Invoice Processing Portal

Complete Project Documentation

Version: 1.0
Date: February 5, 2026
Status: Draft
Document Type: Single Source of Truth (Business & Technical)

Table of Contents

1. Executive Summary
2. Business Overview
3. Project Scope
4. Functional Requirements
5. System Architecture
6. Data Flow
7. Technology Stack
8. Database Design
9. API Specifications
10. User Interface Design
11. Security & Compliance
12. Implementation Phases
13. Testing Strategy
14. Deployment & Infrastructure
15. Glossary
16. **DEVELOPMENT GUIDE: Complete Step-by-Step Implementation**

---

1. Executive Summary

1.1 Purpose

This document outlines the complete requirements and technical specifications for building an **Invoice Processing Portal** — an automated system that reads invoices from emails, extracts key data, stores them in a database, and routes them for approval before sending to the accounting system (Sage).

1.2 Problem Statement

Currently, invoice processing is manual, time-consuming, and error-prone. Staff must:

- Manually check emails for invoices
- Open each PDF and extract data
- Enter data into spreadsheets or systems
- Send for approvals via email chains
- Manually enter approved invoices into Sage

1.3 Solution Overview

An automated portal that:

- **Automatically reads** invoices from a dedicated Outlook mailbox
- **Extracts data** from PDF invoices using OCR/AI
- **Stores** invoice data in a structured database
- **Displays** invoices on a web portal for review
- **Routes** invoices through an approval workflow
- **Integrates** with Sage accounting system upon approval

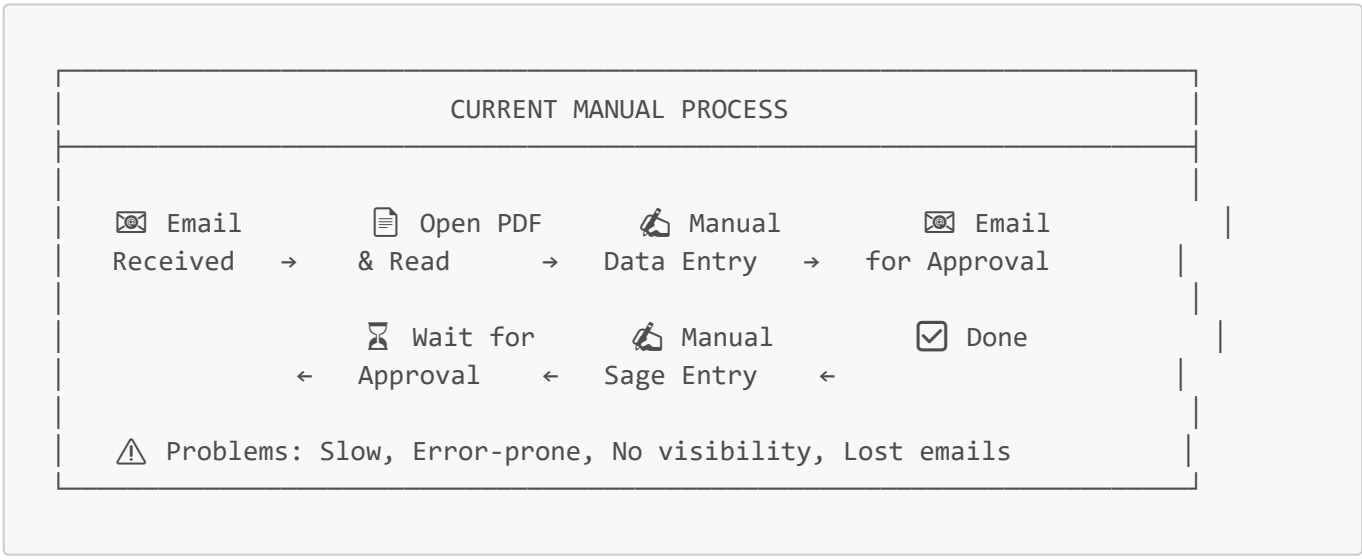
1.4 Key Benefits

Benefit	Business Impact
Time Savings	70-80% reduction in manual data entry
Accuracy	Eliminates human transcription errors
Visibility	Real-time dashboard of all invoices
Audit Trail	Complete history of approvals and changes
Faster Processing	Reduces invoice cycle time from days to hours

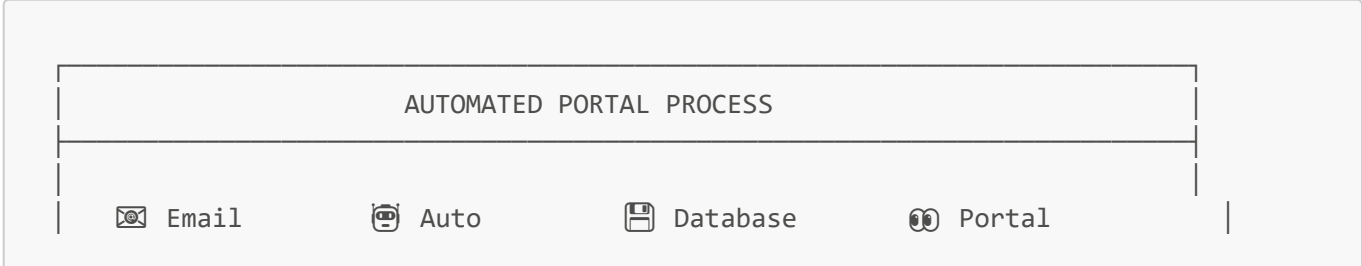
---

2. Business Overview

2.1 Current Process (As-Is)



2.2 Future Process (To-Be)



```
┌──────────────────────────────────────────────────────────────────┐
│                                                                  │
│   Received   →    Extract    →    Storage     →    Review        │
│                                                                  │
│                                                                  │
│                  🔗 Auto          ☑ Approved       ☑ Done       │
│              ←   Sage Sync  ←   in Portal    ←                   │
│                                                                  │
│                                                                  │
│   ☑ Benefits: Fast, Accurate, Full visibility, Audit trail       │
└──────────────────────────────────────────────────────────────────┘
```

2.3 Stakeholders

Role	Responsibility	Portal Access
Finance Team	Reviews and approves invoices	Full Access
Approvers/Managers	Approve/Reject invoices	Approval Access
Accounting Team	Manages Sage integration	Admin Access
IT Team	System maintenance	Admin Access
Vendors	Submit invoices via email	No Portal Access

---

3. Project Scope

3.1 In Scope ☑

#	Feature	Description
1	Email Monitoring	Monitor dedicated Outlook mailbox for incoming invoices
2	PDF Processing	Read and extract data from PDF invoice attachments
3	Data Extraction	Extract: Customer Name, Customer ID, Opportunity Number, Amount
4	Database Storage	Store invoice data with PDF file reference
5	Web Portal	Display invoices in a user-friendly interface
6	Approval Workflow	Route invoices for approval with Accept/Reject actions
7	Sage Integration	Send approved invoices to Sage accounting system
8	Audit Trail	Log all actions for compliance

3.2 Out of Scope ❌

#	Feature	Reason
1	Vendor Portal	Vendors will continue emailing invoices
2	Payment Processing	Handled by Sage
3	Purchase Order Matching	Phase 2 consideration

