Project

Django Application "OCR PDF Processor"

Objective

Develop a web application using Django that takes PDF files as input, uses OCR (Optical Character Recognition) to extract text, stores this text in a database, and generates a carousel of images from the first three pages of the PDF.

Note on the Deliverable

The final product delivered can be in the form of an MVP (Minimum Viable Product) depending on the duration of the internship. This means that some non-essential features may be simplified or postponed to a later phase to ensure the delivery of a functional product within the given timeframe.

Main Features

PDF File Upload

- User interface to upload PDF files.
- Validation of files to ensure they are in PDF format and comply with the maximum allowed size.
- Configuration to define the maximum upload size and accepted PDF file encodings.

Text Extraction via OCR

- Use an OCR library (e.g., Tesseract) to extract text from PDF files.
- Store the extracted text in a relational database (e.g., PostgreSQL).

Image Generation for Carousel

- Extract the first three pages of the PDF files as images.
- Display the images as a carousel on the user interface.

Display Extracted Data

- User interface to display the extracted text from PDFs.

Project

- Display the image carousel.

User Management

- User authentication and authorization system.

- Admin interface to manage uploaded files and users.

Document Categorization

- Ability to categorize uploaded documents.

- Interface to manage document categories.

Technical Architecture

Backend

- Framework: Django

- OCR: Tesseract OCR or an alternative

- Database: PostgreSQL

- File Storage: Use Django File Storage to store PDFs

Frontend

- Framework: Django Templates, Bootstrap for responsive design

- Carousel: Use a JavaScript plugin for the carousel (e.g., Slick Carousel or Bootstrap Carousel)

Deployment

- Use Docker to containerize the application.

- Host on a cloud platform (e.g., Heroku, AWS, or DigitalOcean).

Feature Details

PDF File Upload

- Upload Page: Form to allow users to upload PDF files.

- Validation: Verify that the uploaded file is a PDF, complies with the maximum size, and accepted encodings.

- Configuration:

- Configurable maximum upload size (e.g., 10MB per file).
- Accepted PDF encodings (e.g., UTF-8, ISO-8859-1).

- Storage: Save the file on the server.

Text Extraction via OCR

- OCR Process: Use Tesseract to extract text from the PDF.

- Text Storage: Save the extracted text in a table in the database.

Image Generation for Carousel

- Page Extraction: Use a library like PyMuPDF or PDF2Image to extract the first three pages as images.

- Carousel: Create an image carousel to display these pages on the user interface.

Display Extracted Data

- Extracted Text Page: Display the extracted text from each PDF on a dedicated page.

- Image Carousel: Integrate the image carousel on the same page or a separate page.

User Management

- Registration/Login: Forms for user registration and login.

- Roles and Permissions: Define user roles with different levels of permissions.

- Admin: Admin interface to manage users and files.

Document Categorization

- Category Creation: Interface to create and manage document categories.

- Category Assignment: Allow users to assign a category to the document during upload.

- Filtering and Search: Allow search and filtering of documents by category.

Timeline

Phase 1: Initial Configuration and File Upload

- Deliverables: Upload form, PDF file validation, file storage, size and encoding configuration.

Phase 2: OCR Implementation and Text Storage

Project

- Deliverables: Text extraction with Tesseract, text storage in the database.

Phase 3: Image Generation and Carousel

- Deliverables: Page extraction as images, display in a carousel.

Phase 4: Display Extracted Data

- Deliverables: Pages to display extracted text and image carousel.

Phase 5: User Management

- Deliverables: Authentication system, admin interface.

Phase 6: Document Categorization

- Deliverables: Category creation and management, document category assignment, category-based search and filtering.

Phase 7: Testing and Deployment

- Deliverables: Unit and integration tests, Docker containerization, deployment on a cloud platform.

Recommended Tools

- IDE: Visual Studio Code or PyCharm

- Version Control: Git and GitHub

- Python Libraries: Django, Tesseract, PyMuPDF, Pillow

- Others: Docker, Postman for API testing, Bootstrap for frontend

Success Criteria

- The application should allow PDF file uploads and reliable text extraction via OCR.

- The first three pages of the PDF should be converted to images and displayed in a carousel.

- The extracted text should be correctly stored and accessible in the database.

- The user interface should be intuitive and responsive.

- The authentication system should be secure and functional.

- Documents should be categorizable, and users should be able to filter them by category.

Support and Documentation

Project

- Provide detailed documentation of the code and deployment steps.

- Prepare a user guide to explain how to use the application.

- Set up a bug tracking and improvement system.