Detailed Prompt for `Projects/Show.vue` Update & Client Approval Workflow

Please act as an expert Vue.js developer with a strong understanding of Firebase Firestore integration and backend API communication. Your task is to provide the code modifications for `Projects/Show.vue` and describe the overall workflow, including the backend magic link generation and the client's experience on the approval dashboard.

Part 1: Vue.js Component (`Projects/Show.vue`) Update

Goal: Integrate a feature into `Projects/Show.vue` that allows project managers to add content (blogs, social media posts) for client approval. This content will be stored in Firestore, and a magic link will be generated and sent to the client.

Component to Update: `Projects/Show.vue`

New UI Section Requirements:

1. **Placement:** Add a clearly visible section (e.g., a card or panel) within the `Projects/Show.vue` template. This section should be titled "**Add Content for Client Approval**".
2. **Input Fields:**
    - **Document Title:** A text input field (`<input type="text">`) for the user to enter a descriptive title for the content (e.g., "August Blog Post - SEO Tips"). This should be **required**.
    - **Google Drive File ID or Shareable Link:** A text input field (`<input type="text">`) where the user can paste either the raw **Google Drive file ID** (e.g., `1s_lYq-2A5b9c0d1e2f3g4h5i6j7k8l9`) or the **full shareable link** (e.g., `https://docs.google.com/document/d/1s_lYq-2A5b9c0d1e2f3g4h5i6j7k8l9/edit?usp=sharing`).
        - The Vue.js code should include logic to **extract the Google Drive file ID** from a full shareable URL if a URL is provided. If only the ID is provided, it should use that directly. This field should also be **required**.
        - Add a small helper text below this input explaining how to find the Google Drive File ID from a shareable link.
3. **Action Button:**
    - A primary button titled "**Submit for Client Approval**".

- This button should trigger the logic to save the document to Firestore and initiate the magic link email sending.

4. **Loading/Message Display:** Implement a small area to display loading indicators or success/error messages after submission.

Vue.js Logic (`<script setup>` or `data/methods`):

1. **Firebase Firestore Integration:**

   - Assume `firebaseConfig`, `appId`, `db`, and `auth` (and the currently logged-in user's `uid`) are accessible, similar to how they are used in the React app.
   - When the "Submit for Client Approval" button is clicked:
     - **Validate Inputs:** Ensure both the title and Google Drive ID/URL are provided.
     - **Extract Google Drive ID:** Implement a helper function to safely extract the Google Drive file ID from the input string (handling both direct IDs and full URLs).
     - **Add Document to Firestore:** Add a new document to the Firestore collection: `artifacts/{appId}/public/data/clientPortalDocuments` The document structure should be: JavaScript

```
{
    title: newDocTitle,              // From input
    googleDriveId: extractedId,     // Extracted from input
    status: 'Pending Review',       // Initial status
    magicTokenGroup: YOUR_PROJECT_OR_CLIENT_ID_HERE, // This links it to a specific client dashboard. This should be derived from the current project's ID in Projects/Show.vue.
    createdAt: serverTimestamp(),
    lastUpdatedAt: serverTimestamp(),
}
```

     - **Crucial:** The `magicTokenGroup` field is critical. It should be dynamically set based on the current project's ID (or a unique client

ID associated with this project) that `Projects/Show.vue` already manages.

- **Call Backend Endpoint for Magic Link & Email:** After successfully adding the document to Firestore, the Vue component should make an **API call to your backend** to trigger the magic link generation and email sending.
    - This API call should pass at least the `magicTokenGroup` (project/client ID) and ideally the `title` of the newly added document.
    - Example API call (conceptual):
      JavaScript

```
// Assuming you have an Axios or fetch equivalent setup in Vue
await axios.post('/api/send-approval-magic-link', {
    projectId: YOUR_PROJECT_OR_CLIENT_ID_HERE, // This is your magicTokenGroup
    documentTitle: newDocTitle,
});
```

2. **Clear Form:** After successful submission, clear the input fields and display a success message.
3. **Error Handling:** Implement `try-catch` blocks for Firestore operations and API calls, displaying user-friendly error messages.

Part 2: Backend Logic for Magic Link Generation & Email (Conceptual)

Note: This is *not* Vue.js code, but explains the server-side process that your Vue app will interact with.

When the Vue.js component makes the API call to `/api/send-approval-magic-link`:

1. **Generate Unique Magic Token:** The backend service should generate a **secure, unique, and time-limited magic token** (e.g., a UUID, or a JWT with a short expiry).
2. **Store Token:** This token should be stored in a secure database (not Firestore directly for client-side access) along with the associated `projectId` (which corresponds to your

`magicTokenGroup`) and its expiry date. This allows the React approval portal to validate the token.

3. **Construct Magic Link:** Create the full URL for the client's approval dashboard, incorporating the generated token: `https://your-client-approval-portal-domain.com/?token=GENERATED_MAGIC_TOKEN` (Replace `your-client-approval-portal-domain.com` with the actual URL where your React app is deployed).
4. **Send Email to Client:**
    - Use your preferred email sending service (e.g., SendGrid, Mailgun, or your own SMTP setup).
    - The email should be clear, professional, and contain:
        - A subject line like: "New Content Awaiting Your Approval from [Your Company Name]"
        - A friendly greeting.
        - A clear call to action: "We have new content ready for your review and feedback. Please click the link below to view the dashboard:"
        - The **magic link** (the full URL constructed above).
        - Brief instructions: "On the dashboard, you can view each item, leave comments, or approve them individually or in bulk."
        - Information about the specific new document, if applicable.
5. **Google Chat Message (Optional/Complementary):** If desired, in addition to the email, use your `GoogleChatService > send message` to send an internal notification to your team's Google Chat space, informing them that content has been sent for client approval (e.g., "Content for Project X sent to client for approval. New blog post 'Q3 SEO Trends' included.").

Part 3: Client Experience on the Approval Dashboard (React App)

When the client clicks the magic link from the email, they will be directed to the React application you already have. Here's what they can do:

1. **Dashboard View:**
    - **No Login Required:** The client immediately sees the dashboard without needing to log in or create an account.

- **Content List:** They'll see a clear list of all documents (blog posts, social media posts) currently awaiting their attention for *that specific project/client* (identified by the `magicToken` in the URL). Each item will show its title and current status (e.g., "Pending Review," "Approved").
- **Bulk Approval:** If there are multiple items pending, they can:
  - Select multiple documents using **checkboxes**.
  - Click a "**Approve Selected**" button to quickly approve all chosen items at once.

2. **Individual Document Review:**
   - By clicking on any document in the list, they are taken to a detailed review screen.
   - **Embedded Content:** On this screen, they will see the actual Google Doc (or other Drive file) embedded directly, allowing them to read and review it without leaving the portal.
   - **Leave Comments:** Below the embedded content, there is a dedicated section where they can:
     - Optionally provide their name.
     - Type in detailed comments or feedback for that specific document.
     - Submit their comments, which will be saved and visible to you (and other clients on the same group if configured).
   - **Approve Individually:** They can also click an "**Approve Document**" button on this screen to approve that single item.
   - **Real-time Updates:** Any comments or approvals they make will update in real-time for anyone else viewing the same document.
   - **Navigation:** A "Back to Dashboard" button allows them to easily return to the list of documents.