NotebookLM Workflow

https://gemini.google.com/share/6887ddea5472

https://gemini.google.com/share/728b4b0de85c

Inputting handwritten notes into **NotebookLM** has become significantly easier with recent updates to its OCR (Optical Character Recognition) capabilities. You can use these methods to bring your handwritten derivations or diagrams into your research workflow.

---

1. Direct Upload via Mobile App (Easiest)

The NotebookLM mobile app (iOS and Android) is the most streamlined way to digitize physical notes.

- **Open the App:** Select the notebook you want to work in.
- **Add Source:** Tap the plus (+) icon or "Add Source."
- **Use Camera:** Select the **Camera** option to take a photo of your handwritten page.
- **OCR Processing:** NotebookLM will automatically scan the image, convert the handwriting to text, and add it as a new source.

2. Uploading Photos or PDFs from Desktop

If you have already scanned your notes or taken photos with a digital camera:

- **Image Files:** You can upload **.jpg**, **.png**, or **.webp** files directly as sources.
- **PDF Scans:** If you use a scanning app (like Adobe Scan or Microsoft Lens), upload the resulting **PDF**. NotebookLM will parse the text within the PDF.
- **Google Drive:** Save your images or scans to Google Drive and import them into NotebookLM using the "Google Drive" source option.

3. The "Google Docs" Bridge

If you have particularly "messy" handwriting or complex math that the direct upload struggles with, this method provides a cleaner conversion:

1. Upload your photo to **Google Drive**.
2. Right-click the image and select **Open with > Google Docs**. This forces a high-quality OCR conversion.
3. Clean up any formatting errors in the Doc.
4. In **NotebookLM**, add that **Google Doc** as a source.

4. Digital Handwriting (iPad/Tablets)

If you take notes digitally (e.g., in Goodnotes, Notability, or Apple Notes):

- **Export as PDF:** Most digital paper apps have a "Searchable PDF" export option.
- **Import to NotebookLM:** Upload that PDF. Because the text is usually already "recognized" by the tablet app, NotebookLM will read it with near-perfect accuracy.

Tips for Best Results

- **Lighting:** Ensure your physical notes are well-lit with no shadows across the text.
- **Contrast:** Use dark ink on light paper; pencil can sometimes be too faint for the OCR to pick up clearly.
- **Print vs. Cursive:** While the model is quite capable with cursive, neat printing yields the highest accuracy for technical terms (like "Lorentz transformations" or LaTeX commands).

Python Script to Rename Scanned Notes

To help you automate the organization of your scanned notes, we can use a Python script that leverages the os and datetime modules. This is particularly useful if you are scanning multiple pages of derivations or lecture notes and want to avoid the generic "IMG_1234.jpg" naming convention.

The following script will scan a folder for image files and rename them based on a **prefix** (like "Physics_Notes") and the **date they were created**.

Python Script for Batch Renaming

```
import os
from datetime import datetime

def rename_scans(directory_path, prefix):
    # Supported file extensions
    extensions = ('.jpg', '.jpeg', '.png', '.pdf')

    # Change to the target directory
    try:
        os.chdir(directory_path)
    except FileNotFoundError:
        print("Directory not found.")
        return

    files = [f for f in os.listdir() if f.lower().endswith(extensions)]
```

```
    # Sort files by creation time to keep notes in order
    files.sort(key=os.path.getctime)

    for index, filename in enumerate(files, start=1):
        extension = os.path.splitext(filename)[1]

        # Get creation date
        creation_time = os.path.getctime(filename)
        date_str = datetime.fromtimestamp(creation_time).strftime('%Y-%m-%d')

        # New name format: Prefix_Date_Index.extension
        new_name = f"{prefix}_{date_str}_{index:03d}{extension}"

        print(f"Renaming {filename} to {new_name}")
        os.rename(filename, new_name)

# Usage
folder = "C:/Users/YourName/Documents/ScannedNotes" # Update this path
rename_scans(folder, "Electromagnetism_Notes")
```

How this fits your NotebookLM workflow

- **Sequential Order:** Sorting by creation time ensures that Page 1 of your notes remains "001" even if the scanner software assigned random numbers.
- **Searchability:** When you upload these to NotebookLM, the source titles will be clear and descriptive, making it easier for the AI to cite specific pages in its "Source Guide."
- **Snake Case:** Following your previous interest in string formatting, this script uses **snake_case** for filenames, which is a standard practice for ensuring compatibility across different operating systems.

Adding a Popup File-Explorer Window

To add a "Pop-up" window, we will use **Tkinter**, which is Python's built-in library for creating graphical interfaces. This allows you to select your folder using a standard file explorer window rather than typing paths.

The GUI Version of the Script

```
import os
import tkinter as tk
from tkinter import filedialog, messagebox
```

```
from datetime import datetime

def rename_scans():
    # 1. Open the Folder Selection Window
    root = tk.Tk()
    root.withdraw()  # Hide the main small tk window
    directory_path = filedialog.askdirectory(title="Select Folder Containing Scanned Notes")

    if not directory_path:
        return

    prefix = "Physics_Notes" # You could also use simpledialog to ask for this
    extensions = ('.jpg', '.jpeg', '.png', '.pdf')

    try:
        files = [f for f in os.listdir(directory_path) if f.lower().endswith(extensions)]
        files.sort(key=lambda x: os.path.getctime(os.path.join(directory_path, x)))

        for index, filename in enumerate(files, start=1):
            full_path = os.path.join(directory_path, filename)
            extension = os.path.splitext(filename)[1]
            creation_time = os.path.getctime(full_path)
            date_str = datetime.fromtimestamp(creation_time).strftime('%Y-%m-%d')

            new_name = f"{prefix}_{date_str}_{index:03d}{extension}"
            new_path = os.path.join(directory_path, new_name)

            os.rename(full_path, new_path)

        messagebox.showinfo("Success", f"Renamed {len(files)} files successfully!")
    except Exception as e:
        messagebox.showerror("Error", f"An error occurred: {e}")

if __name__ == "__main__":
    rename_scans()
```

