DS2 2024: Intelligent Document Workflow Management System Using Django 5 and Free Tools

3-BIS and 3-BI

Be Confident, Be Strong, Be Creative, Be Clever—
Leverage Technology to Build Intelligent Systems for the Future.

Project Description

The **Intelligent Document Workflow Management System** is a comprehensive platform designed to revolutionize how organizations manage their documents by automating processes, integrating various APIs, and leveraging advanced artificial intelligence (AI) capabilities. This project combines the power of **Django 5**, a robust Python web framework, with a suite of free and open-source tools to develop a scalable, efficient, and intelligent system.

The platform aims to address common challenges in document workflows, such as inefficient routing, lack of automation, difficulty in integrating legacy systems, and manual document classification. By providing seamless integration, intelligent processing, and dynamic routing, this system will streamline document handling for better productivity and user experience.

Key Objectives:

- **System Integration:** Enable seamless communication between legacy systems and modern applications using **SOAP** and **REST APIs**.
- **Advanced Querying:** Use **GraphQL APIs** to handle complex queries and fetch interconnected data efficiently.
- **CRUD Operations:** Implement Create, Read, Update, and Delete functionalities for managing users, documents, and workflows through APIs.
- **AI Integration:** Leverage tools like **Hugging Face Transformers** for summarization, classification, and contextual processing, and **Haystack** for enhanced data retrieval and query handling.
- **Workflow Optimization:** Use **Reinforcement Learning (RL)** to intelligently route and prioritize workflows, adapting dynamically to organizational needs.
- **User-Friendly Interface:** Develop an intuitive graphical interface to simplify interactions and provide insightful analytics to end-users.

—

Key Features

The platform is built around core features that ensure its functionality, scalability, and intelligence.

1. **User Authentication and Management:**
    - Implement a secure authentication system using Django's built-in authentication framework.
    - Introduce **Role-Based Access Control (RBAC)** to define permissions for different user types:

  - **Administrators:** Full access to manage users, workflows, and documents.
  - **Managers:** Access to manage workflows and monitor document statuses.
  - **Employees:** Limited access to upload, view, and process documents.

2. **Document Integration:**
   - Enable integration with legacy document systems using **SOAP APIs** (e.g., via the `zeep` library).
   - Synchronize documents with modern cloud storage solutions (e.g., Nextcloud) using **REST APIs** built with Django Rest Framework (DRF).
3. **AI-Powered Processing:**
   - **Document Classification:** Automatically categorize documents (e.g., invoices, contracts, reports) using pre-trained NLP models from **Hugging Face Transformers**.
   - **Summarization:** Generate concise summaries of lengthy documents to save time for users.
   - **Workflow Optimization:** Use **Reinforcement Learning (RL)** via `stable-baselines3` to identify the most efficient routing paths for document approvals and processing.
   - **Context-Aware Query Handling:** Use **Retrieval-Augmented Generation (RAG)** with `Haystack` to answer user queries about document content by combining live document retrieval with LLMs (e.g., Open LLAMA).
4. **APIs for CRUD Operations:**
   - **SOAP:** Build SOAP endpoints to handle user and document management for legacy system compatibility.
   - **REST:** Implement RESTful APIs for managing workflows and performing CRUD operations on documents and user data.
   - **GraphQL:** Use **Graphene-Django** to handle complex queries like retrieving workflows with associated users and document statuses.
5. **Graphical User Interface:**
   - Design intuitive dashboards for different user roles using Django templates or **HTMX**.
   - Provide visual insights into workflow statuses, user activities, and document interactions through charts and graphs.

- Enable easy document upload, search, and interaction history tracking from the user interface.

6. **Scalable Architecture:**
   - Implement microservices architecture using Django apps to manage different components like user authentication, document processing, and AI models independently.
   - Use an **API Gateway** (e.g., Kong Gateway CE) to manage API communications securely.

—

Detailed Example: How It Works

To better understand the system, let's walk through an example:

1. **User Workflow:**
   - An employee logs in to the system and uploads a document.
   - The system categorizes the document (e.g., "Invoice") using a pre-trained NLP model.
   - Based on the document type, the workflow routes it to a manager for approval.
   - The manager receives a notification, reviews the document, and either approves or requests modifications.
2. **Query Workflow:**
   - A user asks the system: "What are the pending invoices for this month?"
   - The system retrieves relevant documents from the database using **Haystack** and generates a summary using an LLM.
   - The user receives an AI-generated response with the list of pending invoices.
3. **API Interaction:**
   - Legacy systems use the SOAP API to sync old documents with the system.
   - Modern applications interact with REST APIs to fetch and update document statuses.
   - GraphQL APIs allow developers to query workflows and retrieve insights in a single request.

This example demonstrates the interplay of APIs, AI, and workflows to deliver a seamless user experience.

—

Benefits of the System

- **Improved Productivity:** Automates repetitive tasks like document routing and classification.
- **Enhanced Integration:** Bridges the gap between legacy and modern systems.
- **Scalability:** Modular design ensures the system can grow with organizational needs.
- **Intelligence:** Uses AI to optimize workflows and provide actionable insights.
- **Ease of Use:** Simplifies document management for all user roles with a clean and intuitive interface.

1 Presentation Schedule:

- **Submission deadline:** **17/12/2024**. Submissions must be sent to `homework.isg@gmail.com`.
- **Required materials for submission:**
  - Implemented source code.
  - UML diagrams illustrating models and workflows.
  - Screenshots of key features.
  - A short demo video showcasing the project functionality.
- Projects will be evaluated during the course sessions from **18/12/2024** to **20/12/2024**.

Evaluation Criteria

- Proper implementation of CRUD operations using SOAP, REST, and GraphQL APIs.
- Effective integration of AI components.
- Scalability and usability of the system.
- Quality of documentation and test cases.

Implementation Plan

Week 1: Initialization

- Set up Django 5 project and configure PostgreSQL database.
- Implement user authentication and RBAC.
- Define the UML Class Diagram.

Week 2: Core Development

- Develop CRUD functionality for users and documents using SOAP, REST, and GraphQL.
- Integrate basic AI tools for document classification.

Week 3: Advanced Features

- Integrate advanced AI capabilities (RAG, RL) for document workflows.
- Build the graphical user interface with dashboards for users.

Week 4: Testing and Submission

- Conduct system-wide testing.
- Prepare final documentation, including UML diagrams and test cases.
- Submit materials to `homework.isg@gmail.com`.

Software Stack

- **Framework:** Django 5.
- **APIs:** DRF, Graphene-Django, `zeep`.
- **Database:** PostgreSQL, Oracle, MongoDB or SQLite.
- **AI Libraries:** Hugging Face, `stable-baselines3`, Open LLAMA, Haystack.
- **API Gateway:** Kong Gateway CE.

Additional Notes

- Be creative and think critically while implementing this project.
- Feel free to extend the features with additional functionalities.