Component Overview

The SearchBar component is a functional React component that manages the following:

1. **Search Input**: Allows users to type a query to search for recipes.
2. **Debounced API Calls**: Ensures that API requests are made only after 500ms of inactivity to optimize performance.
3. **Auto-Suggestions**: Displays a dropdown list of recipe suggestions based on the user's input.
4. **Recipe Details**: Shows detailed information about a selected recipe.
5. **All Recipes List**: Displays a grid of all available recipes fetched from the API.

State Management

The component uses the useState hook to manage the following state variables:

- **query**: Stores the current search query entered by the user.
- **suggestions**: Stores the list of recipe suggestions fetched from the API based on the search query.
- **selectedRecipe**: Stores the details of the recipe selected by the user.
- **recipes**: Stores the list of all recipes fetched from the API.

API Fetching

The component uses the useEffect hook to fetch data from the API:

1. **Initial Fetch**:
    - On component mount, it fetches all recipes from https://dummyjson.com/recipes and stores them in the recipes state.
    -

2. **Debounced Fetch**:
   - A second useEffect hook is used to debounce the API calls. It triggers a fetch request only after 500ms of no typing.
   - The fetchRecipes function is called with the current query to fetch matching recipes from https://dummyjson.com/recipes/search?q={query}.

Debouncing

Debouncing is implemented using setTimeout and clearTimeout:

- A timer (debounceTimer) is set for 500ms whenever the user types in the search input.
- If the user continues typing within 500ms, the previous timer is cleared, and a new one is set.
- This ensures that the API call is made only after the user has stopped typing for 500ms.

Event Handlers

1. **handleSuggestionClick**:
   - This function is triggered when a user clicks on a suggestion from the dropdown.
   - It updates the query state with the selected recipe's name, sets the selectedRecipe state to display the recipe details, and clears the suggestions list.

Rendering

The component renders the following UI elements:

1. **Search Bar**:
   - An input field where users can type their search query.

- A dropdown list of suggestions (suggestions-list) is displayed below the input if there are matching recipes.
- If no recipes match the query, a "No recipes found" message is displayed.

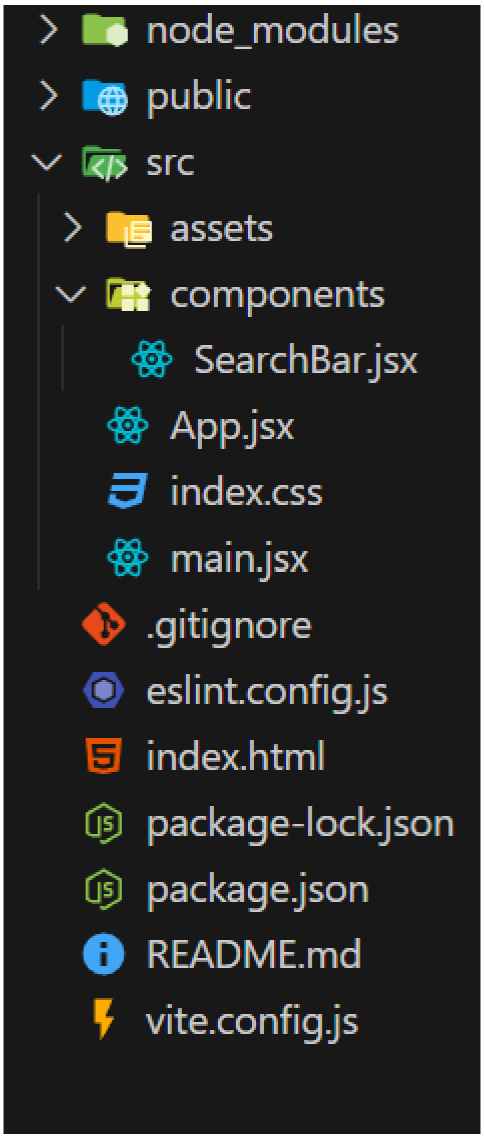
2. **Selected Recipe Details**:
    - If a recipe is selected, its details (name, cuisine, ingredients, and image) are displayed in a card.
3. **All Recipes Grid**:
    - A grid layout displays all recipes fetched from the API.
    - Each recipe card shows the recipe's image, name, preparation time, cooking time, and the number of ingredients.

