Capstone Project: Automated Multilingual Research Submission Processer

1. Project Overview

Project Name: Automated Multilingual Research Submission Processor

Project Description:

A multi-agent AI system that automatically ingests research paper submissions (emails + attachments), extracts structured metadata and key content (title, authors, affiliations, abstract, keywords, figures), validates formatting and compliance (page limits, references), generates a validation summary, and provides a RAG-powered conversational Q&A on stored submissions. The system must support multiple languages, including Human-In-The-Loop (HITL) review mechanism for flagged/low-confidence items, and learn from human corrections to improve future extraction and validation

Business Context:

Research submissions are often received in multiple formats (PDF, Word, Scanned images) and languages. Manual review for quality, plagiarism, formatting compliance and summary generation is time consuming and error prone. This project aims to automate document extraction, validation, summarization and review assistance using a multiagent powered system with human-in-the-loop oversight.

2. Project Requirements:

Design and implement an end-to-end Agentic AI-powered solution that:

1. Monitors an email inbox for incoming invoices. (for this step only design/approach is expected and not the actual implementation)
2. Detect the language of each submission using a language detection model
3. Use OCR for image based or scanned submissions
4. Translates extracted data to English. Store both original and English translations in memory for reference
5. Each submission must pass below defined set of rules
   - Minimum page count is 8
   - Maximum page count is 25
   - Title, abstract, keywords,author and references section should be present
   - Submission should undergo plagiarism check
6. Generate a summary highlighting key findings, major validation issues, missing sections
7. Check for toxicity, illicit content and send for human review

8. Implement an RAG pipeline for retrieval and Q&A
9. Provide an option for the admin to override/correct AI generated validation findings
10. Generate a validation summary report for each submission
11. Log all the actions taken by user/admin
12. The overall solution should be implemented using agentic AI, and should include the following agent roles:

- Ingestion Agent – Continuously monitors the mailbox for emails with invoice attachments (this agent can read the documents from file system rather than from live mailbox)
- Pre-process Agent – Prepares data for Extraction. Validate file type, detect language and use OCR capabilities for scanned documents
- Translation Agent – Translates extracted data from different languages into English
- Extraction Agent – Extract structured fields from submissions. For e.g., title, author, affiliations etc.
- Validation Agent – Validates business rules and semantic checks
- Summary Agent - Produce human-readable summary report for reviewers in not more than 250 words
- RAG Agent – Generate embeddings and maintain vector store for retrieval.
- Q&A Agent – Handle conversational queries about submissions. Support multilingual capability and chat history
- Human Feedback Agent – Present flagged items to admin, accept corrections, and store them. If the deviation if <25%, system should flag for human review.

3. Solution Architecture

The Automated Research Submission Processor is to be built on a modular agentic architecture using Microsoft Agent framework. A RAG-based question answering system enables contextual queries with agents for indexing, retrieval, augmentation, generation, and reflection. The system includes human-in-the-loop feedback, and audit trails for reliability and transparency.

Solution should leverage prompt templates while defining prompts. Solution must leverage below capabilities of Semantic Kernel during implementation of one or more agents

- Semantic Functions
- Native Functions
- Memory
- Plugins
- Filters with Logging
- Agent Framework
- Process Framework

4. Technology Stack

Backend:

- C#
- Microsoft Agent framework
- Vector DB (InMemory or any other supported Vector DB)
- OCR tools (e.g., Tesseract or any other similar tool) - Optional

Models:

- Text Generation models
- Embedding model

Frontend:

- Angular / React

Deployment:

- Local

5. Non-Functional Requirements (Optional)

- Performance: Efficient processing of large invoice batches
- Scalability: Modular agent design allows horizontal scaling
- Security: Secure email access and document handling
- Reliability: Robust error handling and fallback mechanisms
- Usability: Intuitive UI for support agents
- Maintainability: Modular codebase with documentation