#	Feature	Reason
4	Multi-currency Processing	Phase 2 consideration
5	Mobile App	Web portal is mobile-responsive

3.3 Assumptions

1. Invoices will be sent **only** to the dedicated email address
2. Invoices will be in **PDF format**
3. Invoice PDFs **may have different/varying layouts** - system must handle multiple formats
4. Sage has available **APIs** for data integration
5. Users have **modern web browsers** (Chrome, Edge, Firefox)
6. Organization uses **Microsoft 365** for email

3.4 Constraints

1. Must integrate with existing Microsoft 365 infrastructure
2. Must comply with company data security policies
3. Budget and timeline as defined by project governance

---

4. Functional Requirements

4.1 Email Processing Module

FR-001: Email Monitoring

Attribute	Value
ID	FR-001
Priority	High
Description	System shall monitor a dedicated Outlook mailbox continuously
Business Rule	Check for new emails every 5 minutes (configurable)
Acceptance Criteria	New emails are detected within 5 minutes of arrival

FR-002: Email Filtering

Attribute	Value
ID	FR-002
Priority	High
Description	System shall process only emails with PDF attachments
Business Rule	Ignore emails without attachments or non-PDF attachments
Acceptance Criteria	Only PDF attachments are processed

FR-003: Attachment Download

Attribute	Value
ID	FR-003
Priority	High
Description	System shall download PDF attachments from emails
Business Rule	Store PDFs in secure file storage
Acceptance Criteria	PDFs are stored and linked to email record

4.2 PDF Processing Module

FR-004: PDF Text Extraction

Attribute	Value
ID	FR-004
Priority	High
Description	System shall extract text from PDF invoices
Business Rule	Use OCR for scanned documents
Acceptance Criteria	Text is extracted with 95%+ accuracy

FR-005: Data Field Extraction

Attribute	Value
ID	FR-005
Priority	High
Description	System shall extract specific fields from invoice
Fields	Customer Name, Customer ID, Opportunity Number, Invoice Amount
Acceptance Criteria	All required fields extracted or flagged for manual review

FR-005.1: Variable PDF Format Handling

Attribute	Value
ID	FR-005.1
Priority	High
Description	System shall handle invoices with different/varying PDF layouts
Business Rule	Use AI/ML-based extraction that adapts to different formats

Attribute	Value
Acceptance Criteria	System processes invoices regardless of layout without breaking

> ⚠ **CRITICAL REQUIREMENT: Variable PDF Format Support**
>
> Invoice PDFs may come from different vendors with completely different layouts. The system **MUST NOT break** when encountering new or unknown formats. Instead, it should:
>
> - Use intelligent AI-based extraction (not template-based)
> - Extract fields based on semantic understanding, not fixed positions
> - Flag low-confidence extractions for manual review
> - Learn and improve over time

FR-006: Invoice Validation

Attribute	Value
ID	FR-006
Priority	Medium
Description	System shall validate extracted data
Business Rule	Check for required fields, valid formats, reasonable amounts
Acceptance Criteria	Invalid invoices flagged for review

4.3 Database Module

FR-007: Invoice Record Creation

Attribute	Value
ID	FR-007
Priority	High
Description	System shall create database record for each invoice
Data Stored	Invoice ID, Customer Name, Customer ID, Opportunity Number, Amount, PDF Path, Status, Timestamps
Acceptance Criteria	Unique record created with all extracted data

FR-008: Duplicate Detection

Attribute	Value
ID	FR-008
Priority	Medium

Attribute	Value
Description	System shall detect potential duplicate invoices
Business Rule	Flag if same Invoice ID or similar details exist
Acceptance Criteria	Duplicates are flagged, not auto-processed

4.4 Portal Module

FR-009: Invoice Dashboard

Attribute	Value
ID	FR-009
Priority	High
Description	Portal shall display dashboard with invoice summary
Features	Total invoices, Pending approvals, Approved, Rejected counts
Acceptance Criteria	Real-time statistics displayed

FR-010: Invoice List View

Attribute	Value
ID	FR-010
Priority	High
Description	Portal shall display list of all invoices
Features	Search, Filter, Sort, Pagination
Acceptance Criteria	Users can find any invoice within 3 clicks

FR-011: Invoice Detail View

Attribute	Value
ID	FR-011
Priority	High
Description	Portal shall display invoice details with PDF preview
Features	All extracted fields, PDF viewer, Edit capability, Action buttons
Acceptance Criteria	Complete invoice information visible on one page

FR-012: Manual Data Entry

Attribute	Value
ID	FR-012
Priority	Medium
Description	Portal shall allow manual correction of extracted data
Business Rule	Track all changes with user and timestamp
Acceptance Criteria	Users can edit any field and save changes

4.5 Approval Workflow Module

FR-013: Approval Request

Attribute	Value
ID	FR-013
Priority	High
Description	Portal shall allow submitting invoice for approval
Business Rule	Notify approvers via email
Acceptance Criteria	Approvers receive notification and can act

FR-014: Approve/Reject Actions

Attribute	Value
ID	FR-014
Priority	High
Description	Approvers can approve or reject invoices
Business Rule	Rejection requires comments
Acceptance Criteria	Status updated, audit trail created

FR-015: Approval Notifications

Attribute	Value
ID	FR-015
Priority	Medium
Description	System shall send email notifications for approval actions
Triggers	New approval request, Approved, Rejected
Acceptance Criteria	Emails sent within 1 minute of action

4.6 Sage Integration Module

FR-016: Sage Data Export

Attribute	Value
ID	FR-016
Priority	High
Description	System shall send approved invoices to Sage
Data Sent	All invoice fields as per Sage API requirements
Acceptance Criteria	Invoice created in Sage successfully

