

# Technical Report: Cloud-Based Document Organizer

## 1. Introduction

This report outlines the technical details, architecture, and business considerations for a cloud-based document organizer solution. This system aims to streamline document management by automatically extracting information from uploaded files (PDFs, images), making them searchable, and storing them securely. The solution leverages Google Cloud Platform (GCP) services to ensure scalability, reliability, and advanced processing capabilities.

## 2. Case Study: Proposed Project in the International/National Reality

### 2.1 International Context

The global volume of digital documents is expanding exponentially, driven by increased internet usage, mobile device adoption, and business digitization efforts. Organizations across various sectors (legal, healthcare, finance, education) face challenges in managing this vast amount of information. Existing solutions often lack the following:

- **Intelligent Processing:** Many systems rely on manual tagging and organization, which is time-consuming and error-prone.
- **Scalability:** Traditional on-premises solutions struggle to handle the increasing volume of data.
- **Accessibility:** Documents stored in disparate systems are difficult to access and share.
- **Advanced Search:** Basic keyword search is often insufficient for finding specific information within documents.

Several international trends highlight the need for a solution like ours:

- **AI-Powered Automation:** There's a growing demand for AI-driven solutions that can automate document processing tasks, such as data extraction and classification.
- **Cloud Adoption:** Organizations are increasingly migrating to the cloud to leverage its scalability, cost-effectiveness, and security.
- **Remote Work:** The rise of remote work has increased the need for cloud-based document management systems that can be accessed from anywhere.

## 2.2 National Context

In Romania, similar challenges exist, with specific nuances:

- **Digitalization of Public Sector:** The Romanian government is pushing for increased digitalization of public services, which requires efficient document management solutions.
- **SME Adoption:** Small and medium-sized enterprises (SMEs) in Romania are increasingly adopting cloud technologies to improve their operations, but many still struggle with document management.
- **Specific Industry Needs:** Certain sectors in Romania, such as legal and accounting, handle large volumes of documents that require advanced processing and search capabilities.

Our solution addresses these international and national needs by providing an intelligent, scalable, and accessible platform for organizing and searching documents.

## 3. Technical Details of Existing Solutions

Several existing solutions offer document management capabilities, but they often have limitations:

- **Traditional Document Management Systems (DMS):**
  - Examples: SharePoint, Alfresco.
  - **Architecture:** Typically involve a centralized server and database, often deployed on-premises.
  - **Technologies:** Java, .NET, SQL databases (MySQL, PostgreSQL, SQL Server).
  - **Limitations:** Can be costly to maintain, may not scale well, often lack advanced AI-powered features.
- **Cloud-Based Storage and Sharing Services:**
  - Examples: Google Drive, Dropbox, OneDrive.
  - **Architecture:** Distributed cloud storage with web and mobile interfaces.
  - **Technologies:** Cloud-native technologies, REST APIs.
  - **Limitations:** Primarily focused on file storage and sharing, lack advanced document processing and search features.
- **OCR Software:**
  - Examples: Adobe Acrobat, ABBYY FineReader.
  - **Architecture:** Desktop applications or server-based software.
  - **Technologies:** Proprietary OCR engines.
  - **Limitations:** Focus on text extraction, lack document organization and search capabilities, not designed for cloud-native scalability.










## 4. Overview of Technologies and Motivation


Our solution leverages the following Google Cloud technologies:

- **Google Cloud Storage (GCS):**
  - **Motivation:** Scalable, durable, and secure storage for uploaded documents. GCS provides a central repository for all files, accessible to other services.
- **Document AI:**
  - **Motivation:** Provides advanced OCR and document understanding capabilities to extract text and structure from PDFs, enabling intelligent processing, including entity extraction.
- **Cloud Vision API:**
  - **Motivation:** Offers powerful OCR capabilities for extracting text from images, ensuring that the system can handle various document formats.
- **Cloud Firestore:**
  - **Motivation:** A NoSQL database for storing document metadata (extracted text, keywords, file location). Firestore's scalability and flexible data model are well-suited for this purpose.
- **Google Cloud Natural Language API:**
  - **Motivation:** To provide advanced NLP capabilities, including entity extraction and key phrase extraction, offering more in-depth analysis of the document content.
- **Elasticsearch (\*):**
  - **Motivation:** A powerful search and analytics engine that can be integrated for enhanced search capabilities, especially for large datasets and complex search queries.
- **Google Cloud Pub/Sub:**
  - **Motivation:** Asynchronous messaging service to decouple file upload from processing. This improves performance and scalability by allowing the upload process to complete quickly, while the more time-consuming processing happens in the background.
- **Cloud Functions:**
  - **Motivation:** Serverless compute platform to execute the document processing logic (text extraction, keyword generation, entity extraction) in response to Pub/Sub messages. Cloud Functions scales automatically and reduces operational overhead.

