# Dynamic Tabs Component Documentation

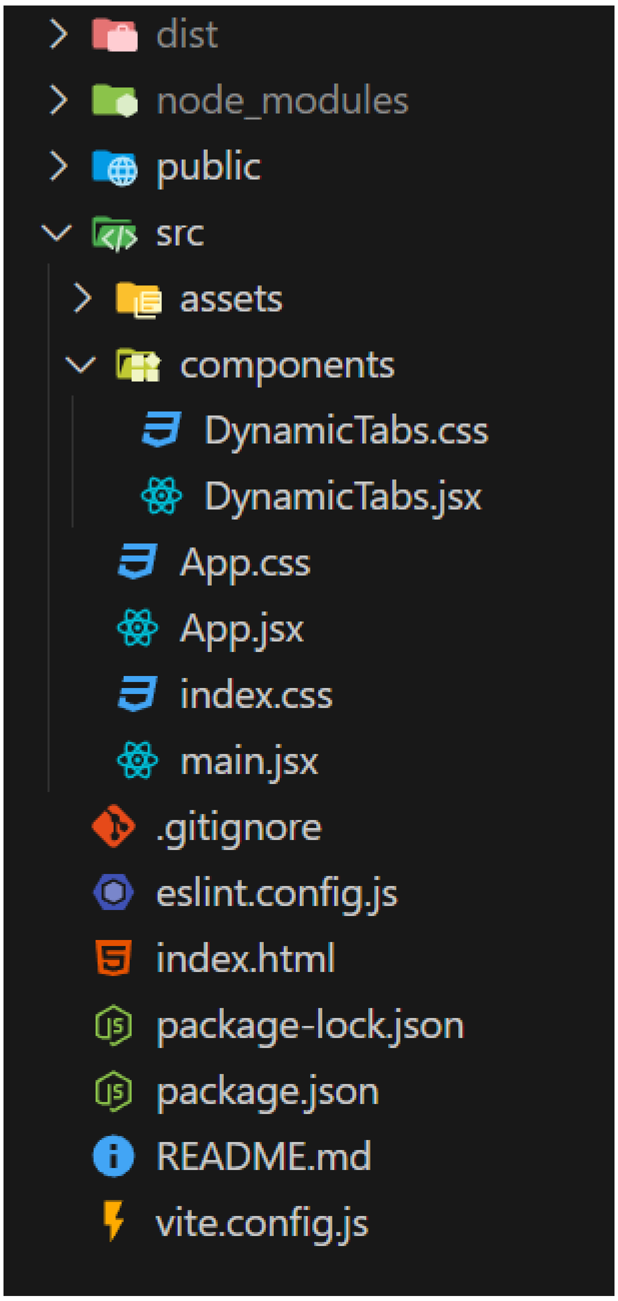
## Overview

The **Dynamic Tabs Component** is a React-based UI component that allows users to dynamically add, remove, and switch between tabs. Each tab has a title and content, and the component ensures that at least one tab is always present. The active tab is visually distinguished from the inactive ones, and users can interact with the tabs to add new ones or remove existing ones.

### Features

1. **Add Tabs**: Users can add new tabs by clicking the "Add Tab" button. Each new tab will have a default title and content.
2. **Switch Tabs**: Users can switch between tabs by clicking on the tab header.
3. **Remove Tabs**: Users can remove any tab except the last one. The component ensures that at least one tab remains.
4. **Active Tab Indicator**: The active tab is highlighted with a distinct border color.
5. **Responsive Design**: The tabs are designed to wrap to the next line if there are too many to fit in a single row.

**Folder Structure**



```
> dist
> node_modules
> public
v src
  > assets
  v components
      DynamicTabs.css
      DynamicTabs.jsx
    App.css
    App.jsx
    index.css
    main.jsx
  .gitignore
  eslint.config.js
  index.html
  package-lock.json
  package.json
  README.md
  vite.config.js
```

```
;
```

