Developing a Cloud-Based Service  
for Basic Data Analytics

|  |
| --- |
| “Student/s name/s and ID/s”  احمد بسام أبو كرش 120200572  محمد صلاح المصري 120200165  عبدالله سالم اليازجي 120210282  Computer Science Department  Faculty of Information Technology  Islamic University of Gaza  A Requirement for the Course: Cloud and Distributed Systems (SICT 4313)  Instructor: Dr. Rebhi S. Baraka |

# Abstract

This report presents **Document Collector**, a desktop application built with Python’s Tkinter library to manage PDF and Word documents. The application allows users to manually upload files or download them from Google Drive, sort by title, perform keyword searches, classify documents according to a predefined taxonomy, and view statistics on file counts, total size, and operation timings.

# Introduction

As the volume of digital documents grows in work environments, there is a pressing need for a simple yet powerful tool to organize, search, and categorize these files quickly. **Document Collector** addresses this need by providing an intuitive graphical interface and leveraging open‑source Python libraries for document handling and the Google Drive API for cloud integration.

# Cloud Software Program/Service Requirements

* **Software Requirements**
  1. Python 3.8 or newer
  2. Libraries:
     1. tkinter (for the GUI)
     2. PyMuPDF (fitz) and PyPDF2 (for PDF processing)
     3. python-docx (for DOCX processing)
     4. google-api-python-client and google-auth (for Google Drive integration)
* **Access Requirements**
  1. OAuth 2.0 credentials file (token.json) authorized for the drive.file scope.
* **System Requirements**
  1. OS supporting Tkinter (Windows, macOS, Linux)
  2. Internet connection for Drive operations

# Software Architecture and Design

* **User Interface (UI)**
  + Main window with buttons for:
    - Manual file upload
    - Download from Google Drive
    - Sort by title
    - Upload to Drive
    - Classify documents
    - Keyword search
    - Show statistics
* **Functional Modules**
  + **Local Storage**: Saves files under manual\_uploads/.
  + **Google Drive Integration**: Uploads and downloads via Drive API v3.
  + **Title Extraction**: Reads PDF metadata title or first line of text; for DOCX, uses first non‑empty paragraph.
  + **Sorting**: Alphabetical order based on extracted title.
  + **Search**: Full‑text scan of each file with preview and keyword highlighting.
  + **Classification**: Static taxonomy (Health, Education, Computer) matched by keyword presence.
  + **Statistics**: Computes file count, total size, and timing of the last sort/search/classify operations.

# Used Cloud Services and its Interfaces

The cloud service used in this project is Google Drive, integrated using the Google Drive API v3. The primary features utilized include:  
- Uploading documents to a specific Drive folder.  
- Downloading documents from Google Drive that match the required MIME types (application/pdf and .docx).  
- Authentication is handled using OAuth 2.0 and a local credentials file (token.json), which scopes access to drive.file.  
  
These services are accessed using:  
- googleapiclient.discovery.build  
- MediaFileUpload for uploads  
- MediaIoBaseDownload for downloads  
  
The interaction is fully programmatic via Python functions, allowing seamless integration with the GUI.

# Implementation

The application is implemented in Python using the following components:  
  
- Tkinter: for building the GUI with buttons and entry fields.  
- Google Drive API: for cloud-based storage and retrieval of documents.  
- PyMuPDF (fitz), PyPDF2: for PDF processing.  
- python-docx: for DOCX processing.  
- os and file handling: to manage local directories and documents.  
  
Key functions include:  
- upload\_to\_drive(file\_path)  
- download\_files\_from\_drive()  
- extract\_title(file\_path)  
- sort\_documents\_by\_title()  
- search\_documents(keywords)  
- classify\_documents()  
- get\_documents\_stats()  
  
Each of these functions is linked to GUI buttons in the Tkinter interface.

# Data

The application works with files stored locally in the manual\_uploads/ directory. Files are either PDF or DOCX format. There is no structured data model (e.g., no database used), but file metadata is handled in-memory during execution. Future versions can incorporate NoSQL storage (e.g., Firebase) to save extracted metadata or classification results.

# The Used Cloud Platform

The application uses Google Drive as its cloud platform. It does not utilize platforms like Firebase or App Engine yet. Cloud architecture includes:  
- Google Drive API for storage.  
- Python application for accessing Drive APIs.  
- OAuth 2.0 for authentication.  
  
Files are stored in a specific Drive folder identified by its ID. Drive interactions are handled via the Google API Python Client.

# Deployment on the Platform

Deployment is limited to local machines where the Python application runs. It interacts with Google Drive online but is not hosted on a cloud runtime like Google App Engine. However, future enhancements may deploy the service as a web app using platforms such as Streamlit, Flask, or Firebase.

# User Support

Users interact with the application through a graphical user interface. Actions include selecting files, searching, classifying, and viewing statistics. Instructions are displayed in Arabic with icons for clarity.  
  
Links:  
- GitHub Repository:

https://github.com/cloud123457/cloud  
- Drive folder access via the authenticated account.

https://drive.google.com/drive/folders/1xIuZs\_-AwFjM7D7Yc8-YWc9gXULl\_SYG?usp=sharing

# Conclusion

The Document Collector provides essential document handling features in an accessible interface, integrated with Google Drive. The tool supports classification, search, and sorting while keeping the system lightweight.  
  
Challenges include:  
- Incomplete or encrypted PDF files.  
- Limited search quality due to basic text matching.  
- Manual setup of OAuth token.  
  
Future improvements may include cloud deployment, NLP-powered classification, and a persistent backend for metadata storage.

# References

1. Google Developers – Google Drive API: https://developers.google.com/drive

2. PyMuPDF Documentation: https://pymupdf.readthedocs.io/

3. python-docx Documentation: https://python-docx.readthedocs.io/

4. PyPDF2 GitHub: https://github.com/py-pdf/PyPDF2

5. Tkinter Documentation: https://docs.python.org/3/library/tkinter.html