- **App Engine:**
  - **Motivation:** Platform as a service to deploy the application.
- **Spring Boot (Backend):**
  - **Motivation:** A robust and popular Java framework for building the backend API.
- **React (Frontend):**
  - **Motivation:** A modern JavaScript library for building a dynamic and user-friendly web interface.


## 5. Business Canvas

The Business Model Canvas		Designed for:	Designed by:	Date:	Version:
<b>Key Partnerships</b>  Google Cloud Platform	<b>Key Activities</b>  Software development and maintenance Customer support Marketing and sales	<b>Value Propositions</b>  Automated document organization Intelligent search capabilities including entity and key phrase search Secure and scalable cloud-based platform Improved productivity and efficiency	<b>Customer Relationships</b>  Online support Documentation and tutorials	<b>Customer Segments</b>  Small and medium-sized businesses Legal firms Healthcare organizations Government agencies	
<b>Key Resources</b>  Software development team Cloud infrastructure Data storage		<b>Channels</b>  Website Online marketing			
<b>Cost Structure</b>  Cloud infrastructure costs (storage, processing) Development and maintenance costs Marketing and sales expenses			<b>Revenue Streams</b>  Subscription fees (tiered pricing based on usage/features)		




Turn ideas into revenue with Strategyzer's innovation programs

Copyright Strategyzer AG  
The makers of Business Model Generation and Strategyzer



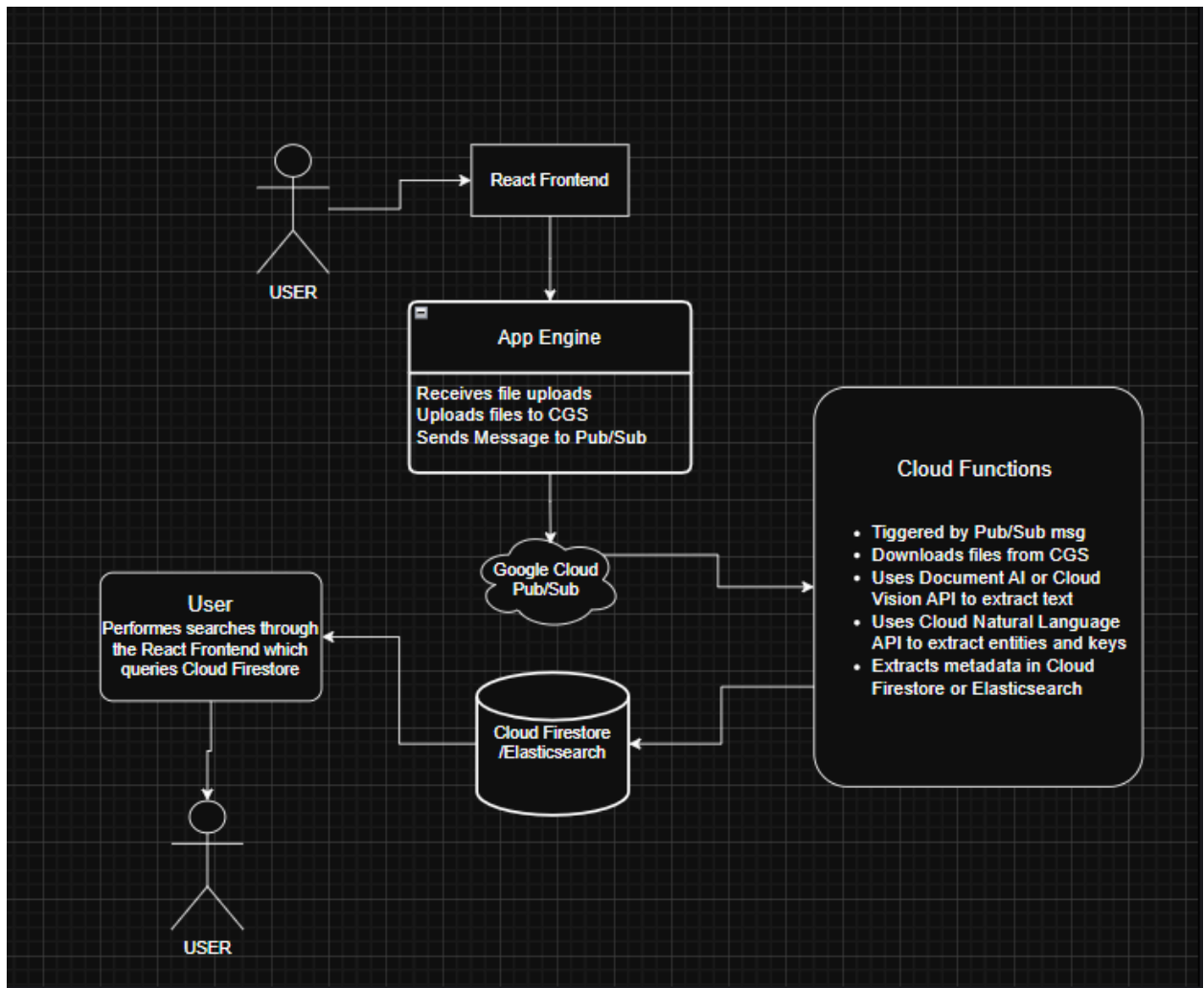
This work is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/> or send a letter to Creative Commons, P.O. Box 1808, Mountain View, CA 94042, USA.



