Application Django "OCR PDF Processor"

Objectif

Développer une application web utilisant Django qui prend en entrée des fichiers PDF, utilise l'OCR (Reconnaissance Optique de Caractères) pour extraire le texte, stocke ce texte dans une base de données, et génère un carrousel d'images des trois premières pages du PDF.

> **Note sur le Livrable**
>
> Le produit final livré peut être sous la forme d'un MVP (Produit Minimum Viable) en fonction de la durée du stage. Cela signifie que certaines fonctionnalités non essentielles pourraient être simplifiées ou reportées à une phase ultérieure afin de garantir la livraison d'un produit fonctionnel dans les délais impartis.

Fonctionnalités Principales

1. **Upload de Fichiers PDF**
    - Interface utilisateur pour télécharger des fichiers PDF.
    - Validation des fichiers pour s'assurer qu'ils sont au format PDF et respectent la taille maximale autorisée.
    - Paramétrage pour définir la taille maximale d'upload et les encodages de fichiers PDF acceptés.
2. **Extraction de Texte via OCR**
    - Utilisation d'une bibliothèque OCR (par exemple, Tesseract) pour extraire le texte des fichiers PDF.
    - Stockage du texte extrait dans une base de données relationnelle (par exemple, PostgreSQL).
3. **Génération d'Images pour le Carrousel**
    - Extraction des trois premières pages des fichiers PDF sous forme d'images.
    - Affichage des images sous forme de carrousel sur l'interface utilisateur.
4. **Affichage des Données Extraites**
    - Interface utilisateur pour afficher le texte extrait des PDF.
    - Affichage du carrousel d'images.
5. **Gestion des Utilisateurs**
    - Système d'authentification et d'autorisation des utilisateurs.
    - Interface d'administration pour gérer les fichiers téléchargés et les utilisateurs.
6. **Catégorisation des Documents**
    - Possibilité de catégoriser les documents téléchargés.
    - Interface pour gérer les catégories de documents.

Architecture Technique

1. **Backend**
    - **Framework**: Django

    - **OCR**: Tesseract OCR ou une alternative
    - **Base de Données**: PostgreSQL
    - **Stockage des Fichiers**: Utilisation de Django File Storage pour stocker les PDF
2. **Frontend**
    - **Framework**: Django Templates, Bootstrap pour le design responsive
    - **Carrousel**: Utilisation d'un plugin JavaScript pour le carrousel (par exemple, Slick Carousel ou Bootstrap Carousel)
3. **Déploiement**
    - Utilisation de Docker pour conteneuriser l'application.
    - Hébergement sur une plateforme cloud (par exemple, Heroku, AWS, ou DigitalOcean).

Détails des Fonctionnalités

1. **Upload de Fichiers PDF**
    - **Page de téléchargement**: Formulaire pour permettre aux utilisateurs de télécharger des fichiers PDF.
    - **Validation**: Vérifier que le fichier téléchargé est bien un PDF, respecte la taille maximale et les encodages acceptés.
    - **Paramétrage**:
        - Taille maximale d'upload configurable (par exemple, 10MB par fichier).
        - Encodages PDF acceptés (par exemple, UTF-8, ISO-8859-1).
    - **Stockage**: Sauvegarde du fichier sur le serveur.
2. **Extraction de Texte via OCR**
    - **Processus OCR**: Utiliser Tesseract pour extraire le texte du PDF.
    - **Stockage du Texte**: Enregistrement du texte extrait dans une table de la base de données.
3. **Génération d'Images pour le Carrousel**
    - **Extraction des Pages**: Utiliser une bibliothèque comme PyMuPDF ou PDF2Image pour extraire les trois premières pages sous forme d'images.
    - **Carrousel**: Créer un carrousel d'images pour afficher ces pages sur l'interface utilisateur.
4. **Affichage des Données Extraites**
    - **Page de Texte Extrait**: Afficher le texte extrait de chaque PDF sur une page dédiée.
    - **Carrousel d'Images**: Intégrer le carrousel d'images sur la même page ou sur une page distincte.
5. **Gestion des Utilisateurs**
    - **Inscription/Connexion**: Formulaires pour l'inscription et la connexion des utilisateurs.
    - **Rôles et Permissions**: Définir des rôles d'utilisateurs avec différents niveaux de permissions.
    - **Admin**: Interface d'administration pour gérer les utilisateurs et les fichiers.
6. **Catégorisation des Documents**
    - **Création de Catégories**: Interface pour créer et gérer des catégories de documents.

    - **Assignation de Catégories**: Lors de l'upload, permettre à l'utilisateur d'assigner une catégorie au document.
    - **Filtrage et Recherche**: Permettre la recherche et le filtrage des documents par catégorie.

Chronogramme

1. **Phase 1: Configuration Initiale et Upload de Fichiers**
    - Livrables: Formulaire d'upload, validation des fichiers PDF, stockage des fichiers, paramétrage de la taille et de l'encodage.
2. **Phase 2: Implémentation de l'OCR et Stockage du Texte**
    - Livrables: Extraction de texte avec Tesseract, stockage du texte dans la base de données.
3. **Phase 3: Génération d'Images et Carrousel**
    - Livrables: Extraction des pages en images, affichage en carrousel.
4. **Phase 4: Affichage des Données Extraites**
    - Livrables: Pages pour afficher le texte extrait et le carrousel d'images.
5. **Phase 5: Gestion des Utilisateurs**
    - Livrables: Système d'authentification, interface d'administration.
6. **Phase 6: Catégorisation des Documents**
    - Livrables: Création et gestion des catégories, assignation des catégories aux documents, filtrage et recherche par catégorie.
7. **Phase 7: Tests et Déploiement**
    - Livrables: Tests unitaires et d'intégration, conteneurisation avec Docker, déploiement sur une plateforme cloud.

Outils Recommandés

- **IDE**: Visual Studio Code ou PyCharm
- **Gestion de Version**: Git et GitHub
- **Bibliothèques Python**: Django, Tesseract, PyMuPDF, Pillow
- **Autres**: Docker, Postman pour les tests d'API, Bootstrap pour le frontend

Critères de Réussite

- L'application doit permettre le téléchargement de fichiers PDF et l'extraction fiable de texte via OCR.
- Les trois premières pages du PDF doivent être converties en images et affichées dans un carrousel.
- Le texte extrait doit être correctement stocké et consultable dans la base de données.
- L'interface utilisateur doit être intuitive et responsive.
- Le système d'authentification doit être sécurisé et fonctionnel.
- Les documents doivent pouvoir être catégorisés et les utilisateurs doivent pouvoir les filtrer par catégorie.

Support et Documentation

- Fournir une documentation détaillée du code et des étapes de déploiement.

- Préparer un guide utilisateur pour expliquer comment utiliser l'application.
- Mettre en place un système de suivi des bugs et des améliorations.