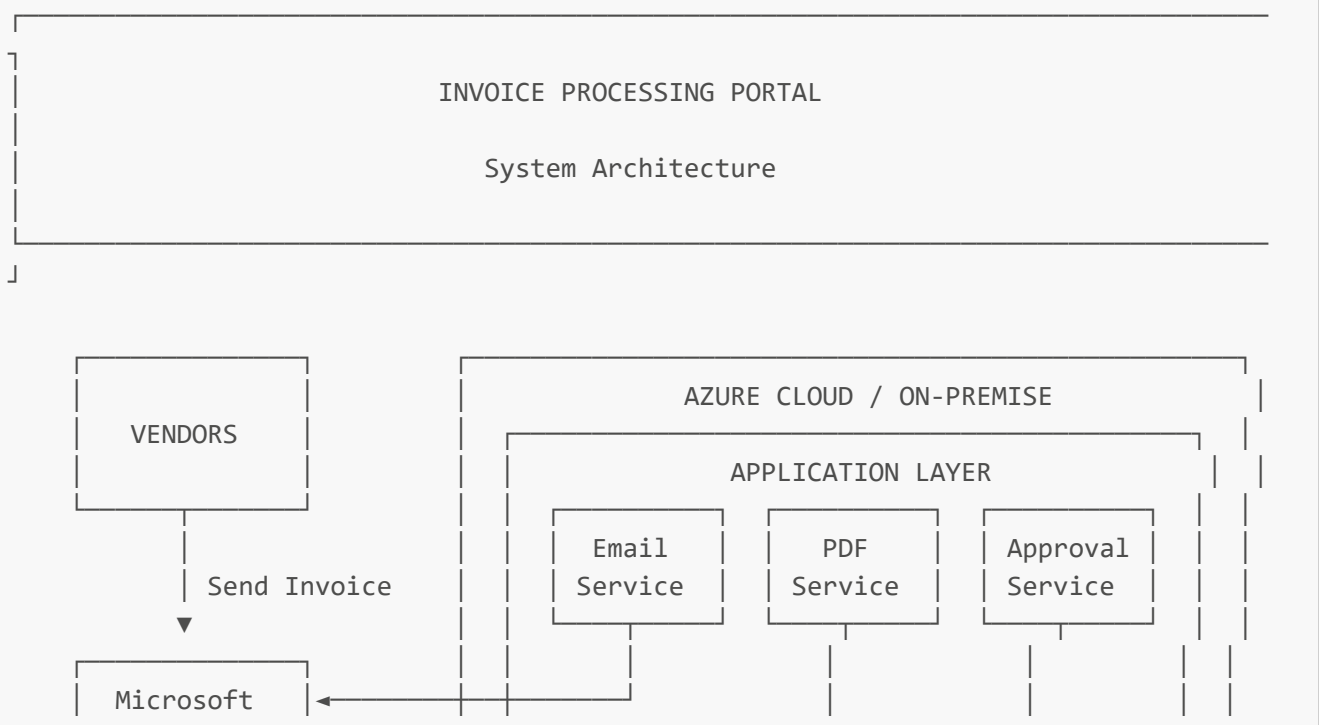
FR-017: Sage Sync Status

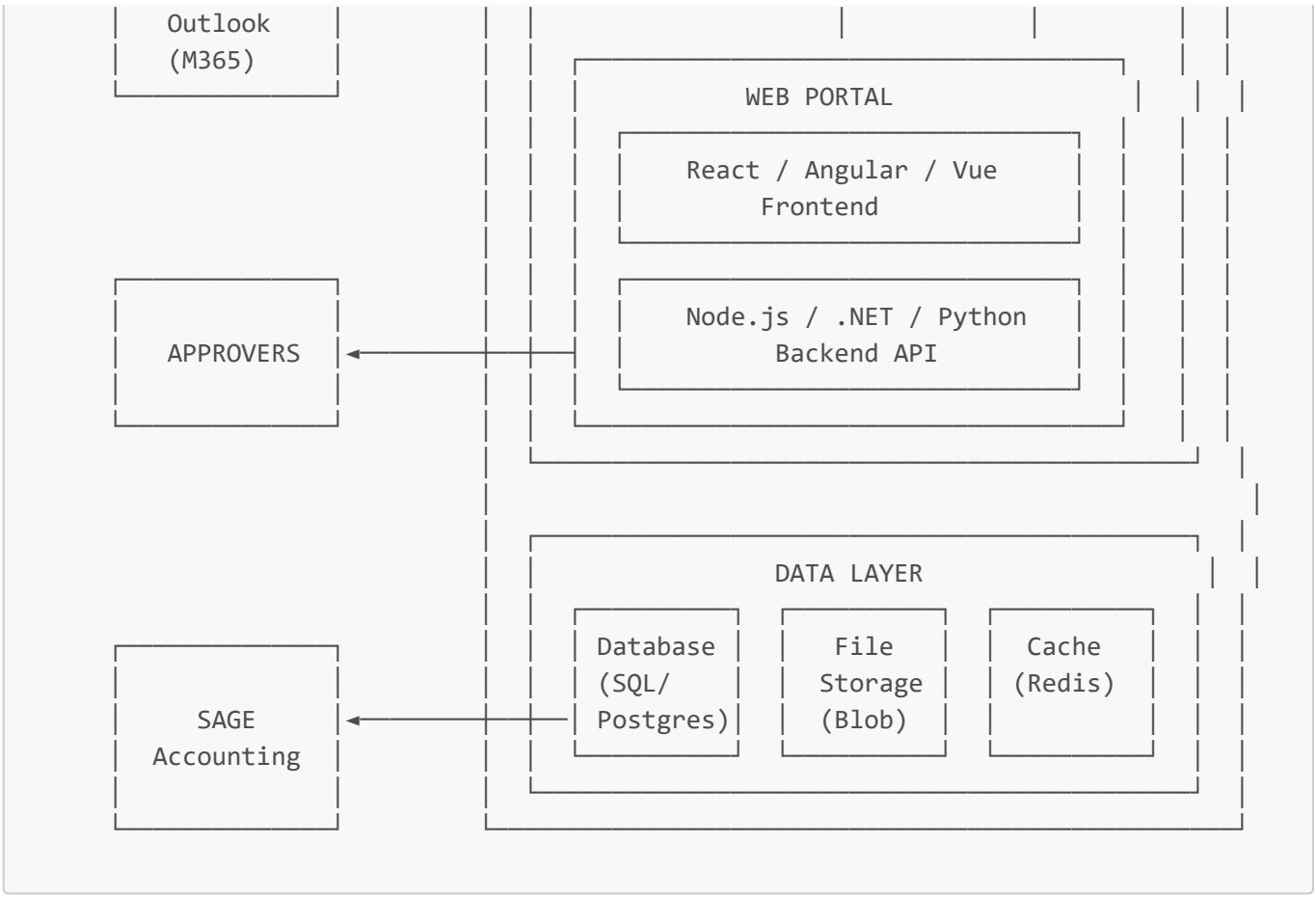
Attribute	Value
ID	FR-017
Priority	Medium
Description	System shall track Sage sync status
Statuses	Pending, Synced, Failed
Acceptance Criteria	Status visible in portal, failures alerted

