Noet App - Component Reference Card

🏗️ Architecture Overview

Total Lines: ~2000+ lines
Components: 5 modular components + 1 main app component
State Management: Local state with prop drilling
Styling: Tailwind CSS classes

📦 Component Breakdown

1. AuthenticationFlow (~150 lines)

Purpose: Handles user login and 2FA authentication
Location: Lines 1-150 (approx)

Key Props:

typescript
```
{
  users: User[],
  setUsers: Function,
  onLoginSuccess: (user) => void,
  onSecurityLog: (event, details) => void
}
```

Key Functions:

- `handleLogin()` - Validates credentials
- `completeLogin()` - Finalizes authentication
- OTP verification logic

States:

- `authStep`: 'login' | 'otp'

- `email`, `password`, `otp`
- `loginError`

---

2. Sidebar (~400 lines)

Purpose: Left navigation panel with folders, notebooks, tags
Location: Lines 150-550 (approx)

Key Props:

typescript
```
{
  user, currentView, selectedNotebook, notes, notebooks,
  folders, tags, expandedFolders, leftPanelWidth,
  draggedNote, draggedNotebook, draggedTag, dragOverNotebook,
  onCreateNewNote, onDeleteFolder, onDeleteNotebook, onDeleteTag,
  onNotebookDrop, onFolderDrop, onTagDrop, // drag handlers
  onShowFolderModal, onShowNotebookModal, onShowTagModal // modal triggers
}
```

Key Functions:

- `getNotebookCount()` - Count notes in notebook
- `getTagCount()` - Count notes with tag
- `toggleFolderExpanded()` - Expand/collapse folders
- `getOrganizedNotebooks()` - Structure notebooks by folder

Features:

- Drag & drop for notebooks and tags
- Nested folder structure
- User menu dropdown
- Visual drop indicators

---

3. NotesList (~250 lines)

Purpose: Middle panel displaying filtered/sorted notes
Location: Lines 550-800 (approx)

Key Props:

typescript
{
  currentView, selectedNotebook, notes, trashedNotes,
  selectedNote, setSelectedNote, searchTerm, setSearchTerm,
  filterBy, setFilterBy, sortBy, setSortBy, tags,
  onToggleNoteShortcut, onDeleteNote, onNoteDragStart,
  middlePanelWidth

}

Key Functions:

- `getCurrentItems()` - Filter/sort notes
- `getCurrentViewTitle()` - Display title

Features:

- Search functionality
- Filter by tags
- Sort by title/date
- Drag notes to notebooks
- Empty state handling

---

4. NoteEditor (~600 lines)

Purpose: Rich text editor with formatting and version history
Location: Lines 800-1400 (approx)

Key Props:

typescript
{
  selectedNote, setSelectedNote, notes, setNotes,
  tags, onAddTag, onRemoveTag, onShowTagModal,
  user, setUsers, onCreateNewNote

}

Key Functions:

- `execCommand()` - Text formatting (bold, italic, etc.)
- `saveCursorPosition()` - Preserve cursor location
- `restoreCursorPosition()` - Restore cursor after format

- `updateNoteContent()` - Track changes
- `saveNote()` - Auto-save functionality
- `manualSave()` - Force save with version
- `addVersionToNote()` - Version management
- `viewVersion()` - View historical versions
- `restoreVersion()` - Restore old version

Features:

- Rich text formatting toolbar
- Version history panel
- Auto-save with debouncing
- Tag management
- Cursor preservation fix

States:

- `noteTitle`, `noteContent`
- `isEditing`
- `showVersionHistory`
- `viewingVersion`
- Toolbar dropdowns (color, font size, etc.)

---

5. ModalsCollection (~400 lines)

Purpose: All modal dialogs in one component
Location: Lines 1400-1800 (approx)

Modals Included:

1. **Folder Management**
   - Create Folder Modal
   - Rename Folder Modal
2. **Notebook Management**
   - Create Notebook Modal
   - Rename Notebook Modal
3. **Tag Management**
   - Create Tag Modal
4. **User Management**
   - Change Password Modal
   - User Settings Modal
   - User Admin Modal

    - Create User Modal
    - Delete User Confirmation
5. **Security**
    - OTP Setup Modal
    - Backup Codes Modal

Props Pattern: Each modal needs:

- `show[ModalName]` - Visibility state
- `setShow[ModalName]` - Toggle function
- Form values and setters
- Action handlers

---

6. NoetApp (Main) (~400 lines)

Purpose: Main app component that orchestrates everything
Location: Lines 1800-2200 (approx)

Key Responsibilities:

- User state management
- Data persistence to users array
- Panel resizing logic
- Drag & drop coordination
- CRUD operations for all entities
- Session timeout management

Major State Groups:

typescript
```
// Authentication
user, users, isAuthenticated

// UI State
currentView, selectedNotebook, selectedNote

// User Data
notes, notebooks, folders, tags, trashedNotes

// Drag State
draggedNote, draggedNotebook, draggedTag

// Layout
```

leftPanelWidth, middlePanelWidth

Key Functions:

- `createNewNote()`, `deleteNote()`, `toggleNoteShortcut()`
- `createFolder()`, `deleteFolder()`, `renameFolder()`
- `createNotebook()`, `deleteNotebook()`, `renameNotebook()`
- `createTag()`, `deleteTag()`, `addTagToNote()`
- `handleNoteDragStart()`, `handleNotebookDrop()`, etc.
- `logSecurityEvent()`, `changePassword()`
- `exportUserData()`, `setupOtp()`

🔧 Common Issues & Solutions

Editor Cursor Jumping

Component: NoteEditor
Functions: `saveCursorPosition()`, `restoreCursorPosition()`
Fix: Save range before command, restore after with setTimeout

Drag & Drop Not Working

Components: Sidebar, NotesList
Check: All handlers connected (onDragStart, onDrop, onDragOver, etc.)
Key State: `draggedNote`, `draggedNotebook`, `dragOverNotebook`

Delete Functions

Component: NoetApp (main)
Functions: `deleteNote()`, `deleteFolder()`, `deleteNotebook()`, `deleteTag()`
Pattern: Confirm → Update state → Update user data

Version History

Component: NoteEditor
Functions: `addVersionToNote()`, `shouldCreateVersion()`
Logic: Auto-save every 2min or on significant changes

💬 Efficient Chat Strategies

Starting a New Chat:

"I have a modular Noet app with [Component Name].
Issue: [Specific problem]
The component has these key functions: [List relevant functions]

Here's the problematic code section: [Paste only relevant part]"

Referencing Fixes:

"Previously fixed cursor jumping with saveCursorPosition() in NoteEditor"
"Drag-drop uses dragOverNotebook state for blue indicators"

"Delete functions follow pattern: confirm → update → persist"

Component Interface Only:

"My Sidebar component receives props: {notes, notebooks, onDeleteFolder}

and calls onDeleteFolder(folderId) but it's not working"

📊 Size Guidelines

Share This	Size	Risk of Max Length
Full app	2000+ lines	⚠️ HIGH
Single component	200-600 lines	✅ LOW
Component interface	10-30 lines	✅ NONE
Specific function	20-50 lines	✅ NONE
Problem description + code	100-200 lines	✅ NONE

🚀 Quick Reference

To discuss editor issues: Share NoteEditor component
To discuss navigation: Share Sidebar component
To discuss auth: Share AuthenticationFlow component
To add modals: Share ModalsCollection component
To discuss state management: Reference NoetApp main functions

Always include:

1. Component name
2. Relevant props
3. Specific function/area
4. What you've tried
5. Expected vs actual behavior