Styling

The component uses CSS for styling, with a focus on responsiveness and modern design:

1. **Dark Theme**:
    - The background is dark (#121212), and the text is light (#e0e0e0) for better readability.
    - Gradient backgrounds and vibrant colors are used for cards and hover effects.
2. **Responsive Design**:
    - The layout adjusts based on screen size:
        - **Mobile**: 1 column grid.
        - **Tablet**: 2 columns grid.
        - **Desktop**: 3 columns grid.
3. **Hover Effects**:
    - Recipe cards and suggestion items have hover effects (e.g., scale-up, color change) to enhance user interaction.

Folder Structure



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

Code Breakdown

1. State Initialization

```
const [query, setQuery] = useState("");

const [suggestions, setSuggestions] = useState([]);

const [selectedRecipe, setSelectedRecipe] = useState(null);

const [recipes, setRecipes] = useState([]);
```

2. Fetching All Recipes on Mount

```
useEffect(() => {
  fetch("https://dummyjson.com/recipes")
    .then((res) => res.json())
    .then((data) => setRecipes(data.recipes))
    .catch((error) => console.error("Error fetching recipes:", error));
}, []);
```

3. Debounced API Call

```
useEffect(() => {
  const debounceTimer = setTimeout(() => {
    if (query.trim()) {
      fetchRecipes(query);
    } else {
      setSuggestions([]);
    }
  }, 500);

  return () => clearTimeout(debounceTimer);
}, [query]);
```

4. Fetching Recipes Based on Query

```
const fetchRecipes = async (searchQuery) => {
  try {
    const response = await axios.get(
      `https://dummyjson.com/recipes/search?q=${searchQuery}`
    );
    setSuggestions(response.data.recipes);
  } catch (error) {
    console.error("Error fetching recipes:", error);
    setSuggestions([]);
  }
};
```

5. Handling Suggestion Click

```
const handleSuggestionClick = (recipe) => {
  setQuery(recipe.name);
  setSelectedRecipe(recipe);
  setSuggestions([]);
};
```

6. Rendering the Search Bar and Suggestions

```
<div className="search-bar">
  <input
    type="text"
    placeholder="Search for recipes..."
    value={query}
    onChange={(e) => setQuery(e.target.value)}
    className="search-input"
  />
  {suggestions.length > 0 && (
    <ul className="suggestions-list">
      {suggestions.map((recipe) => (
        <li
          key={recipe.id}
          onClick={() => handleSuggestionClick(recipe)}
          className="suggestion-item"
        >
          {recipe.name}
        </li>
      ))}
    </ul>
  )}
  {query && suggestions.length === 0 && (
    <p className="no-results">No recipes found.</p>
  )}
</div>
```

7. Rendering Selected Recipe Details

```
{selectedRecipe && (
 <div className="recipe-card">
  <h2 className="recipe-title">{selectedRecipe.name}</h2>
  <p className="recipe-cuisine">Cuisine: {selectedRecipe.cuisine}</p>
  <p className="recipe-ingredients">
   Ingredients: {selectedRecipe.ingredients.join(", ")}
  </p>
  <img
   src={selectedRecipe.image}
   alt={selectedRecipe.name}
   className="recipe-image"
  />
 </div>
)}
```

8. Rendering All Recipes Grid

```
<div className="container">
  <h1 className="title">All Recipe List</h1>
  <div className="grid-container">
    {recipes.map((recipe) => (
      <div key={recipe.id} className="card">
        <img src={recipe.image} alt={recipe.name} className="card-image" />
        <h2 className="card-title">{recipe.name}</h2>
        <p className="card-info">Prep Time: {recipe.prepTimeMinutes} min</p>
        <p className="card-info">Cook Time: {recipe.cookTimeMinutes} min</p>
        <p className="card-ingredients">
          {recipe.ingredients.length} ingredients
        </p>
      </div>
    ))}
  </div>
</div>
```

Conclusion

This React component provides a robust solution for implementing a debounced search bar with auto-suggestions. It efficiently manages API calls, state, and user interactions while maintaining a clean and responsive UI. The use of modern CSS techniques ensures a visually appealing design across different devices.