Developing a Cloud-Based Service for Basic Data Analytics

Anas Rami Abu Nahl – 120200928
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 project is a cloud-based service that allows users to upload, sort, search, and classify PDF and Word documents. Using a web interface built with Streamlit and cloud storage via Google Drive API, documents are processed and analyzed for title-based sorting, keyword-based searching with highlighting, and category classification. It supports uploading documents to the cloud, extracting metadata and content, and performing analytics operations all in a user-friendly environment.

The system was developed using Python libraries such as PyPDF2, python-docx, and scikit-learn, and deployed using Google Cloud services. The main goal of the project was to demonstrate cloud-based data analytics as part of the Cloud and Distributed Systems course (SICT 4313).

1. Introduction

This software system is designed to provide core functionalities of document analytics in a cloud-based environment. It enables uploading of PDF and DOCX files, extracts document content and titles, sorts documents, allows full-text search with result highlighting, and classifies documents using a keyword-based classifier. The system was built using a modular and agile methodology with separation between frontend (Streamlit), backend (Python), and cloud (Google Drive).

This software documentation report explains the software design methodology and documents all the required functions, architecture, algorithms, implementation decisions, and the deployment process of the cloud-based analytics system.

2. Cloud Software Program/Service Requirements

The main requirements of the software system include:

* Upload PDF and DOCX files through a web interface.
* Store uploaded documents automatically to Google Drive using the Google Drive API.
* Extract the content and title (from document text, not file name).
* Sort the documents based on the extracted title.
* Search for a keyword or phrase in all documents and highlight matches.
* Classify documents into categories based on a predefined classification tree and keywords.
* Display statistics (number of documents, size, execution time of each operation).

The system works in layers: a user interface layer (Streamlit), a processing layer (Python functions), and a storage layer (Google Drive).

3. Software Architecture and Design

System Architecture Diagram

The system follows a modular architecture consisting of the following components:

* Frontend (Streamlit Web App): For file upload, interaction, and display.
* Backend Processing:
  * File parsing and title extraction (PyPDF2, python-docx)
  * Search and highlighting
  * Classification engine (keyword-based)
* Cloud Storage: Google Drive via Google API

[User Interface] ⇄ [Streamlit App] ⇄ [Processing Layer (Python)] ⇄ [Google Drive API]

Functional Components

* upload_to_drive(): Uploads files to Google Drive.
* extract_text_title(): Parses documents and extracts titles.
* search_documents(): Searches text and returns highlights.
* classify_documents(): Categorizes documents based on keywords.

Design Decisions

* Used Streamlit for fast web deployment.

* Used Google Drive for scalable cloud storage.
* Maintained loose coupling between interface and logic to ease testing and expansion.

4. Used Cloud Services and Interfaces

* Google Drive API: For file upload and cloud storage.
* Google Cloud Console: To create service account and credentials.
* Streamlit Sharing / Local Deployment: For frontend interaction.

5. Implementation

The application is implemented in Python. It uses:

* streamlit for UI
* PyPDF2 for PDF parsing
* python-docx for DOCX reading
* google-api-python-client for Drive interaction
* scikit-learn for optional classification enhancements

The source code is organized in modular form inside utils/:

* drive.py, parser.py, search.py, classifier.py

6. Data

* The data consists of user-uploaded PDF and DOCX files.
* Files are stored directly in the user's Google Drive.
* The document text is stored temporarily in memory during processing.
* Classification relies on keyword matching, not database storage.

7. The Used Cloud Platform

* The system is integrated with Google Drive for real cloud storage.
* Google Cloud Platform was used to create a service account and download a credentials file (credentials.json).
* Files are uploaded using MediaIoBaseUpload via Google Drive API.

8. Deployment on the Platform

The application can be deployed in two ways:

* Locally: Using streamlit run app.py

* Online: Through Streamlit Cloud or via a deployed server with Python environment.

Google Drive is used as external storage, mapped via the API credentials.

9. User Support

To run the system:

1. Clone the GitHub repository.
2. Install dependencies: pip install -r requirements.txt
3. Place credentials.json in the root directory.
4. Run: streamlit run app.py

GitHub Repository: [to be added]
Live Link (if deployed): [to be added]

The interface supports:
* Drag-and-drop file upload
* Search input
* Display of classified results and metrics

10. Conclusion

This project provides a practical example of building a cloud-based data analytics platform. It successfully demonstrates integration with cloud services, real-time document processing, and analytics features through a simple yet effective interface.

Future improvements may include:
* Integrating document OCR for scanned PDFs
* Saving metadata in a NoSQL database
* Adding user authentication

11. References

* Google Drive API Documentation: https://developers.google.com/drive/api
* Streamlit Documentation: https://docs.streamlit.io
* PyPDF2 Library: https://pythonhosted.org/PyPDF2/
* Python-docx Library: https://python-docx.readthedocs.io

12. GitHub link

* https://github.com/AnasAbuNahel/cloud-docs-app.git