6. Input Data set

TensorFlow Paper.pdf

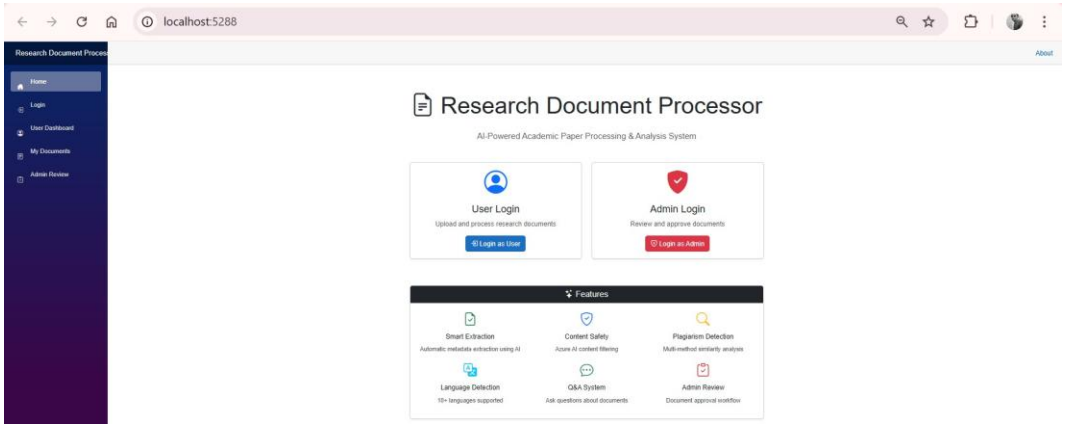
Sample papers can be downloaded from https://arxiv.org/

7. Final Deliverables

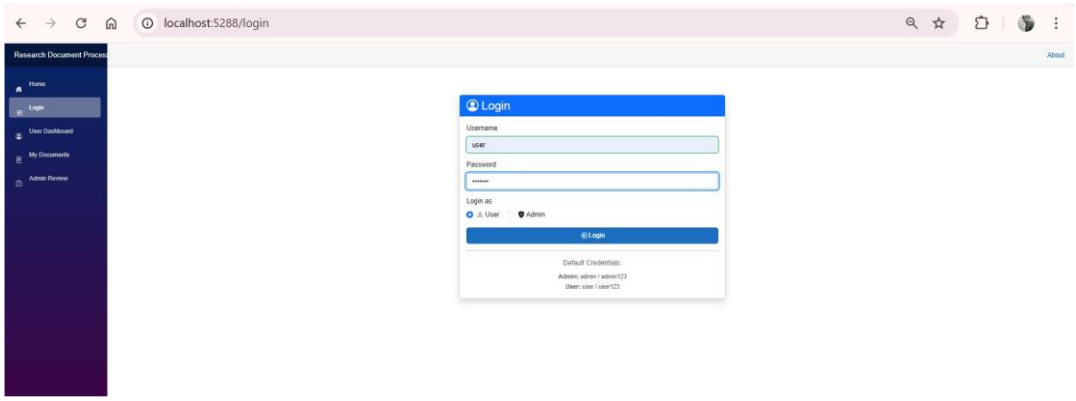
- Solution Design Diagram
- Source Code (C#, modularized)
- Functional Application (demo UI + backend agents)
- Sample Data (multilingual research papers)
- Documentation (setup, usage, agent design, prompt templates) -optional