---

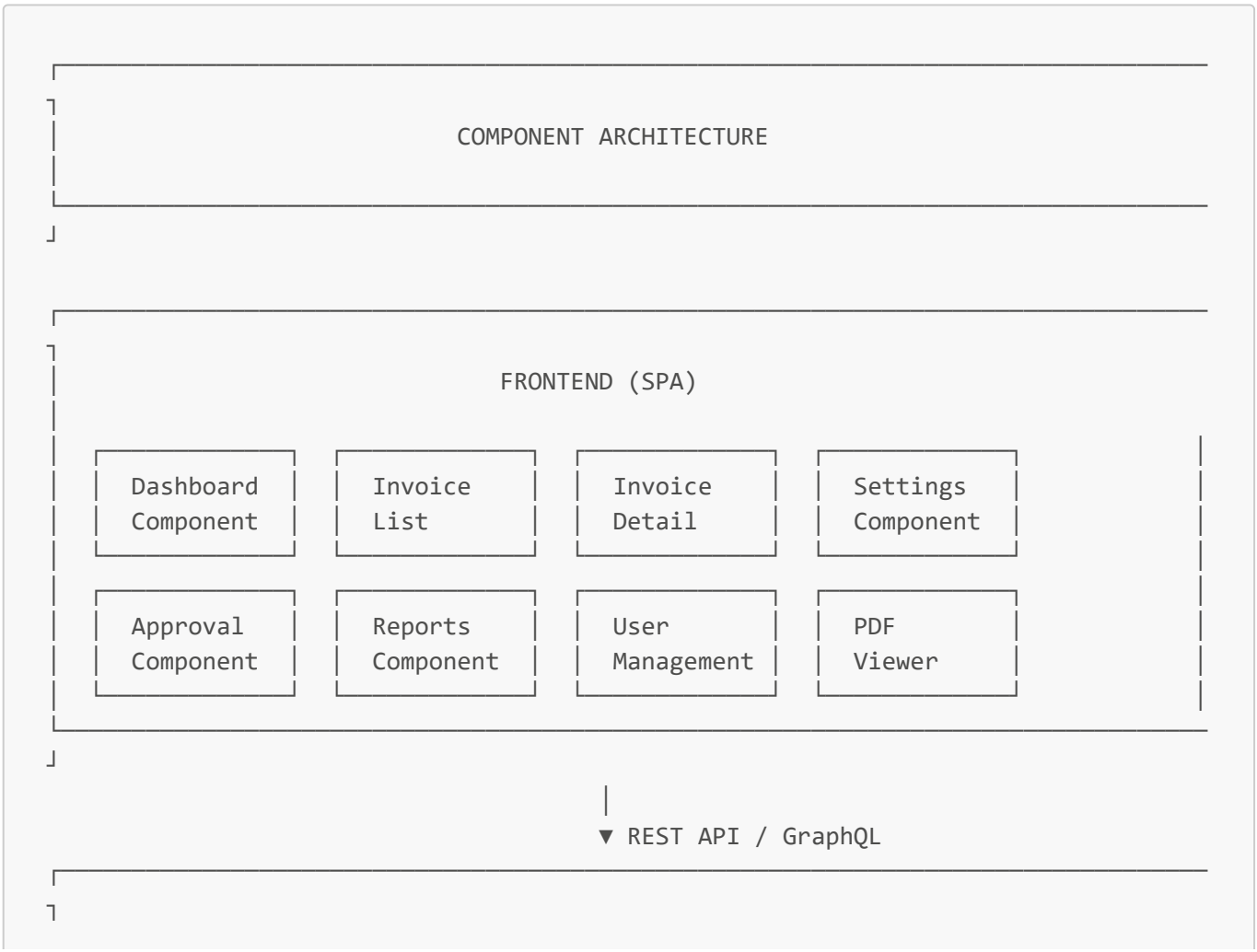
5. System Architecture

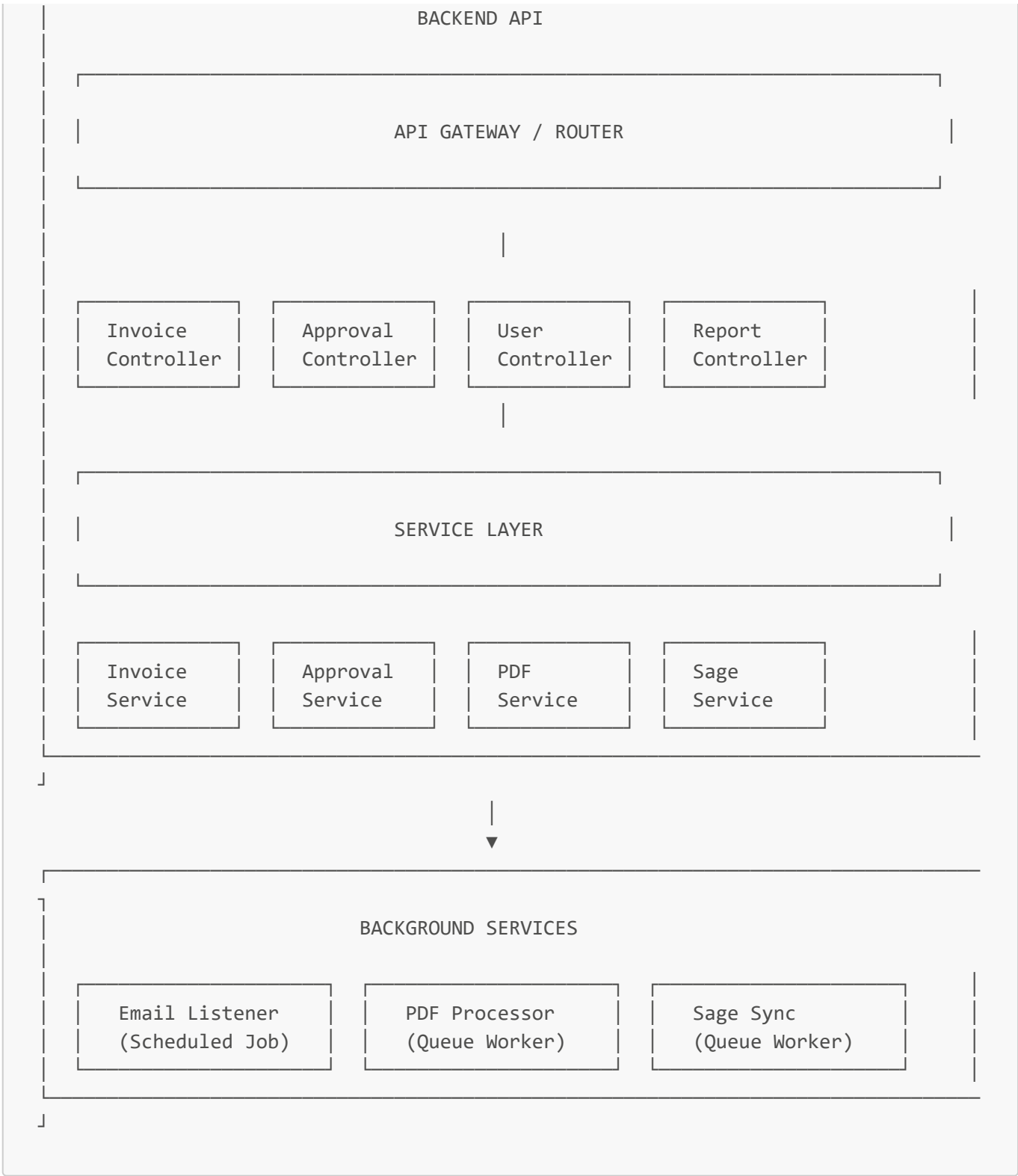
5.1 High-Level Architecture



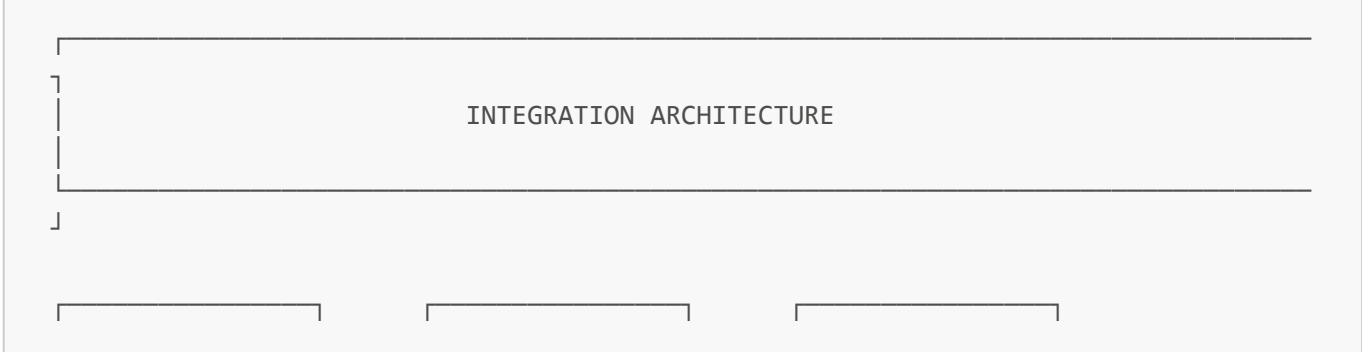


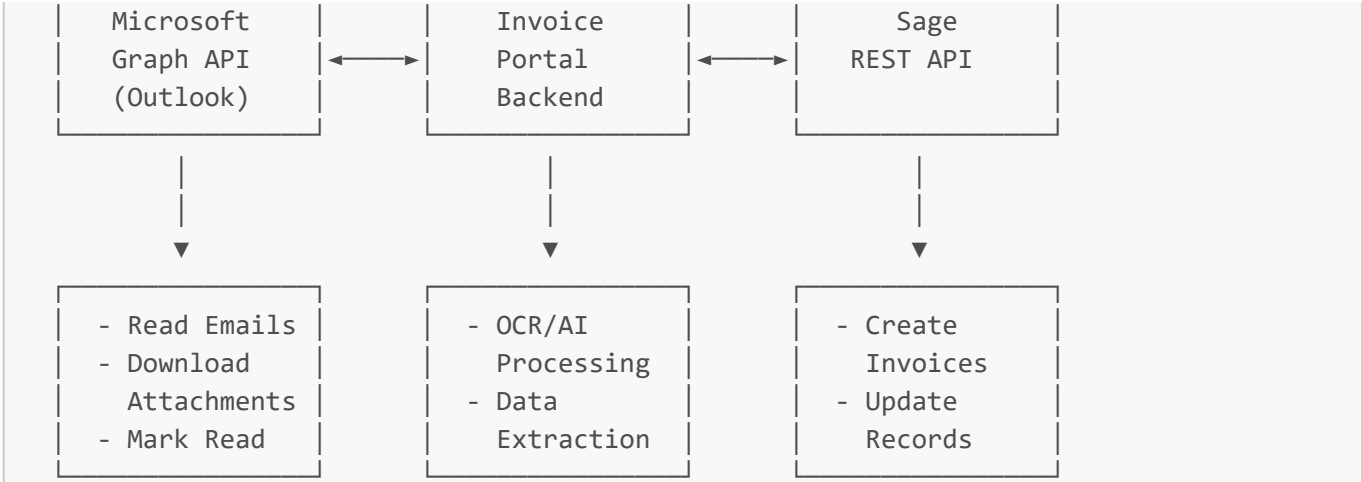
5.2 Component Architecture





