DOCUMENTATION

# ON

**Web Based Writing Tool**

*Prepared by*

**Abhijit Kumar Mahato (dev.abhijitkrma@gmail.com)**

**Date**: 09-04-2024

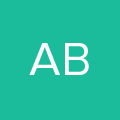
**CHALLENGES FACED WHILE DEVELOPING THE PRODUCT:**

* **Learned AntDesign**
* **Implementing Picture adding feature**
* **Inserting content and implement basic editors**: Explored TipTap Rich Editor.
* **Inserting cards at any position (above/below)**
* **Hovering and Focusing the Text content of card to show the basic tools**
* **Making the cards/blocks draggable**: Explored Framer Motion.

**PRODUCT FEATURES THAT CAN BE INCLUDED AS NEXT STEPS:**

* **Collaborative Editing**
* **Export and Import Options**
* **Customizable Themes and Styling**
* **Include backend/local storage to save the changes**
* **Spelling and Grammar Check**

**CHATGPT/LLM PROMPTS USED TO WRITE THE CODE:**



**You**

template to create modal using ant design in react

ChatGPT

**ChatGPT**

Creating a modal using Ant Design in React involves a few steps. First, you'll need to install Ant Design and its dependencies if you haven't already. Then you can create a modal component and use it where needed. Below is a template to help you create a modal using Ant Design in React:

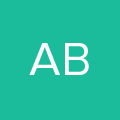
jsxCopy code

// Import necessary modules import React, { useState } from 'react'; import { Modal, Button } from 'antd'; // Create ModalComponent const ModalComponent = () => { // State to manage modal visibility const [visible, setVisible] = useState(false); // Function to handle modal open const showModal = () => { setVisible(true); }; // Function to handle modal close const handleCancel = () => { setVisible(false); }; return ( <div> {/\* Button to trigger modal \*/} <Button type="primary" onClick={showModal}> Open Modal </Button> {/\* Modal component \*/} <Modal title="Basic Modal" visible={visible} onCancel={handleCancel} footer={[ <Button key="back" onClick={handleCancel}> Cancel </Button>, <Button key="submit" type="primary" onClick={handleCancel}> OK </Button>, ]} > <p>Modal content goes here...</p> </Modal> </div> ); }; export default ModalComponent;

In this template:

1. We import the necessary modules from Ant Design (**Modal**, **Button**).
2. We define a functional component called **ModalComponent**.
3. Inside the component, we define state using the **useState** hook to manage the visibility of the modal.
4. We define two functions **showModal** and **handleCancel** to control the visibility of the modal.
5. Inside the JSX, we use a **Button** component to trigger the modal, and the **Modal** component to display the modal content. We pass **visible** state to control the visibility of the modal, and **onCancel** to handle modal close event.
6. In the **Modal** component, we define a title, footer with OK and Cancel buttons, and the content of the modal.

You can use this **ModalComponent** in your React application by importing it and rendering it wherever needed.



**You**

how to make cards draggable in react using framer motion

2 / 2

ChatGPT

**ChatGPT**

To make cards draggable in React using Framer Motion, you can follow these steps:

1. Install Framer Motion if you haven't already:

bashCopy code

npm install framer-motion

1. Create a draggable card component using Framer Motion.

Here's an example implementation:

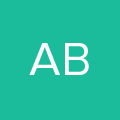
jsxCopy code

import React from 'react'; import { motion } from 'framer-motion'; const DraggableCard = () => { return ( <motion.div drag dragConstraints={{ left: 0, right: 0, top: 0, bottom: 0 }} style={{ width: 200, height: 200, backgroundColor: 'lightblue', borderRadius: 10, padding: 20, margin: 20, boxShadow: '0px 4px 10px rgba(0, 0, 0, 0.1)', }} > Drag me! </motion.div> ); }; export default DraggableCard;

In this code:

* We import **motion** from **framer-motion**.
* We create a **DraggableCard** component.
* Inside the component, we use the **motion.div** component provided by Framer Motion.
* We add the **drag** prop to make the card draggable.
* We specify **dragConstraints** to limit the dragging area. In this case, the card can be dragged within its parent element.
* We style the card with some basic styles.

You can use this **DraggableCard** component in your React application wherever you need draggable cards. Simply import it and render it as desired.



**You**

insert picture using ant design

ChatGPT

**ChatGPT**

To insert a picture using Ant Design in React, you can use the **Image** component provided by Ant Design. Below is an example of how to use the **Image** component:

First, make sure you have Ant Design installed in your project:

bashCopy code

npm install antd

Then, you can create a component and use the **Image** component like this:

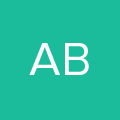
jsxCopy code

import React from 'react'; import { Image } from 'antd'; const PictureComponent = () => { return ( <div> <Image width={200} src="https://via.placeholder.com/200" alt="Placeholder Image" /> </div> ); }; export default PictureComponent;

In this example:

* We import the **Image** component from 'antd'.
* Inside the **PictureComponent** component, we render the **Image** component.
* We provide the **width** prop to specify the width of the image.
* We provide the **src** prop to specify the URL of the image.
* We provide the **alt** prop for the alternative text of the image.

You can replace the **src** prop value with the URL of the image you want to display. Additionally, you can customize the width, height, and other properties of the **Image** component as needed.



**You**

insert element in any position of array in react js

2 / 2

ChatGPT

**ChatGPT**

To insert an element at any position in an array in React.js, you can use the **splice()** method or the spread operator (**...**). Below are examples of both methods:

Using **splice()**:

javascriptCopy code

const newArray = [...originalArray]; // Make a shallow copy of the original array newArray.splice(index, 0, elementToAdd); // Insert the element at the desired index

Using spread operator (**...**):

javascriptCopy code

const newArray = [ ...originalArray.slice(0, index), // Get elements before the index elementToAdd, // Insert the element at the desired index ...originalArray.slice(index) // Get elements after the index ];

Here's how you can use these methods in a React component:

jsxCopy code

import React, { useState } from 'react'; const MyComponent = () => { const [array, setArray] = useState([1, 2, 3, 5, 6]); // Example array const indexToInsert = 3; // Example index const elementToAdd = 4; // Example element to add // Function to insert an element at a specific index const insertElement = (index, element) => { const newArray = [...array]; // Make a shallow copy of the original array newArray.splice(index, 0, element); // Insert the element at the desired index setArray(newArray); // Update the state with the new array }; return ( <div> <p>Original Array: {array.join(', ')}</p> <button onClick={() => insertElement(indexToInsert, elementToAdd)}> Insert Element at Index {indexToInsert} </button> <p>New Array: {array.join(', ')}</p> </div> ); }; export default MyComponent;

In this example:

* We have a component **MyComponent** that maintains an array state using **useState**.
* We define a function **insertElement** that takes an index and an element to be inserted.
* Inside **insertElement**, we create a shallow copy of the original array using the spread operator (**...**) and then use **splice()** to insert the element at the specified index.
* We update the state with the new array using **setArray**.
* We render a button that calls **insertElement** when clicked.
* We display the original and the updated arrays in the component.

Top of Form

Top of Form

Top of Form