8. Mock Screenshots

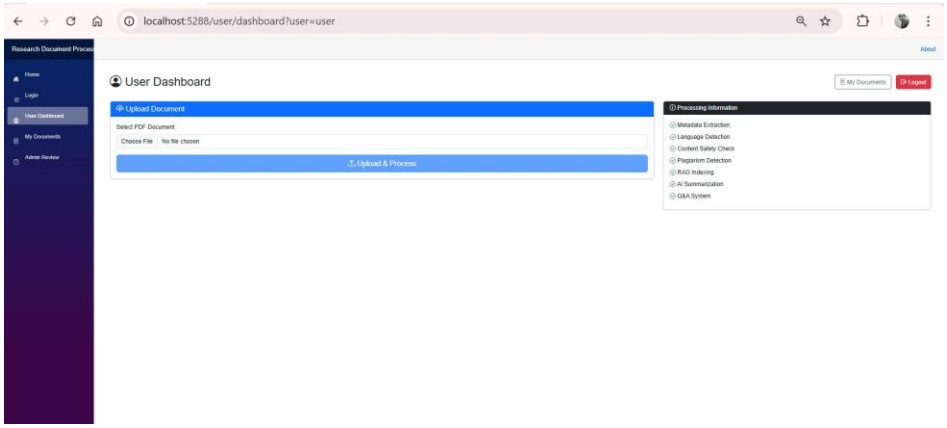
Home Page



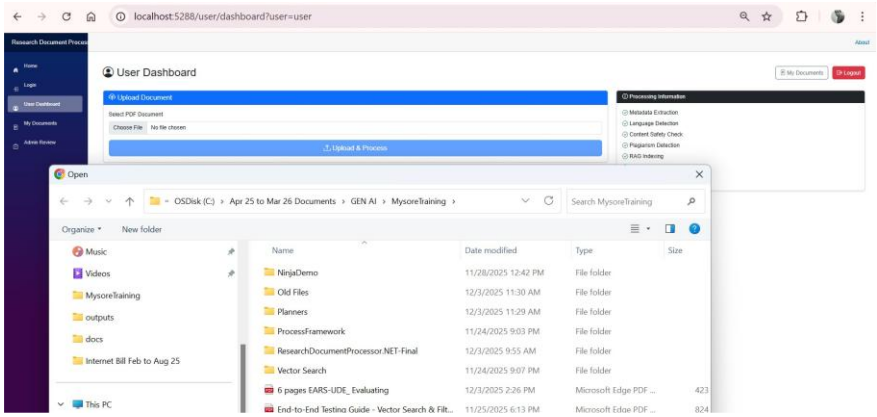
User Login



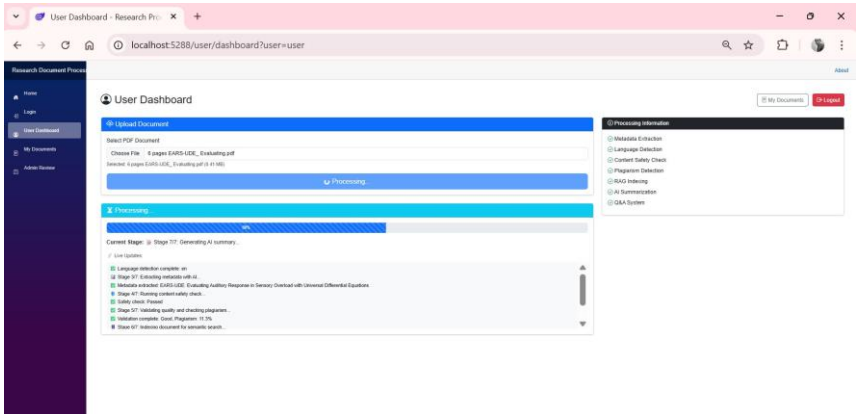
User Dashboard



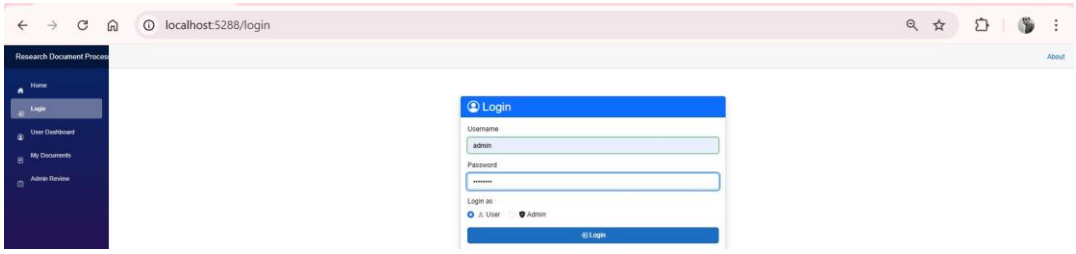
Upload Screen