5.3 Integration Architecture

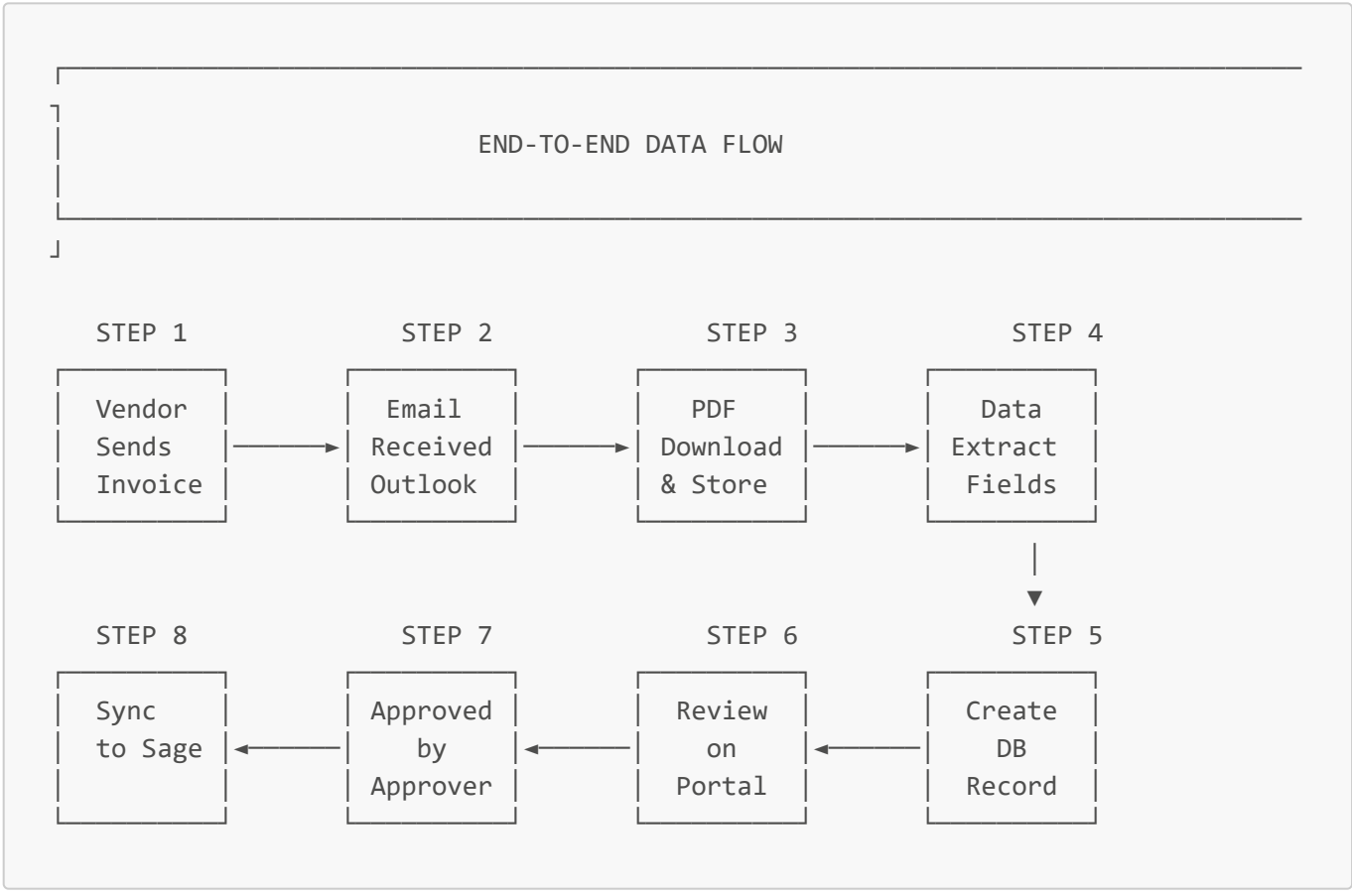




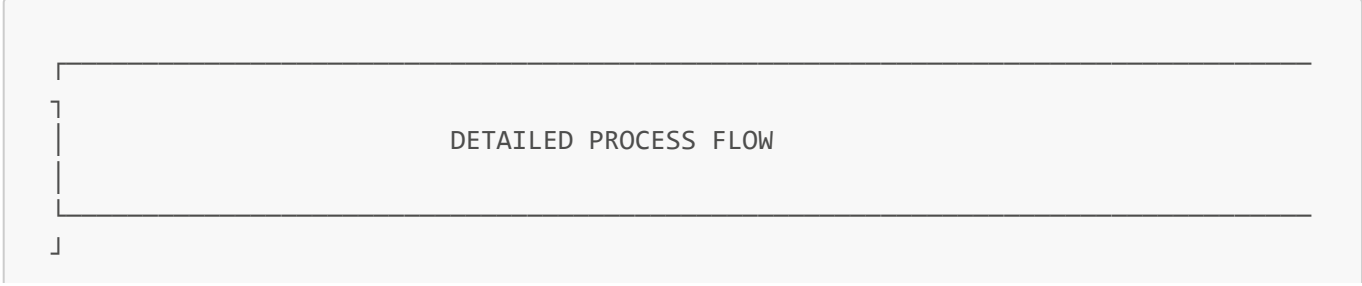
---

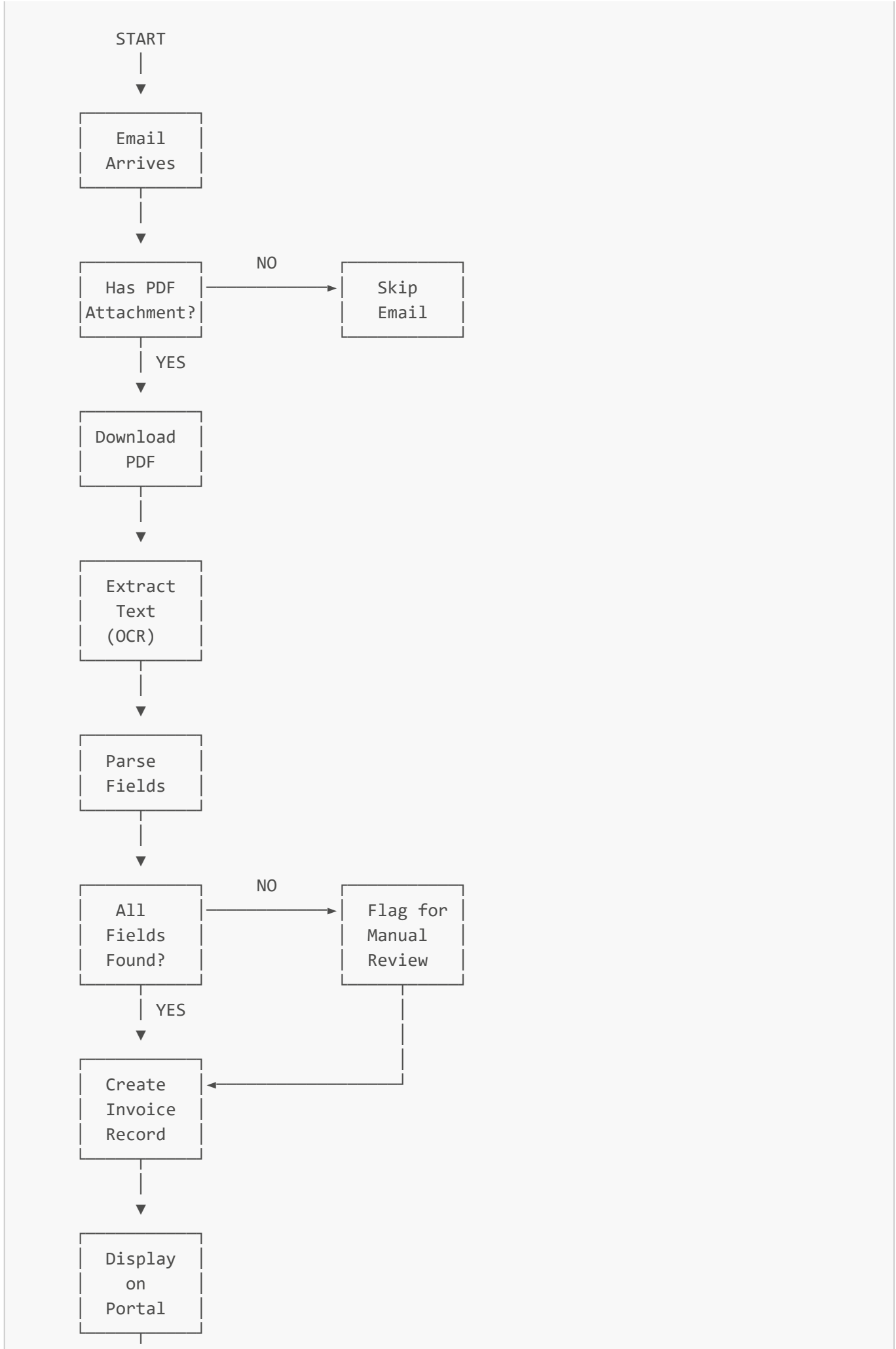
6. Data Flow

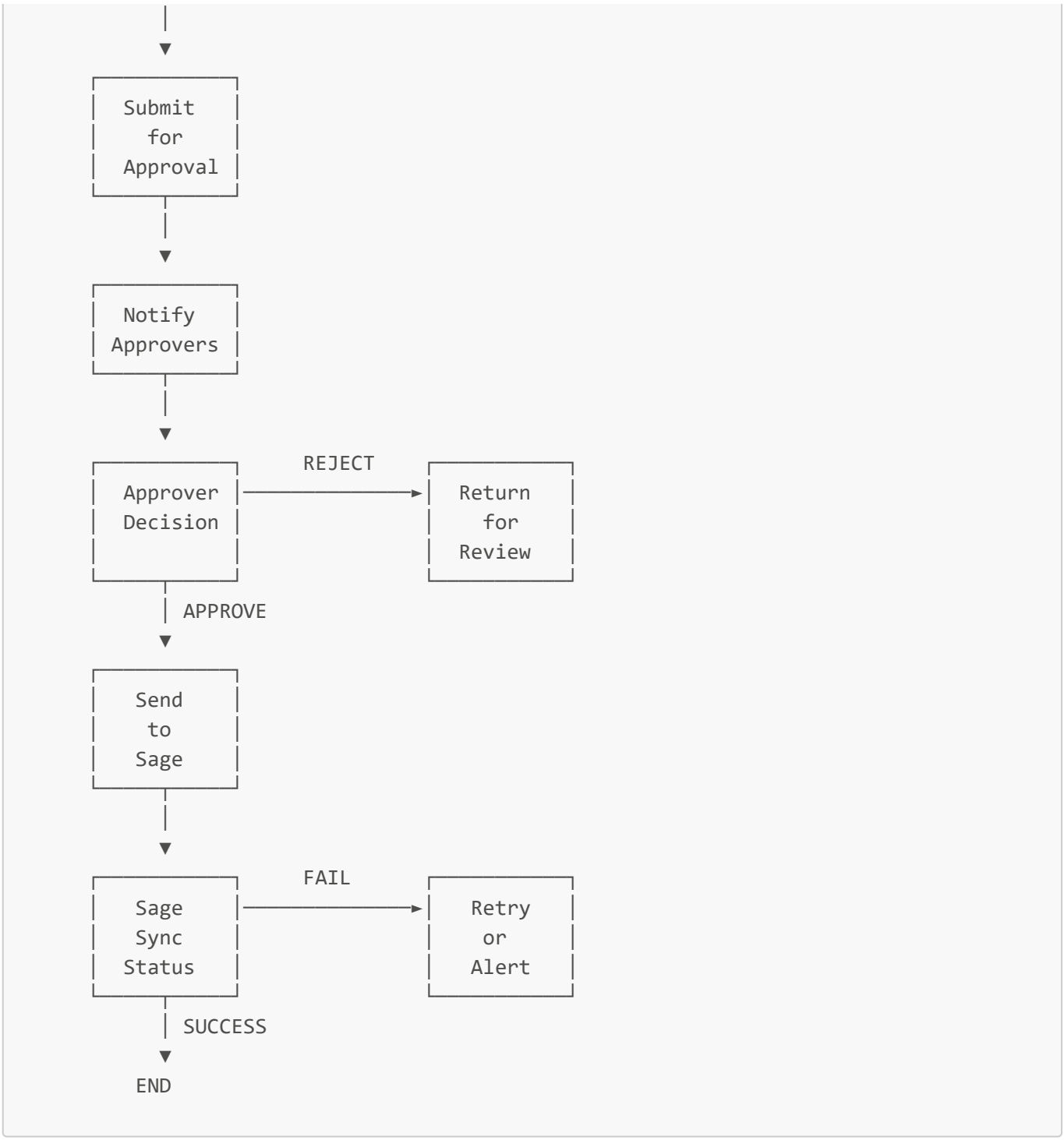
