Court Document Processing System - Workflow Documentation

System Overview

This is an intelligent banking document processing system that automatically processes court documents (PDFs) to extract national IDs and banking actions, then executes appropriate banking operations. The system leverages a FastAPI backend with PostgreSQL database, React frontend, and LangGraph-based agent workflow for intelligent document processing.

Complete Document Processing Workflow

Step 1: Document Upload

- User accesses the banking system website
- Clicks "Upload Document" button
- Selects a PDF file containing a court order
- System receives the document and initiates processing

Step 2: Document Reading & Text Extraction

Technologies Used: PyPDF2, pdfplumber, pytesseract, Pillow, opencv-python

- System opens the PDF file using PyPDF2 and pdfplumber libraries
- Extracts all text from the document
- **For scanned documents:**
  - Uses OCR (Optical Character Recognition) with pytesseract and Pillow
  - Enhances image quality using opencv-python before OCR processing
- Combines all extracted text into a readable format

Step 3: National ID & Banking Action Identification

Technologies Used: GPT-4o (OpenAI), langchain-openai, sentence-transformers

- System uses GPT-4o to intelligently analyze the document text
- **Extracts:**
  - National identification number from the document

    - Banking instructions and actions
- Creates pairs: one national ID + one banking action
- Uses sentence-transformers to create embeddings of extracted text

Step 4: Customer Verification

Technologies Used: psycopg2-binary, PostgreSQL

- System queries the PostgreSQL customer database using the extracted national ID
- **Possible Outcomes:**
    - ✅ **Customer Found:** Proceed to next step
    - ❌ **Customer Not Found:** Stop processing and notify user "Customer not found"

Step 5: Banking Action Validation

Technologies Used: ChromaDB, semantic search

- Performs semantic search using ChromaDB vector database
- **Valid Banking Actions:**
    - freeze_funds - Temporarily block access to customer funds
    - release_funds - Restore access to previously frozen funds
- Uses AI embeddings to find the closest matching valid action
- **Possible Outcomes:**
    - ✅ **Valid Action Found:** Continue to next step
    - ❌ **No Valid Action:** Stop processing and notify user "No banking instructions found"

Step 6: Action Processing Decision

Only executed if customer exists and valid action is found

- System evaluates confidence level of the banking action match
- **Decision Logic:**
    - **High Confidence:** Execute action automatically
    - **Low Confidence OR Unclear Action:** Send to human reviewer for approval

Step 7A: Automatic Processing (High Confidence Actions)

Technologies Used: Internal banking APIs, pydantic

- System immediately performs the banking action
- Updates customer's account status
- Records action in audit logs using pydantic data validation
- Notifies user "Action completed successfully"

Step 7B: Human Review Process (Low Confidence Actions)

- System queues the request for human review
- **Reviewer Interface Shows:**
  - Customer details
  - Requested action with confidence score
  - Original document text and context
  - Semantic search results showing possible action matches
- **Reviewer Decision:**
  - **Approve:** System executes the banking action
  - **Reject:** No action taken, reason logged

Step 8: Final Results

Possible Outcomes:

- ✅ **"Banking action completed successfully"** - Action executed (freeze/release funds)
- ⚠️ **"Customer not found"** - No matching customer in database
- 📋 **"Sent for review"** - Awaiting human approval
- ❌ **"No banking instructions found"** - Document contained no valid banking actions
- ❌ **"No national ID found"** - Document contained no identifiable customer ID

Sequential Agent Processing Flow

Flow Summary

Preprocessing → Extraction → Validation → Customer Lookup → Action Matching → Review Router → Execution → Logging → END

1. Preprocessing Agent

- **Role:** Entry point of the pipeline
- **Purpose:** Initial document preparation and setup
- **Output:** Prepared document state for extraction

2. Extraction Agent

- **Input:** Preprocessed document
- **Purpose:** Extract raw text from PDF (includes OCR for scanned documents)
- **Technologies:** PyPDF2, pdfplumber, pytesseract, opencv-python
- **Output:** Raw text content

3. Validation Agent

- **Input:** Raw extracted text
- **Purpose:** Use GPT-4o to extract and validate national ID and banking action pairs
- **Technologies:** OpenAI GPT-4o, langchain
- **Output:** Validated pairs with confidence scores

4. Customer Lookup Agent

- **Input:** Validated pairs with national IDs
- **Purpose:** Verify customer existence in database
- **Technologies:** PostgreSQL, psycopg2
- **Critical Behavior:** Zero-tolerance for missing customers - throws AgentError if any customer not found
- **Output:** Customer mappings and verified pairs

5. Action Matching Agent

- **Input:** Verified pairs with customer data

- **Purpose:** Match extracted actions with valid banking operations using semantic search
- **Technologies:** ChromaDB, sentence-transformers
- **Valid Actions:** freeze_funds, release_funds
- **Output:** Action-matched pairs with confidence scores

6. Review Router Agent

- **Input:** Action-matched pairs
- **Purpose:** Decision engine for routing based on confidence levels
- **Logic:**
    - High confidence + valid action → Auto-execution
    - Low confidence or unclear action → Human review queue
- **Output:** Routing decision (auto-execute or review required)

7. Execution Agent

- **Input:** Approved actions (auto or human-reviewed)
- **Purpose:** Execute banking operations on customer accounts
- **Actions:** Perform actual banking operations (freeze/release funds.)
- **Output:** Execution results and status

8. Logging Agent

- **Input:** Final execution results
- **Purpose:** Record complete audit trail and finalize job status
- **Output:** Complete processing logs and job completion
- **Role:** Final agent before workflow termination

Agent Flow Characteristics

Architecture Features

- **Linear Pipeline:** Each agent processes sequentially, passing state forward
- **Fail-Fast Design:** Customer Lookup Agent can terminate workflow immediately
- **State Persistence:** Each agent updates the DocumentState object
- **Complete Audit:** Every step logged for compliance and debugging