/docs/ckeditor5/latest/getting-started/installation/self-hosted/react/react-default-npm.html
18. https://docs.expo.dev/guides/editing-richtext/
19. https://www.tiny.cloud/solutions/cms-editor/
20. https://froala.com/blog/general/what-is-wysiwyg-editor-a-comprehensive-developers-insight/