**Task: Real-Time Collaborative Note-Taking Application**

**Objective:**

Develop a real-time, collaborative Note-Taking Application that allows multiple users

to create, edit, and share notes in a highly interactive and responsive web interface.

The application should showcase advanced React.js and Next.js concepts, responsive

CSS, and complex functionality.

**Requirements:**

**1. User Authentication:**

* Implement a user authentication system with login and signup using a mock API or Firebase.
* Allow users to reset passwords via email or a mock reset process.

**2. Dashboard:**

* After logging in, users are redirected to a dashboard that displays:
  + Recent notes with search and filter options.
  + Categorized folders for organizing notes.
  + Real-time updates when a new note is created or updated (e.g. WebSockets).

**3. Note Editor:**

* Create a rich text editor for writing and formatting notes (bold, italics, bullet points, code blocks, etc.).
* Allow users to:
  + Add inline tags (e.g., #todo , #important ) for easy
  + categorization.
  + Attach images by dragging and dropping or uploading.
* Implement autosave functionality to save drafts locally and sync changes to the server.
* Show real-time word and character count.

**4. Collaboration:**

* Add functionality for real-time collaboration:
  + Allow users to share notes with others via email or a unique link.
  + Multiple users can edit the same note simultaneously with real-
  + time updates (e.g., using WebSocket or libraries like Socket.IO).
  + Display active collaborators and their cursors in the editor.

**5. Sharing and Exporting:**

* Enable sharing options:
  + Public: Generate a read-only link.
  + Private: Share with specific users who can view or edit.
* Export notes to PDF, Markdown, or plain text.

**6. Responsive Design:**

* Ensure pixel-perfect responsiveness across desktop, tablet, and mobile devices.
* Use CSS Grid or Flexbox for layout and advanced CSS techniques for animations (e.g., smooth transitions, collapsible sidebars).

**7. Additional Features:**

* Dark Mode: Implement a toggle for light and dark themes with smooth transitions.
* Search: Build a powerful search bar with support for full-text search and tag-based filtering.
* Notifications: Add notifications for shared note updates or collaboration requests using a library like React Hot Toast.

**8. Admin Panel (Optional):**

* Create an admin panel where admin users can:
  + Monitor active users and notes.
  + Delete inappropriate content.

**9. Testing:**

* Write unit tests for key components using Jest and React Testing Library.
* Add end-to-end tests for critical user flows like note creation, sharing, and collaboration using Cypress or Playwright.

**Technical Constraints:**

* Use React.js and Next.js for the entire application.
* Use State Management tools like React Context, Zustand, or Redux for managing the note state and collaboration data.
* Avoid third-party libraries for real-time functionality (e.g., WebSocket or custom implementation preferred over services like Pusher).

**Deliverables:**

1. A GitHub repository with:

* Clean, modular, and reusable code.
* Well-written documentation in a README.md file with setup instructions and feature explanations.

2. Deployed application on Vercel or Netlify.

3. A short video or documentation explaining:

* How the collaborative feature works.
* How performance optimizations were handled.

**Evaluation Criteria:**

**1. Functionality:** Completeness of the required features.

**2. Code Quality:** Modularity, maintainability, and use of best practices.

**3. Design and Responsiveness:** Adherence to pixel-perfect design and responsiveness.

**4. Performance:** Optimized handling of real-time updates and heavy data.

**5. Innovation:** Implementation of advanced features like collaboration.

**6. Testing:** Coverage and quality of tests.

**7. User Experience:** Usability and intuitiveness of the app.