6.1 End-to-End Process Flow



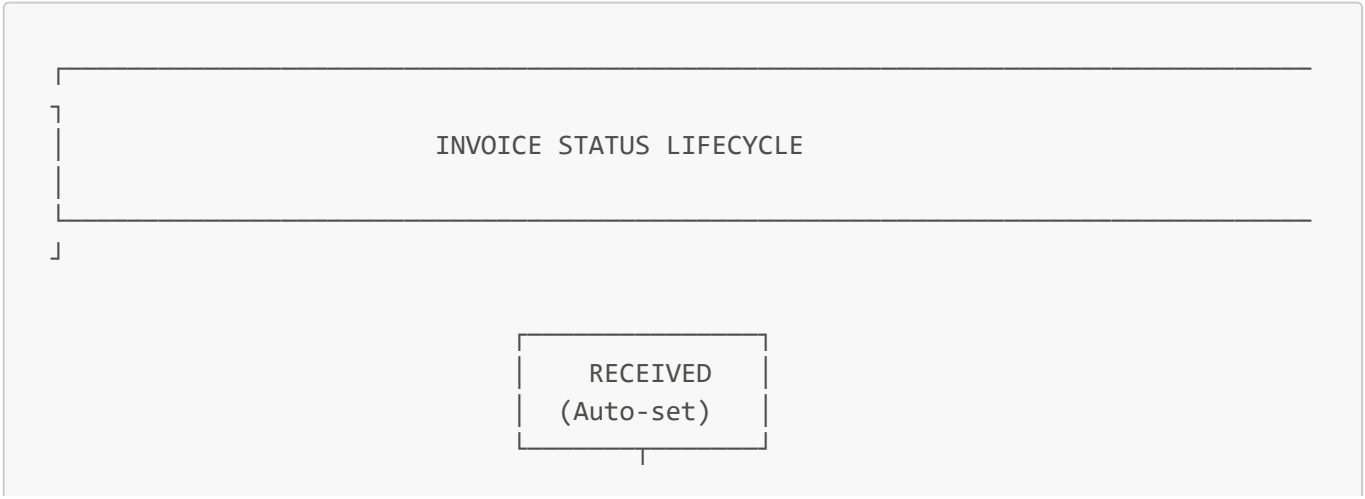
6.2 Detailed Process Flow Diagram

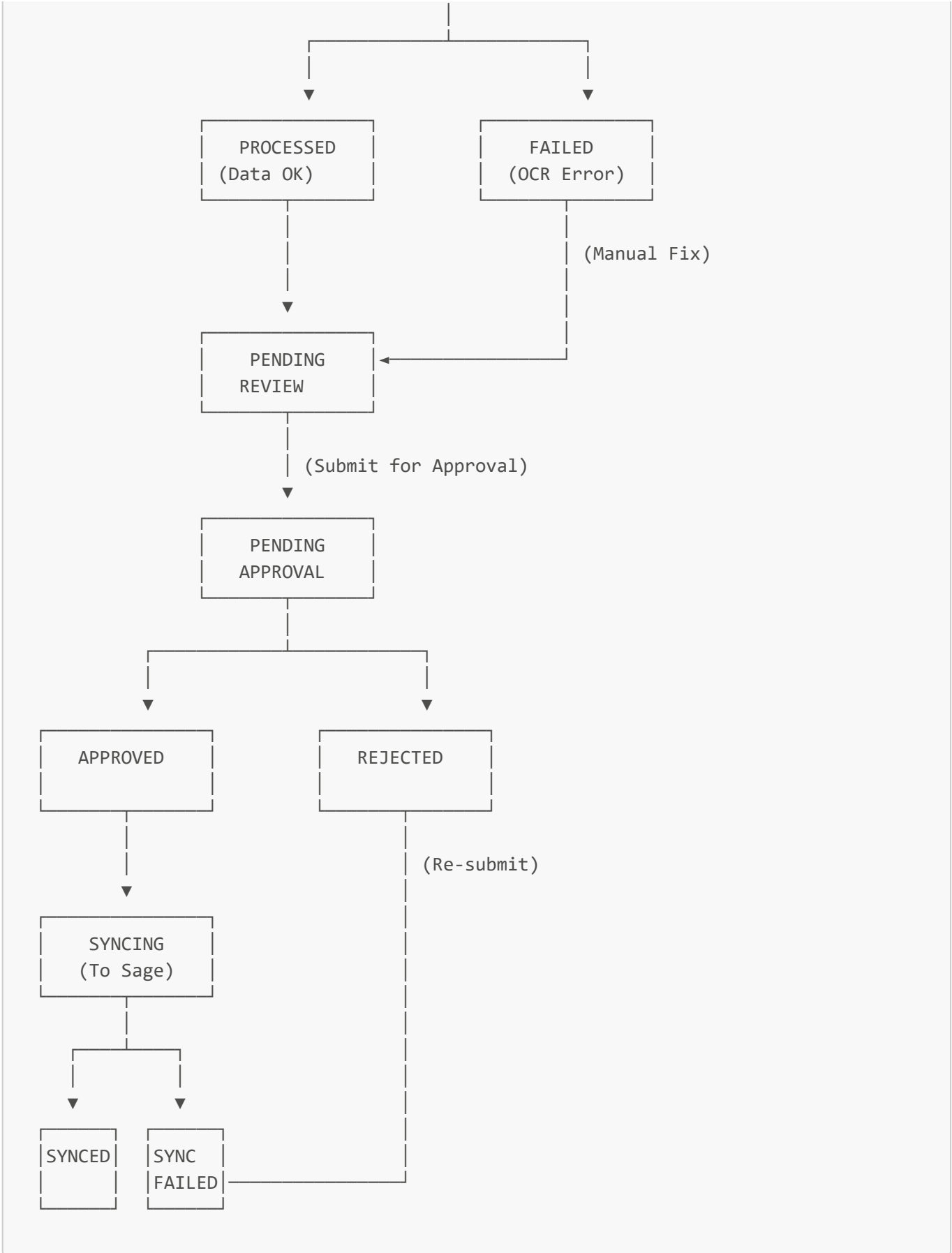



START
Email Arrives
Has PDF Attachment?
NO
Skip Email
YES
Download PDF
Extract Text (OCR)
Parse Fields
All Fields Found?
NO
Flag for Manual Review
YES
Create Invoice Record
Display on Portal



6.3 Invoice Status State Diagram





7. Technology Stack

7.1 Recommended Technology Options

Option A: Microsoft/.NET Stack (Recommended for Microsoft Shops)

