Cityflo

Full Stack Developer Assignment

Submission: Live demo and GitHub repository of code along with some documentation

Problem Statement

Cityflo works with multiple third-party vendors for various operational needs including bus maintenance, fuel, spare parts, and other services. Currently, the invoice processing workflow involves manual data entry, which is time-consuming and error-prone.

Your task is to build an Invoice Processing System that automates the extraction of invoice data from PDF documents and provides role-based interfaces for employees and the accounts team.

Functional Requirements

1. User Management & Authentication

- Implement a secure authentication system (username and password)
- Support two user roles: Employee and Accounts Team
- Users should only access features relevant to their role

2. Invoice Submission (Employee View)

Employees should be able to:

- Upload PDF invoices (single or multiple files)
- Categorize submissions as either:
  - Vendor Payment (invoice from a third-party vendor)
  - Reimbursement (employee expense claim)
- Add optional notes or comments with each submission
- View history of their own submissions
- Track the status of each submission (Pending Review, Approved, Rejected, Paid)

3. PDF Data Extraction

The system must automatically extract the following fields from uploaded invoices:

- Vendor/Merchant name
- Invoice number
- Invoice date
- Due date (if present)
- Line items (description, quantity, unit price, total)

- Subtotal, taxes, and grand total
- Payment terms (if present)
- Bank account details (if present)

Handle edge cases:

- Invoices with varying formats and layouts
- Scanned documents vs. digitally generated PDFs
- Missing or unclear fields
- Multiple pages

4. Accounts Team Dashboard

The accounts team should be able to:

- View all submitted invoices in a paginated, sortable list
- Filter invoices by:
  - Status
  - Date range
  - Submission type (Vendor Payment / Reimbursement)
  - Submitting employee
  - Amount range
- View extracted data alongside the original PDF (side-by-side comparison)
- Edit/correct extracted fields if the system made errors
- Approve or reject invoices with mandatory comments on rejection
- Mark invoices as "Paid" after processing
- Export filtered data to CSV

5. Notifications

- Email notifications to employees when their invoice status changes (For the simplicity of the assignment, this can just be logged)

Technical Requirements

Backend

- Docker based
- RESTful API design with proper HTTP methods and status codes
- Input validation and sanitization
- Error handling with meaningful error messages
- Database schema design that supports audit trails
- File storage solution for uploaded PDFs

- Background job processing for PDF extraction (should not block the upload request)

Frontend

- Responsive design that works on desktop and tablet
- Clean, intuitive interface with appropriate loading states
- Form validation with user-friendly error messages
- Proper handling of file uploads with progress indication

Additional Features (Implement at least 3)

1. Duplicate Detection: Alert when an invoice with the same invoice number from the same vendor already exists in the system
2. Approval Workflow: Implement a two-level approval process where invoices above a certain amount require additional approval from a senior accounts member
3. Analytics Dashboard: Show metrics like: Total invoices processed (weekly/monthly), Average processing time, Breakdown by category, Top vendors by invoice volume
4. Audit Log: Track all actions performed on each invoice (who viewed, edited, approved, etc.) with timestamps
5. Bulk Operations: Allow accounts team to approve/reject multiple invoices at once
6. OCR Confidence Score: Display confidence levels for each extracted field, highlighting low-confidence extractions for manual review

Deliverables

1. Source Code

- Clean, well-organized codebase
- Meaningful commit history showing progression
- Environment configuration via environment variables

2. Documentation

- README with setup instructions
- API documentation (can use tools like Swagger/OpenAPI)
- Brief explanation of architectural decisions

3. Database

- Schema diagram or ERD
- Migration files or setup scripts

4. Demo

- Either a deployed version OR a recorded walkthrough video (5-10 minutes)
- Include sample PDF invoices used for testing