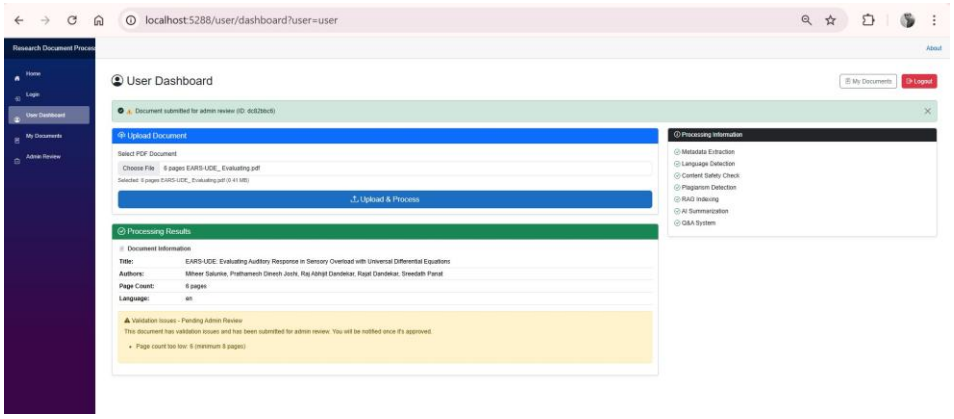
Processing screen



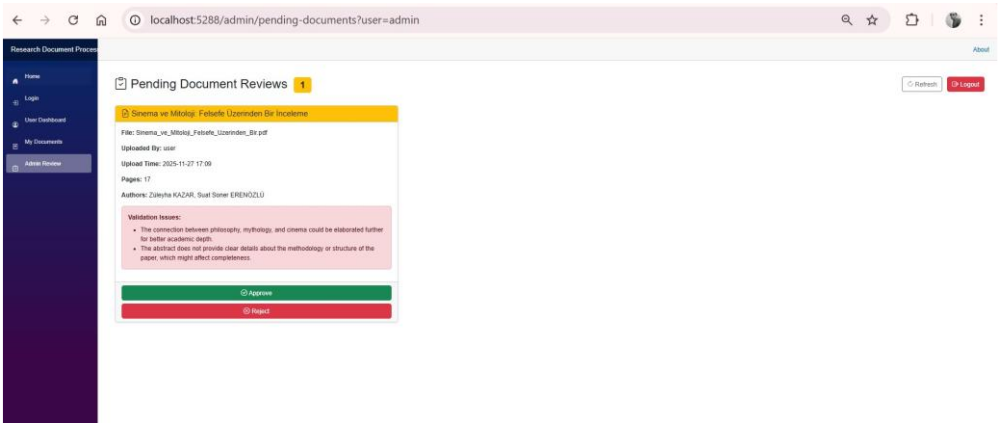
Admin Login



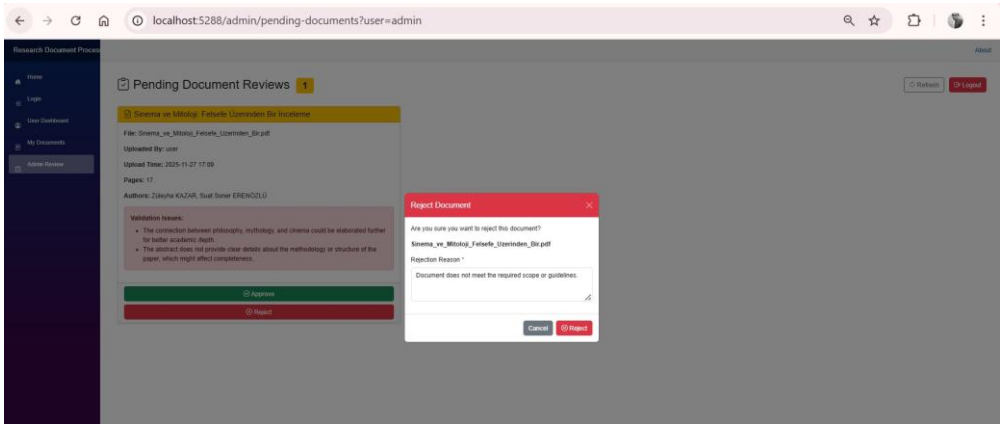
Processed Screen



Admin Dashboard – Approval / Reject Page



Reject Screen 1



Reject Screen 2