strategyzer.com/innovation

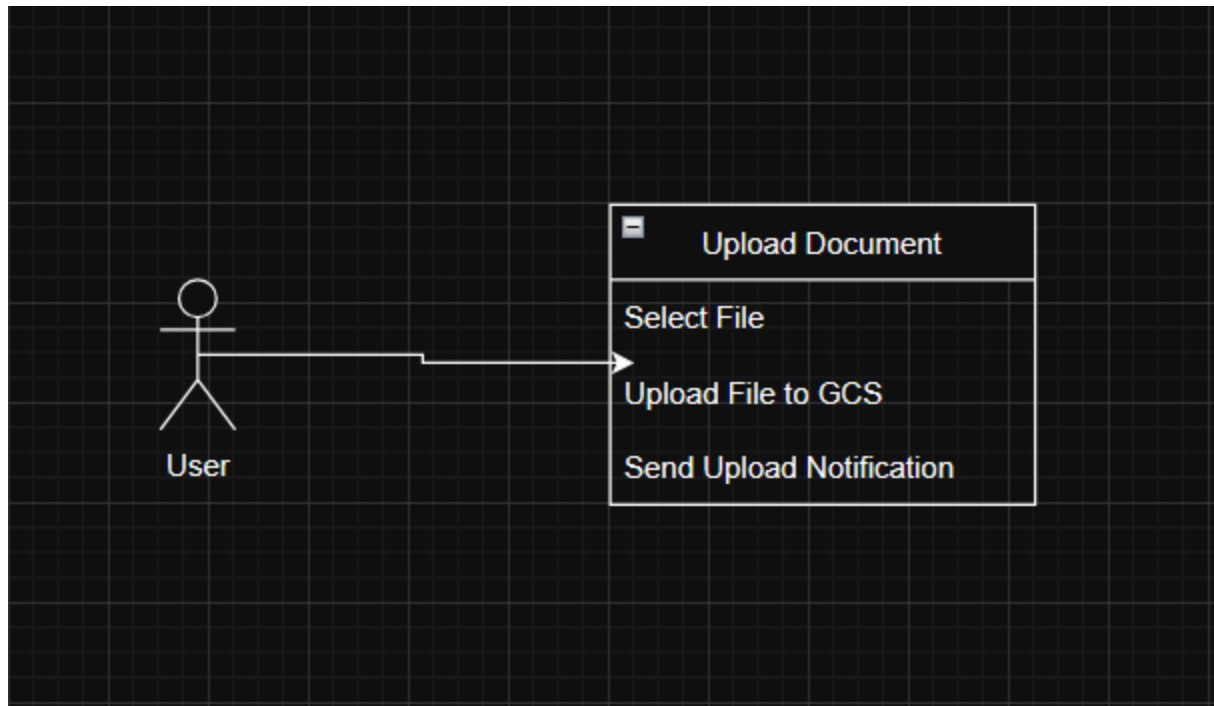
## 6. Architectural Diagram

Here's a diagram illustrating the architecture of the proposed solution:

