No-code Content Editor

To empower our internal teams and accelerate content creation, we are launching a strategic initiative to develop an in-house, "No-Code" visual editor, codenamed "Aura".

The long-term vision is to provide teams like Marketing and Product with a tool that allows them to build and modify simple web layouts and promotional materials without direct engineering support

Please refer the below online tools to understand the editor functionality, how to select and drop an element on the canvas and then changing the properties of that element:

* [Beefree](https://beefree.io/start-designing?template=empty&type=page&catalog=true&e=true)
* [GrapesJs](https://grapesjs.com/demo.html)

**1. Core Functional** **Requirements**

* **Component Palette:** A panel displays a list of pre-defined components that can be added to the editor.
* **Drag and Drop:** Users must be able to drag components from the palette and drop them onto a freeform canvas area.
* **Freeform Canvas:** Dropped components can be selected and moved freely to any position within the canvas.
* **Dynamic Properties Panel:** When a component on the canvas is selected, a properties panel must display the specific settings for that component.
* **Real-Time Updates:** Any change made in the properties panel must be reflected on the selected component in the canvas instantly.
* **Preview & Copy Html:** Provide the ability to preview the final html and also copy the final html to clipboard.
* **For I3 and above members:**
  + **Undo/Redo:** The application must support a history of at least **50 steps**. Users must be able to undo/redo their actions (e.g., adding a component, moving a component, changing a property), with the canvas updating to reflect the state at each step.
  + **Custom Inline tool Editor:** Develop a custom inline tool editor for Text and TextArea elements (Do not use inline ckeditor)
  + **Mobile & Desktop Preview**: The Preview functionality should support Mobile view as well as Desktop view.
* **Please note: We do not need to build/integrate API to store/persist the content. Kindly leverage browser localstorage for this, treat localstorage as the DB.**

**2. UI Layout**

The application should have a three-panel layout:

* **Left (20% width): Palette Panel** - Contains the list of available components.
* **Middle (60% width): Canvas Panel** - The main interactive area where components are placed and manipulated.
* **Right (20% width): Properties Panel** - Displays the configuration options for the currently selected component.

**3. Component Palette & Properties**

The Palette should contain the following components. When selected, the Properties Panel should display the corresponding options.

* **Text**
  + The below properties should be provided on the right panel:
    - **Font-size**: Number Input / Slider
    - **Font-weight**: Dropdown ('400 - Normal', '700 - Bold')
    - **Color**: color picker
* **TextArea**
  + The below properties should be provided on the right panel:
    - **Font-size**: Number Input / Slider
    - **Color**: color picker
    - **Text-Align**: Button Group ('Left', 'Center', 'Right')
* **Image**
  + The below properties should be provided on the right panel:
    - **Image URL**: Text Input
    - **Alt Text**: Text Input
    - **Object Fit**: Dropdown ('Cover', 'Contain', 'Fill')
    - **Border Radius**: Number Input / Slider
    - **Height**: Number Input / Slider
    - **Width**: Number Input / Slider
* **Button**
  + The below properties should be provided on the right panel:
    - **URL**: Text Input
    - **Button Text**: Text Input
    - **Font-size**: Number Input / Slider
    - **Padding:** Number Input / Slider
    - **Background Color:** Color Picker
    - **Text Color:** Color Picker
    - **Border Radius:** Number Input / Slider

**4. Technical Constraints & Rules**

* The use of comprehensive, "all-in-one" solution libraries (e.g., GrapesJS, HTML5 canvas, reactdnd etc) that solve the entire problem or part of problem is **prohibited**.
* The drag-and-drop functionality for positioning and moving components on the canvas **must be built from scratch** using native browser events (mousedown, mousemove, etc.). For this specific requirement, UI interaction libraries like react-dnd, konva.js, interact.js, etc., are **prohibited**.

**5. Deliverables**

Create a private GitHub repo and share with the below members:

* dmistryTal
* nileshmallick1606
* Sachin-Salunke-Talentica

**Kindly note the names of each of the expected files should be the same. The automated evaluation mechanism expects that those file names are accurate, if not then it will impact the final score.**

Your submission will be a single private GitHub repository containing the following:

1. **Source Code:** The complete, running source code, including any data simulators required for testing.
2. README.md: A clear overview of the project and detailed, step-by-step instructions on how to build and run the entire system locally.
3. PROJECT\_STRUCTURE.md: Explaining the structure of the project and the purpose foreach of the folder and key modules.
4. ARCHITECTURE.md:
   * **This is the most important document.** It must detail and **justify** your design.
   * An explanation of your chosen architectural pattern and why you selected it.
   * A diagram illustrating the components and their communication flows.
   * A **Technology Justification** section explaining your choice of major technologies (e.g., from the landscape above, or alternatives) and why they were the best fit for this problem.
   * A section explaining the "why" behind your state management strategy, component structure, and your approach to the undo/redo functionality.
5. CHAT\_HISTORY.md**:** A summary document that chronicles your design journey with your AI assistant, highlighting key decision points and how you used AI to evaluate alternatives.
6. **Video**: An 8-10 min video explaining:
   * Design, architecture and the different components and how they communicate with each other.
   * Explain the journey from initial brainstorming till the final implementation and the conversation with the coding assistant.
   * Key decisions and trade-offs.
   * Demo of the entire working of the application.
   * Test case coverage %
   * **Upload the video on your one drive and share the access with the below members:**
     1. [Dipen.mistry@talentica.com](mailto:Dipen.mistry@talentica.com)
     2. [Nilesh.Mallick@talentica.com](mailto:Nilesh.Mallick@talentica.com)
     3. [Sachin.Salunke@talentica.com](mailto:Sachin.Salunke@talentica.com)