TaskList Application Updates

Recent Improvements

1. Theme Persistence with localStorage

The application now saves the selected theme in localStorage, ensuring the theme persists across page refreshes.

Implementation in Header.jsx:

```
// Initialize theme from localStorage or use default
const [theme, setTheme] = useState(
    localStorage.getItem("theme") || "gTwo"
);

// Save theme to localStorage whenever it changes
useEffect(() => {
    document.documentElement.className = theme;
    localStorage.setItem("theme", theme);
}, [theme]);
```

Why this change?

- Provides better user experience by remembering theme preference
- Maintains consistency across sessions
- Uses localStorage for persistent storage
- Default theme (gTwo) is used if no theme is stored

2. Empty Task Prevention

Added validation to prevent adding empty tasks to the list.

Implementation in AddTask.jsx:

```
function handleSubmit(e) {
    e.preventDefault();

    const taskValue = e.target.task.value.trim();
    if (!taskValue) {
        alert('Please enter a task');
        return;
    }
    // ... rest of the code
}
```

Why this change?

- Improves data quality by preventing empty tasks
- Provides user feedback through alert message
- Uses trim() to remove whitespace-only tasks
- Maintains clean task list

3. Improved Edit Mode Experience

Enhanced the editing experience with automatic focus and cursor positioning.

Implementation in AddTask.jsx:

```
const inputRef = useRef(null);

useEffect(() => {
    if (task.id && inputRef.current) {
        inputRef.current.focus();
        // Set cursor position to the end of the text
        const length = inputRef.current.value.length;
        inputRef.current.setSelectionRange(length, length);
    }
}, [task.id]);

// In the input element
<input
    ref={inputRef}
    // ... other props
/>
```

Why this change?

- Better user experience when editing tasks
- Automatically focuses input when entering edit mode
- Places cursor at the end of existing text
- Makes editing more efficient and intuitive

Technical Details

Theme Storage

- Uses localStorage API for persistent storage
- Theme state is managed with useState hook
- useEffect hook handles theme persistence
- Default theme is "gTwo" if no theme is stored

Task Validation

- Implements client-side validation
- Uses trim() to handle whitespace
- Provides immediate user feedback
- Prevents unnecessary state updates

Edit Mode Improvements

- Uses useRef hook for input element reference
- useEffect hook for automatic focus and cursor positioning
- setSelectionRange for precise cursor placement
- Improved input handling and state management

Benefits

1. **Theme Persistence**
   - Consistent user experience
   - Personalization across sessions
   - No theme reset on refresh
2. **Task Validation**
   - Cleaner task list
   - Better data quality
   - Improved user feedback
3. **Edit Mode**
   - Smoother editing experience
   - Automatic input focus
   - Cursor positioned at text end

  - More efficient task updates