FINAL HANDS-ON PROJECT

BY FATEEMAH TAJI ABUBAKAR

FEATURES OF THE APP

-Create Notes: Users can create new notes.

- Edit/Delete Notes: Users can modify or remove notes.

- Search Functionality: Users can search for notes.

- Responsive Design: The app works well on both mobile and desktop devic

USER INTERFACE OVERVIEW

The app features a clean, user-friendly interface with easily accessible buttons for creating, editing, and deleting notes. The search bar allows users to quickly find specific notes, while categorization helps keep notes organized

CODE SNIPPETS OVERVIEW

ADDING A NEW NOTE

- const addNote = (note) => {
- setNotes([...notes, note]);
- };
- **Purpose**: This function adds a new note to the current list of notes.
- **How it Works**:
- **Parameter**: It takes a single parameter, `note`, which represents the new note to be added.
- **State Update**: It uses the `setNotes` function to update the `notes` state.
  - The spread operator (`...notes`) creates a new array that includes all the existing notes, followed by the new `note` object.
  - **Importance**: This function is crucial for enabling users to expand their collection of notes. Every time a user creates a new note, this function ensures the updated list is displayed, reflecting the latest changes.

UPDATING AN EXISTING NOTE

- const updateNote = (updatedNote) => {
- setNotes(notes.map(note => note.id === updatedNote.id ? updatedNote : note));
- };
- **Purpose**: This function updates an existing note in the notes collection.
- **How it Works**:
- **Parameter**: It accepts `updatedNote`, which is the modified note object containing the updated title and content.
- **State Update**: The `setNotes` function updates the state by mapping over the current `notes` array. For each note:
  - If the `id` of the note matches the `id` of `updatedNote`, it returns the `updatedNote`.
  - Otherwise, it returns the existing note.
  - **Importance**: This function allows users to edit notes seamlessly. By maintaining the integrity of the notes array and only updating the relevant note, it ensures that all other notes remain unchanged.

DELETING A NOTE

- const deleteNote = (id) => {
- setNotes(notes.filter(note => note.id !== id));
- };
- **Purpose**: This function removes a note from the collection based on its unique identifier.
- **How it Works**:
- **Parameter**: It takes `id`, which is the identifier of the note to be deleted.
- **State Update**: The `setNotes` function is called with a new array created by filtering the `notes` array, returning only those notes whose `id` does not match the provided `id`.
- **Importance**: This function is essential for managing notes efficiently, allowing users to remove notes they no longer need. It ensures that the user interface updates dynamically to reflect the current list of notes.

RENDERING LIST OF NOTES

```
- const NoteList = ({ notes, deleteNote, setSelectedNote }) => {
-     return (
-         <div className="note-list">
-             {notes.map(note => (
-                 <Note
-                     key={note.id}
-                     note={note}
-                     deleteNote={deleteNote}
-                     setSelectedNote={setSelectedNote}
-                 />
-             ))}
-         </div>
-     );
- };
```

SELECTING A NOTE FOR EDITING

- const handleEdit = () => {
- setSelectedNote(note);
- };
- **Purpose**: This function sets the currently selected note for editing.
- **How it Works**:
  - **No Parameters**: It does not take any parameters, as it uses the note from the surrounding scope.
  - **State Update**: It updates the `selectedNote` state in the parent `App` component with the note that the user wants to edit.
- **Importance**: This function allows users to easily access the note they wish to modify. By setting the selected note, the `NoteEditor` component can then pre-fill the input fields, improving the user experience.

HANDLING FORM SUBMISSION

```
- const handleSubmit = (e) => {
-     e.preventDefault();
-     const note = { id: Date.now(), title, content };
-     if (selectedNote) {
-         updateNote({ ...note, id: selectedNote.id });
-     } else {
-         addNote(note);
-     }
-     setTitle('');
-     setContent('');
- };
```

SUMMARY

- These key code snippets form the backbone of the React Notes App, enabling essential functionalities like adding, updating, and deleting notes. Each function is designed to manage the state effectively and ensure the user interface reflects the current state of notes, resulting in a smooth and responsive user experience.