Smart Document Management System

Objective:

Develop a Document Management System that allows users to upload various types of documents (PDF, Word, Image), automatically extract their content, classify the document type, and store them in an organized folder structure with metadata saved to a database.

Deliverables:

A working Flask API for uploading documents.

Text extraction logic using OCR and parsing libraries depending on the file type.

Rule-based logic to classify documents into types such as Resume, Invoice, Medical, and Others.

A mechanism to store documents in structured folders based on classification.

A database schema and code to log document metadata such as filename, upload time, and document type.

Basic HTML + JavaScript interface for file upload and search.

Tech Stack:

Language: Python

Framework: Flask (for backend APIs)

OCR/Text Extraction: Tesseract (for images), pdfplumber (for PDFs), python-docx (for Word docs)

File Detection: python-magic (to detect MIME types)

Storage: Local file system structured by document category

Database: SQLite or PostgreSQL (for logging metadata)

UI: Basic HTML + JavaScript (for file upload/search interface)

Task Details:

Set Up Environment

Initialize a GitHub repository.

Create and configure a Flask project with required dependencies.

File Upload Handling

Implement a POST API that allows users to upload files.

Ensure proper validations: file type, file size, and secure naming.

File Type Detection

Use python-magic to accurately determine file MIME type.

Decide processing pipeline (OCR vs parser) based on the file type.

Text Extraction

For PDFs: Use pdfplumber

For DOCX files: Use python-docx

For images: Use pytesseract with PIL

Document Classification

Implement rule-based classification based on keywords (e.g., “Invoice No.”, “Skills”, “Prescription”).

Define at least 4 document types: Resume, Invoice, Medical, Others.

Organized Storage

Store uploaded files in directories like /documents/invoice/, /documents/resume/, etc.

Ensure files are named uniquely (with timestamp or UUID).

Metadata Logging

Design a database schema with columns like: id, filename, file_path, doc_type, upload_time.

Use SQLite or PostgreSQL and integrate with Flask.

Search & Retrieval

Create a GET API to search documents by type or keyword.

Enable file download or preview via API.

Front end Interface

Build a minimal HTML + JS interface to upload documents and search for them.

Testing and Documentation

Write clear README with setup instructions, dependencies, and usage.

Include sample input files and screenshots (if UI is implemented).