ChatGPT

Synchronize Working Files to Notion

This repository contains a sample Python script (`sync_working_files_to_notion.py`) that demonstrates how to synchronize file metadata stored in a Google Sheet into a Notion database.
It is designed for teams who use Google Drive as the source-of-truth for marketing and operational documents but want a unified, searchable index in Notion.

Prerequisites

1. **Google Cloud Service Account**
2. Create a service account in the Google Cloud console and enable the **Google Sheets API** and **Google Drive API**.
3. Share the target Google Sheet and any Drive folders/files you wish to access with the service account's client email.
4. Download the service account's JSON key and save it locally.
5. **Notion Integration**
6. Create a Notion integration via Notion Integrations and share your target Notion database with the integration.
7. Note your integration's secret and the database ID of the table you want to populate (e.g. "File Library").
8. **Python Environment**
9. Install dependencies:

```
pip install google-api-python-client google-auth-httplib2 google-auth-oauthlib notion-client
```

Environment Variables

Set the following environment variables before running the script:

- `GOOGLE_SERVICE_ACCOUNT_FILE`: Path to your service account JSON key file.
- `SHEET_ID`: The ID of the Google Sheet containing the data to sync.
- `WORKSHEET_NAME`: Name of the worksheet/tab to process (e.g. `Microneedling`).
- `NOTION_API_KEY`: Your Notion integration secret.
- `NOTION_DATABASE_ID`: The ID of the Notion database where entries should be created.

For example, on Linux/macOS:

```
export GOOGLE_SERVICE_ACCOUNT_FILE="/path/to/your/service-account.json"
export SHEET_ID="your-google-sheet-id"
export WORKSHEET_NAME="Microneedling"
export NOTION_API_KEY="secret_api_key"
export NOTION_DATABASE_ID="your-notion-db-id"
```

Running the Script

Execute the script using Python:

```
python sync_working_files_to_notion.py
```

The script will: 1. Determine which rows are new since the last run (tracked in `last_synced_row.txt`).
2. For each new row, it constructs the Notion page properties from the columns (e.g. File Name, Category, Remark, File Date).
3. If a "PDF Link" is provided, it will be stored as the page's link; otherwise it will attempt to convert a Google Docs/Slides file at the "Link" URL into a PDF via the Drive API.
4. Create a new page in your Notion database with these properties.
5. Update `last_synced_row.txt` so subsequent runs only process newly-added rows.

Customization Notes

- **File Uploads**: The provided implementation attaches a link to the file rather than uploading the binary. To upload PDFs directly to Notion, you must first upload the file to a public storage service (e.g. S3) or use Notion's file upload API (requires a signed URL).
- **Multiple Worksheets**: You can reuse the `read_sheet_rows` function for different tabs by changing `WORKSHEET_NAME` or looping through a list of sheet names.
- **Data Schema**: Modify the `properties` dictionary in the script to match the column names and property types defined in your Notion database.

Next Steps

1. Commit both `sync_working_files_to_notion.py` and this `README_SYNC.md` to your GitHub repository (e.g. `Chengmun58/mumsrelle-ops`).
2. In your Make.com scenario, invoke this script via a scheduled run or a custom webhook each time the Google Sheet is updated.
3. Adjust the code to handle other file types (e.g. images, spreadsheets) and to log errors for missing permissions.

Feel free to expand this script to suit your team's workflow, such as adding logging, error handling, or support for updating existing Notion entries.
