**ACROSS ALL PLATFORMS**

**Grabs metadata**

* Filename
  + YYYY - ####
* Date posted
  + YYYY – MM - DD
* Caption
* Hashtags
* Number of
  + Likes
  + Comments
  + Shares
* Repost worthy?
  + When?

12/13/2024

* Manual archive button
* What filter categories?
* Arrange by categories?

Which of these is the most important?

**Stores metadata**

* distinction between reels/shorts & longform
  + based on duration of video
* Boolean posted
* Thumbnail of video
* **Store data locally**

**User has direct access to wherever stored**

* can copy individual hashtags/caption while preserving format
  + **currently copying multiple paragraphs doesn’t maintain format**
* can view data through filters
  + date posted
  + file number
  + hashtags used
  + caption
  + number of views
* displays common hashtags

I want to create a program that displays metadata of my social media accounts across Instagram, TikTok, Youtube, and Facebook. The program should be accessible on either mobile or desktop(any OS).

* **Frontend**: Use **Electron.js** (desktop) or **React Native** (mobile/desktop hybrid).
  + Both are beginner-friendly and support cross-platform development.
* **Backend**: Use **Node.js** to interact with social media APIs.
* **Database**: Start with **SQLite** or **Firestore** for simplicity.
* **API Access**: Use official SDKs/APIs provided by Instagram, TikTok, YouTube, and Facebook.
* **Libraries**:
  + **Axios** for API calls.
  + **Moment.js** for date formatting.
* **Deployment**: Run locally on your machine; package the app for Windows, macOS, or mobile devices.

**Strategies to Handle a Large Backlog**

**1. Batch Processing**

* Split the backlog into smaller chunks and fetch metadata incrementally.
* For example:
  + Fetch 100 videos, then wait until the rate limit resets.
  + Use setTimeout or a scheduling library to manage the process.

**2. Prioritize Recent Data**

* Start with the most recent posts/videos since they are more relevant.
* Historical data can be populated later when API limits reset.

**3. Use Multiple API Keys**

* If allowed, create multiple API keys or tokens under different accounts.
* Distribute the workload across these tokens to increase throughput.

**4. Use Webhooks for Updates**

* Some platforms (like YouTube) provide webhooks to notify you of changes or new uploads. Set up webhooks for real-time updates after the backlog is processed.

**5. Cache Metadata Locally**

* Store data locally after fetching to avoid repeatedly hitting APIs for the same information.