Scratchpad

Link-Folder: Auto-create new folders with links, or point to existing?

When a link is created, a corresponding folder is automatically generated for it. Users can also generate links for folders they have already created (not only those generated by links).

Folders can move freely between **personal context** and **link context**:

- **Generate a link from a personal folder** → links it to external sharing.
- **Create a link directly** → automatically generates a new folder tied to that link.
- **Decouple a folder from its link** → the link becomes inactive, while the folder remains. To reactivate, the user must enable the link again, which re-generates a new folder for it.
- **Delete a folder** → prompts the user to choose whether to:
  1. Delete both the folder and its link, or
  2. Only decouple and delete the folder, keeping the link (inactive).

Issue 1: What happens to external uploads inside folder when decoupled from link?

They remain in the folder

Issue 2: What happens to the folders when they are coupled/decoupled with/from a link?

Folder transitions from personal ⇛ shared context and vice versa

Issue 3: How should links be categorized?

Option	Description	Pros	Cons	Your Position
A. Base/Custom/Generated (V1)	Old schema	Granular control	Too complex	❌ Overkill
B. Dedicated + Public (V2)	Only two types: dedicated (email-bound) and public (anyone)	Simple, intuitive	May miss edge cases	❓ Needs more debate
Hybrid w/ Auto-first-link	Account creation auto-generates a first link (like	Great onboarding	Needs rules for editing/deleting	⚖️ Worth deeper discussion

Option	Description	Pros	Cons	Your Position
	Calendly), user can then add/edit more			

File Ownership: External uploads count toward user's quota, or separate bucket?

External uploads count towards user’s quota.

Issue 1: Can external users create subfolders?

They can, based on permissions defined by the link owner.

Option	Description	Pros	Cons	Your Position
A. Root-only	Uploaders can only drop files in link root	Simpler, no folder conflicts	No organization	❌ Too limiting
B. Allow subfolders (like Google Drive)	Uploaders can create folders/subfolders freely	Full flexibility, mirrors Drive	Potential ownership/deletion issues	✅ Preferred — but ownership rules needed

Action	Owner	Collaborator (verified)	Contributor (unverified)
View files	✅	✅	✅
Upload files	✅	✅	✅
Create folders	✅	✅	✅
Delete own files	✅	✅	✅
Delete others' files	✅	✅	❌
Delete own folders	✅	✅	✅
Delete others' folders	✅	✅	❌
Rename/move files	✅	✅	❌
Manage permissions	✅	❌	❌
Delete link	✅	❌	❌

Role 1: OWNER (Link Creator)
├── Full control over everything
├── Can delete link
├── Can manage permissions
└── Always has highest privileges

Role 2: COLLABORATOR (Verified Email Required)
├── All Contributor permissions
├── Can delete ANY files/folders in link
├── Can rename/move content
├── Requires email verification via code
└── Owner must grant this role explicitly

Role 3: CONTRIBUTOR (Unverified Email)
├── Can view files in link
├── Can upload files
├── Can create folders
├── Can delete own files/folders only
├── Cannot delete others' content
└── Default for new uploaders

Alternative Role Names (Pick what sounds better):

Option A	Option B	Option C	Your Vote 👍👎
Contributor	Uploader	Guest	
Collaborator	Editor	Member	
Owner	Admin	Owner	

Notes:

1. **Email Verification via OTP** – If the link owner grants an uploader **collaborator permissions**, the uploader must verify their email address using a one-time password/code (OTP) before beginning an upload session.
2. **Contributor & Public Uploaders** – Uploaders with **contributor permissions**, or new/public uploaders, do not require verification unless their permissions are later upgraded to **collaborator** by the link owner.

Batch (upload session grouping) Purpose: UI feature or just backend tracking?

Backend tracking.