Finally, let's convert the script into an executable

Creating an Executable (OS Specific)

First, ensure PyInstaller is installed: pip install pyinstaller. Use the following commands based on your operating system:

🪟 Windows

- Open **PowerShell** or **Command Prompt**.
- Navigate to your script folder.
- Run:
- `pyinstaller --onefile --noconsole rename_notes.py`
    - **Note:** The --noconsole (or -w) flag is key here. Since we added a GUI, this prevents a black terminal box from appearing behind your pop-up window.
    - **Result:** Find rename_notes.exe in the dist folder.

🍎 macOS

5. Open **Terminal**.
6. Run:
7. `pyinstaller --onefile --windowed rename_notes.py`
    - **Note:** On Mac, --windowed creates a .app bundle. This makes it feel like a native Mac application that you can move to your Applications folder.
    - **Result:** Find rename_notes.app in the dist folder.

🐧 Linux

- Open your **Terminal**.
- Run:
  `pyinstaller --onefile rename_notes.py`
    - **Note:** Linux binaries don't typically use a "no console" flag in the same way, but the script will launch the Tkinter window perfectly. Ensure you have python3-tk installed on your system (sudo apt install python3-tk).
    - **Result:** Find the executable (no extension) in the dist folder.

Adding a Popup File-Explorer Window

To add a "Pop-up" window, we will use **Tkinter**, which is Python's built-in library for creating graphical interfaces. This allows you to select your folder using a standard file explorer window rather than typing paths.

The GUI Version of the Script

```
import os
import tkinter as tk
from tkinter import filedialog, messagebox
from datetime import datetime

def rename_scans():
    # 1. Open the Folder Selection Window
    root = tk.Tk()
    root.withdraw()  # Hide the main small tk window
    directory_path = filedialog.askdirectory(title="Select Folder Containing Scanned Notes")

    if not directory_path:
        return

    prefix = "Physics_Notes" # You could also use simpledialog to ask for this
    extensions = ('.jpg', '.jpeg', '.png', '.pdf')

    try:
        files = [f for f in os.listdir(directory_path) if f.lower().endswith(extensions)]
        files.sort(key=lambda x: os.path.getctime(os.path.join(directory_path, x)))

        for index, filename in enumerate(files, start=1):
            full_path = os.path.join(directory_path, filename)
            extension = os.path.splitext(filename)[1]
            creation_time = os.path.getctime(full_path)
            date_str = datetime.fromtimestamp(creation_time).strftime('%Y-%m-%d')

            new_name = f"{prefix}_{date_str}_{index:03d}{extension}"
            new_path = os.path.join(directory_path, new_name)

            os.rename(full_path, new_path)

        messagebox.showinfo("Success", f"Renamed {len(files)} files successfully!")
    except Exception as e:
        messagebox.showerror("Error", f"An error occurred: {e}")

if __name__ == "__main__":
    rename_scans()
```

Adding an Optional Professional Flair

If you want to give your app a "branded" look, you can add an icon by adding

`--icon=atom.ico` (Windows) or

`--icon=atom.icns` (Mac)

to the PyInstaller command. Linux handles icons differently than Windows and macOS. Unlike those systems, where the icon is "baked" into the file itself, Linux usually relies on **.desktop files** to associate an image with an executable. When you run PyInstaller on Linux, the `--icon flag` technically exists, but it doesn't do much for the binary file. To get a custom icon to show up in your app launcher or on your desktop, you have to create a small text file.

Steps to add an icon on Linux

1. Generate the Executable

Run your PyInstaller command as usual:

```
pyinstaller --onefile rename_notes.py
```

This gives you the binary file in the dist folder.

2. Create a .desktop File

Open a text editor and paste the following, replacing the paths with your actual file locations:

```
[Desktop Entry]
Version=1.0
Type=Application
Name=Notes Renamer
Exec=/path/to/your/dist/rename_notes
Icon=/path/to/your/icon_image.png
Terminal=false
Categories=Utility;
```

3. Make it Official

- Save this file as notes_renamer.desktop.
- Right-click the file, go to **Properties > Permissions**, and check **"Allow executing file as**

program."

- (Optional) Move it to ~/.local/share/applications/ if you want it to show up in your system's App Menu.

Summary of Commands per OS

OS	PyInstaller Command	Icon File Format
Windows	pyinstaller --onefile --noconsole --icon=icon.ico rename_notes.py	.ico
macOS	pyinstaller --onefile --windowed --icon=icon.icns rename_notes.py	.icns
Linux	pyinstaller --onefile rename_notes.py (+ .desktop file)	.png or .svg

A Quick Note on Icon Formats

- **Windows:** Needs a .ico file (which contains multiple sizes like 16x16, 32x32).
- **Mac:** Needs a .icns file (Apple's specific high-res format).
- **Linux:** Is the most flexible; it will happily take a standard .png.

Since you're using this for **NotebookLM**, having these distinct icons for different subjects can make your pre-upload workflow feel much more professional and organized.