## Code Explanation

## React Component

```
import React, { useState } from 'react';
import './DynamicTabs.css';

const DynamicTabs = () => {
  // State to manage the list of tabs
  const [tabs, setTabs] = useState([
    { id: 1, title: 'Tab 1', content: 'Content for Tab 1' },
    { id: 2, title: 'Tab 2', content: 'Content for Tab 2' },
  ]);

  // State to manage the active tab
  const [activeTab, setActiveTab] = useState(1);

  // State to generate unique IDs for new tabs
  const [nextId, setNextId] = useState(3);

  // Function to add a new tab
  const addTab = () => {
    const newTab = {
      id: nextId,
      title: `Tab ${nextId}`,
      content: `Content for Tab ${nextId}`,
    };
    setTabs([...tabs, newTab]); // Add the new tab to the list
    setActiveTab(newTab.id); // Set the new tab as active
    setNextId(nextId + 1); // Increment the ID for the next tab
  };

  // Function to remove a tab
  const removeTab = (id) => {
    if (tabs.length > 1) { // Ensure at least one tab remains
      const updatedTabs = tabs.filter((tab) => tab.id !== id); // Remove the tab
      setTabs(updatedTabs);
      if (activeTab === id) {
        setActiveTab(updatedTabs[0].id); // Set the first tab as active if the
removed tab was active
      }
    }
  };

  return (
    <div className="tabs-container">
      {/* Button to add a new tab */}
      <button className="add-tab" onClick={addTab}>
        Add Tab
      </button>

      {/* Tab headers */}
      <div className="tabs-header">
        {tabs.map((tab) => (
          <div
      key={tab.id}
      className={`tab ${activeTab === tab.id ? 'active' : ''}`}
      onClick={() => setActiveTab(tab.id)}
    >
      {tab.title}
      {/* Button to close the tab */}
      {tabs.length > 1 && (
        <button
          className="close-tab"
          aria-label={`Close ${tab.title}`}
          onClick={(e) => {
            e.stopPropagation(); // Prevent the tab from being activated
            removeTab(tab.id);
          }}
        >
          &times;
        </button>
      )}
    </div>
  ))}
</div>

{/* Tab content */}
<div className="tabs-content">
  {tabs.find((tab) => tab.id === activeTab)?.content}
</div>
```

```
    </div>
  );
};
export default DynamicTabs;
```

## CSS Styling

```
.tabs-container {
    max-width: 800px;
    margin: 0 auto;
    font-family: Arial, sans-serif;
    background-color: #121212;
    padding: 10px; /* Optional: Add some padding */
}

.tabs-header {
    margin: 0 30px;
    color: whitesmoke;
    display: flex;
    flex-wrap: wrap; /* Allow tabs to wrap to the next line */
    gap: 10px; /* Add gap between tabs */
}

.tab {
    width: 100px;
    padding: 10px 20px;
    cursor: pointer;
    position: relative;
    border-top-left-radius:10px;
    border-top-right-radius: 10px;
    background-color:#3c3c3d;
    flex-shrink: 0; /* Prevent tabs from shrinking */
}

.tab.active {
    background-color: #2d2d2d;
    border:2px solid #1DB954;
}

.close-tab {
    margin-left: 15px;
    color: whitesmoke;
    font-size: 1.8rem;
    cursor: pointer;
    background: none;
    border: none;
}
.close-tab:hover {
    color:#1DB954;
    transform: scale(1.5);
}
.add-tab {
    margin-top: 20px;
    margin-bottom: 20px;
    padding: 10px 20px;
    font-size: 1.5rem;
    cursor: pointer;
    background-color: #333333;
    color: white;
    border: none;
    border-radius: 5px;
    margin-left: 30px;
    flex-shrink: 0; /* Prevent the add button from shrinking */
}

.add-tab:hover {
    color:#1DB954;
    border: 2px solid #1DB954;
}

.tabs-content {
    margin:20px 30px;
    padding: 20px;
    border: 1px solid #ccc;
    color:whitesmoke;
}
.tabs-content:hover{
    font-size: 1.2rem;
    border:2px solid #14dc5a;
}
```

## Usage

1. **Adding a Tab**: Click the "Add Tab" button to add a new tab. The new tab will be automatically set as the active tab.
2. **Switching Tabs**: Click on any tab header to switch to that tab.
3. **Removing a Tab**: Click the close button (×) on a tab to remove it. The component ensures that at least one tab remains.

## Styling

- **Active Tab**: The active tab is highlighted with a green border (#1DB954).
- **Inactive Tabs**: Inactive tabs have a darker background (#3c3c3d).
- **Close Button**: The close button (×) appears on each tab (except the last one) and changes color on hover.
- **Add Tab Button**: The "Add Tab" button changes color and border on hover.

## Constraints

1. **At Least One Tab**: The component ensures that there is always at least one tab.
2. **Unique IDs**: Each tab has a unique ID to handle addition and removal correctly.
3. **State Management**: The component uses React's useState to manage the list of tabs and the active tab.

## Edge Cases

- If the active tab is removed, the first tab in the list becomes the new active tab.
- The component prevents the removal of the last tab to ensure there is always at least one tab.

## Conclusion

The **Dynamic Tabs Component** is a flexible and user-friendly UI element that allows users to manage multiple tabs dynamically. It is built using React and styled with CSS to provide a clean and responsive user experience.