Option	Description	Pros	Cons	Your Position
A. Session-based batches	Each upload event grouped separately	Preserves upload context	Hard to get “all files by John”	❌ Confusing

Option	Description	Pros	Cons	Your Position
B. Email repositories	Permanent grouping by uploader email	Easy to query all John's files	Extra structure, heavier model	❌ Too heavy
C. Hybrid (tags + optional batch_id)	Always track uploader_email, batch_id optional	Flexible, allows both views	Slightly more metadata	✅ Preferred — email is main key, batch is secondary

---

Email Repositories: Real folders or filtered views?

Filtered views.

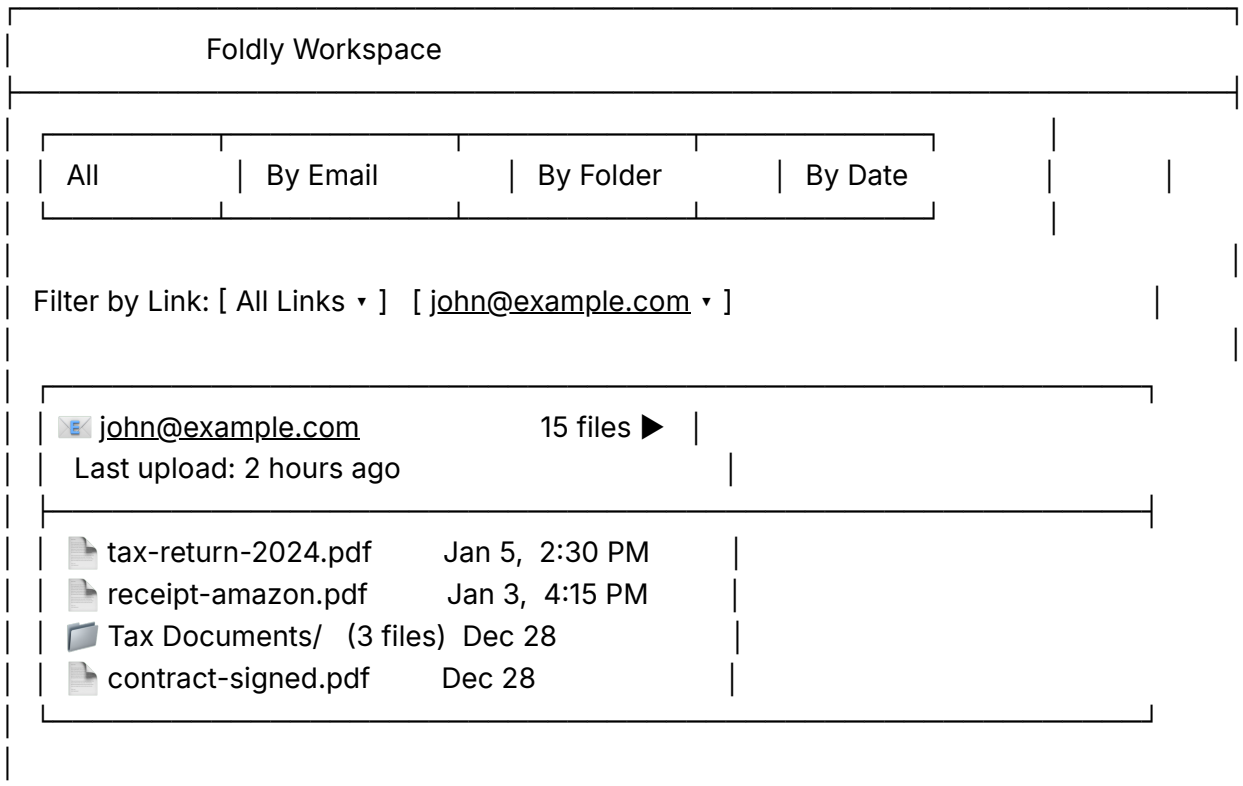
Issue 1: Should email grouping show:

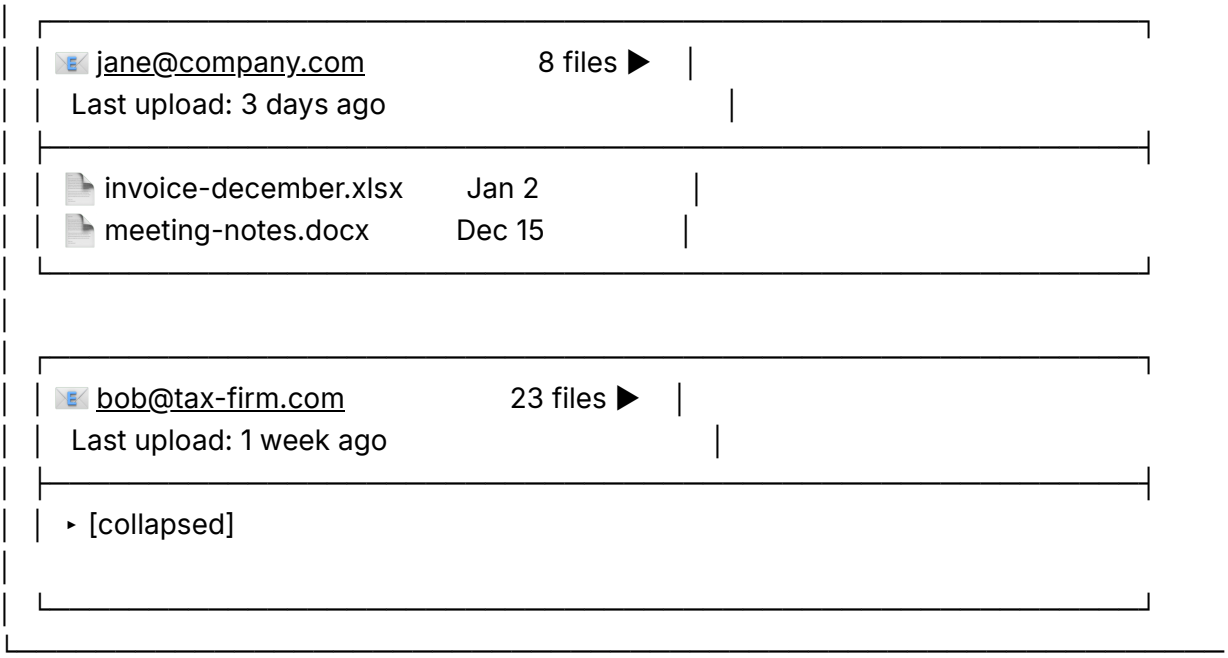
a. All files from that email across ALL links