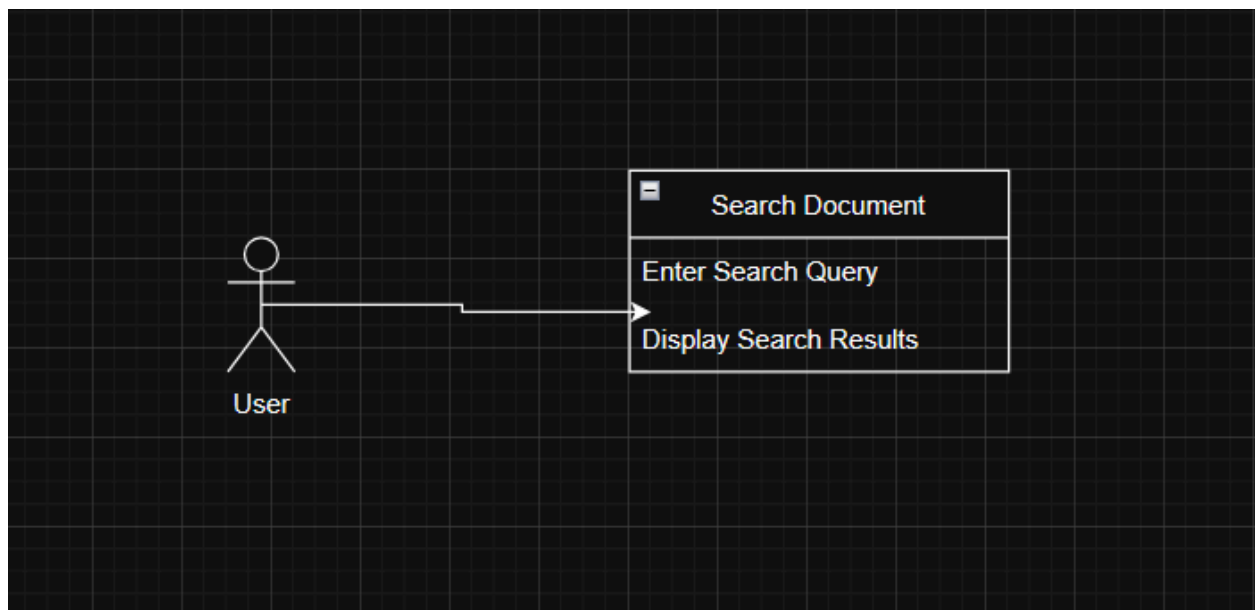


## 7. Use Case Diagrams

### 7.1 Upload Document Use Case



## 7.2 Search Document Use Case



## 8. APIs & Functionality Flows Documentation

### 8.1 API Documentation (OpenAPI Specification using SwaggerHub)

**SwaggerHub Project:** Document Organizer API

**Base URL:** <https://rich-surge-455615-u1.ey.r.appspot.com/api>

#### Endpoints:

- **POST /documents/upload:**
  - **Description:** Uploads a document (PDF or image) and processes it.
  - **Request Body:**
    - file (file): The document to upload.
  - **Response:**
    - 200 OK: Returns the GCS URI of the uploaded file.
    - 400 Bad Request: If the file is empty or invalid.
    - 500 Internal Server Error: If an error occurs during upload or processing.
- **GET /documents/search:**
  - **Description:** Searches for documents based on a keyword.
  - **Query Parameters:**
    - keyword (string, required): The keyword to search for.
  - **Response:**
    - 200 OK: Returns an array of document objects, where each object contains the document's metadata (URI, type, extractedText, entities, keyPhrases, keywords).
    - 500 Internal Server Error: If an error occurs during the search.

# Document Organizer API

1.0

OAS 3.0

API for managing and searching documents.

Servers

<https://rich-surge-455615-u1.ey.r.appspot.com/api> - Product... ▾

default



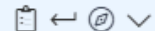
**POST**

**/documents/upload** Upload a document



**GET**

**/documents/search** Search documents



SSL certificates are validated | [Do not validate](#) ⓘ

Routing requests via proxy | [Use browser instead](#) ⓘ

## 8.2 Functionality Flows

### 8.2.1 Document Upload Flow

1. The user selects a document (PDF or image) through the web interface.
2. The React frontend sends a POST request to the `/documents/upload` endpoint on the App Engine backend.
3. The App Engine backend receives the file and uploads it to Google Cloud Storage (GCS).
4. App Engine publishes a message to a Google Cloud Pub/Sub topic. The message contains the GCS URI of the uploaded file.
5. A Cloud Function, subscribed to the Pub/Sub topic, is triggered by the message.
6. The Cloud Function retrieves the file from GCS.
7. The Cloud Function uses Document AI (for PDFs) or Cloud Vision API (for images) to extract the text from the file.



8. The Cloud Function uses the Google Cloud Natural Language API to extract entities and key phrases from the text.
9. The Cloud Function extracts keywords from the extracted text.
10. The Cloud Function stores the document's metadata (GCS URI, file type, extracted text, entities, key phrases, keywords) in Cloud Firestore or Elasticsearch. The choice of which datastore to use can be a configuration option.
11. The App Engine backend returns the GCS URI to the React frontend.
12. The React frontend displays a success message to the user.

### 8.2.2 Document Search Flow

1. The user enters a search query (keyword) in the web interface.
2. The React frontend sends a GET request to the /documents/search endpoint on the App Engine backend, including the keyword as a query parameter.
3. The App Engine backend receives the search request.
4. The App Engine backend queries either Cloud Firestore or Elasticsearch, depending on the configured search solution.
  - **Cloud Firestore:** If using Firestore, the FirestoreService is called to perform the query.
  - **Elasticsearch:** If using Elasticsearch, the backend sends a request to the Elasticsearch cluster.
5. The selected datastore returns the matching document data to the App Engine backend.
6. The App Engine backend formats the search results and sends them to the React frontend.
7. The React frontend displays the search results to the user.