b. Only files in currently selected link

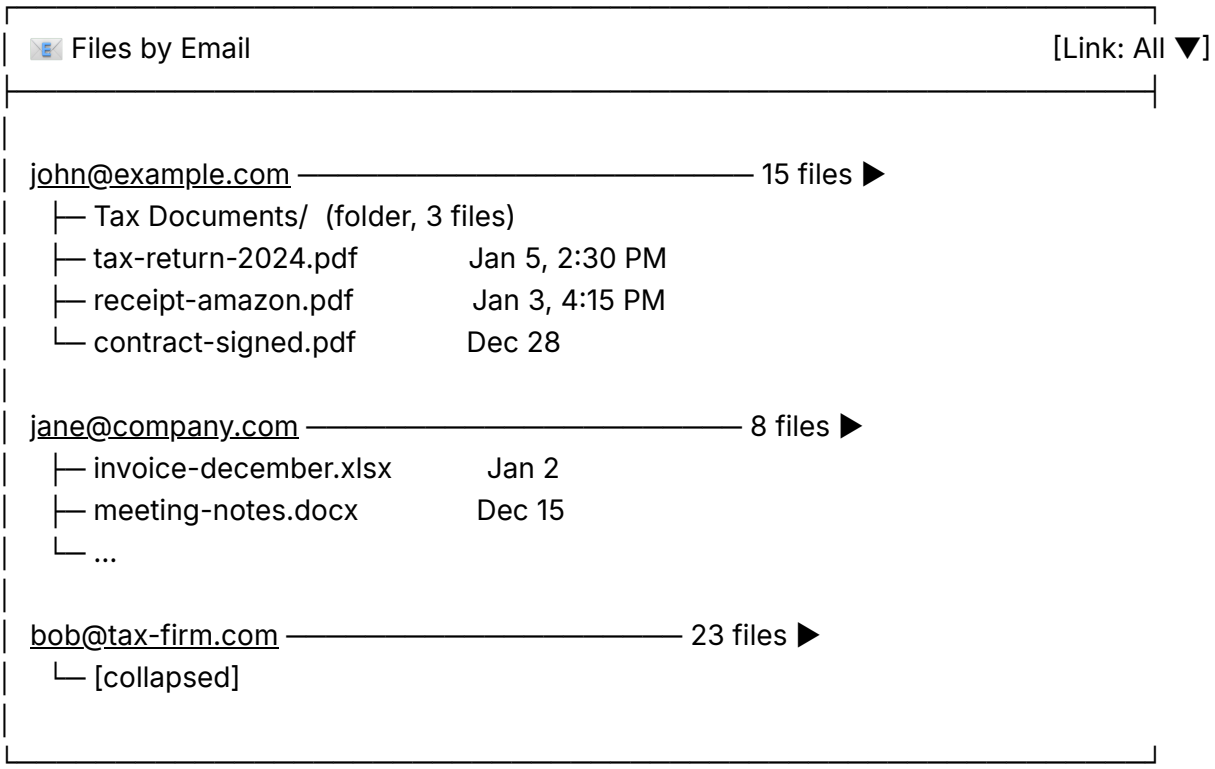
c. Separate view: "Files by Email" filter

The answer is to show each option as a filter and allow the user to have all possible views depending on their needs:





or:



Issue 2: When user copies external file to a personal context, does it keep uploader email tag?

Yes. Files and folders tags always remained stored regardless of the context.

🔷 Behavior Reference: Google Drive

- When a user clicks **“Add to My Drive”** in Google Drive:
  - **No physical copy** of the file is made.
  - Google simply **adds a pointer (shortcut/symlink)** to the user’s Drive tree.
  - Storage is still counted only against the **original owner’s quota**.

🔷 Foldly Architecture

1. **Storage Layer (GCS or Supabase)**

- All files physically live inside a single bucket.
- Partitioning is done by **owner ID** and **link ID**.
- Example path:

```
gs://foldly-bucket/{owner_user_id}/{link_id}/{file_id}
```

2. **Database Layer (Logical Hierarchy)**

- The database is the **source of truth** for file structure, ownership, and permissions.
- Suggested tables:
  - `users`
  - `links`
  - `folders`
  - `files`
  - `permissions`

This allows you to fully mimic a Google Drive–style tree structure while keeping ownership simple.

3. **Quota Tracking**

- Each file uploaded logs its `file_size` under the **owner’s account**.
- Quotas are tracked at the **DB level** (sum of all file sizes for a given `owner_user_id`).

- GCS does not manage quotas directly for individual app users — Foldly's DB enforces this.

4. File Ownership Rule

- **Uploader = Temporary Contributor**
  - Can upload, but does not own files.
  - Uploaded files do **not** count toward their own quota.
- **Link Owner = Absolute Owner**
  - All files uploaded through their links count toward their quota.
  - Owner can delete files and folders.
- **Deletion Rule**: If the owner deletes a file, it disappears for everyone with access.

🔷 Parallel with Google Drive

Aspect	Google Drive	Foldly
File added to "My Drive"	Pointer/shortcut (no copy)	Not applicable (uploads always belong to link owner)
Quota rules	Can shift between owners (complex)	Always counts toward link owner (simple)
Ownership	Multiple possible	Single owner (link creator)
Source of truth	Google's DB	Foldly DB (`files` , `folders` , `permissions`)

👉 This makes Foldly **simpler than Drive**:

- No cross-quota complexity.
- No multi-owner problems.
- One **absolute owner** per file = easier quota enforcement & permission logic.