Layer	Technology	Reason
Frontend	React + TypeScript	Modern, performant, large ecosystem
Backend	ASP.NET Core 8	Enterprise-grade, excellent M365 integration
Database	SQL Server / Azure SQL	Native Microsoft integration
File Storage	Azure Blob Storage	Scalable, secure
Authentication	Azure AD / Entra ID	SSO with existing Microsoft accounts
Email	Microsoft Graph API	Native Outlook integration
PDF Processing	Azure Document Intelligence	AI-powered extraction
Queue	Azure Service Bus	Reliable message processing
Hosting	Azure App Service	PaaS, easy deployment

Option B: Java Spring Boot Stack

Layer	Technology	Reason
Frontend	React + TypeScript / Angular	Modern SPA frameworks
Backend	Java 21 + Spring Boot 3.x	Enterprise standard, robust ecosystem
Database	PostgreSQL / MySQL	Open source, reliable
File Storage	AWS S3 / Azure Blob / MinIO	Flexible cloud storage
Authentication	Spring Security + OAuth2	Industry-standard security
Email	Microsoft Graph SDK for Java	Outlook integration
PDF Processing	Apache PDFBox + Tesseract OCR / AWS Textract	Open source + cloud AI
Queue	Apache Kafka / RabbitMQ / AWS SQS	High-throughput messaging
Hosting	Docker + Kubernetes / AWS ECS	Container orchestration

Option C: Python Stack

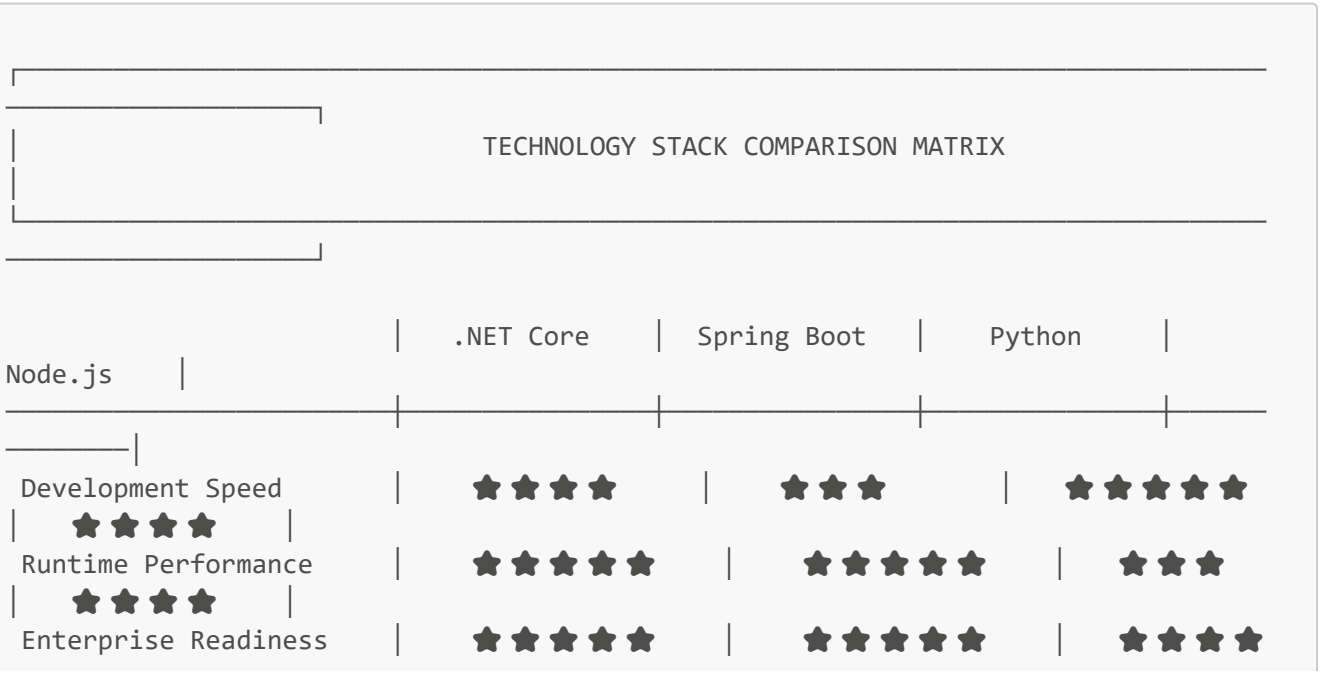
Layer	Technology	Reason
Frontend	React + TypeScript / Vue.js	Modern SPA frameworks
Backend	Python 3.12 + FastAPI / Django	Rapid development, excellent AI/ML libraries
Database	PostgreSQL	Robust, Python-friendly

Layer	Technology	Reason
File Storage	AWS S3 / Azure Blob	Cloud storage
Authentication	OAuth2 + JWT (Authlib)	Flexible auth
Email	Microsoft Graph SDK for Python	Outlook integration
PDF Processing	PyMuPDF + pytesseract / Azure Document Intelligence	Native Python libraries
Queue	Celery + Redis / RabbitMQ	Async task processing
Hosting	Docker + Kubernetes / AWS Lambda	Flexible deployment

Option D: Open Source Full Stack

Layer	Technology	Reason
Frontend	React + TypeScript	Modern, performant
Backend	Node.js + Express / NestJS	JavaScript full-stack
Database	PostgreSQL	Robust, free
File Storage	AWS S3 / MinIO	Cost-effective
Authentication	Auth0 / Keycloak	Flexible identity
Email	Microsoft Graph API	Required for Outlook
PDF Processing	Tesseract + Custom ML	Open source OCR
Queue	RabbitMQ / Redis	Message processing
Hosting	Docker + Kubernetes	Portable, scalable

7.2 Technology Stack Comparison Matrix



```
| ★★★ |
M365 Integration | ★★★★★ | ★★★★ | ★★★★
| ★★★★ |
AI/ML Libraries | ★★★ | ★★★ | ★★★★★
| ★★★ |
PDF Processing | ★★★★ | ★★★★ | ★★★★★
| ★★★ |
Developer Availability | ★★★★ | ★★★★★ |
★★★★★ | ★★★★★ |
Learning Curve | Medium | Medium-High | Low-Medium | Low
|
Community Support | ★★★★ | ★★★★★ |
★★★★★ | ★★★★★ |
Long-term Maintenance | ★★★★★ | ★★★★★ | ★★★★
| ★★★★ |
──────────────────────────┼──────────────┼──────────────┼─────────────┼──────
──────┤
```

7.3 Detailed Stack Analysis

7.3.1 Java Spring Boot - Detailed Analysis

```
┌──────────────────────────────────────────────────────────────────────────────
┐
│                        JAVA SPRING BOOT STACK ANALYSIS
│
└──────────────────────────────────────────────────────────────────────────────
┘
```

Architecture:

```
┌──────────────────────────────────────────────────────────────────────────────
┐
│
│
│
│    ┌──────────────┐     ┌──────────────┐     ┌──────────────┐
│
│    │   React/     │     │  Spring Boot │     │  PostgreSQL  │
│
│    │   Angular    │────▶│  REST API    │────▶│  Database    │
│
│    │   Frontend   │     │  (Java 21)   │     │              │
│
│    └──────────────┘     └──────┬───────┘     └──────────────┘
│
│                                │
│
│                                ▼
│
│    ┌──────────────┐     ┌──────────────┐     ┌──────────────┐
│
│    │   Apache     │     │   Spring     │     │  Tesseract/  │
│
│    │   Kafka      │◀────│   Scheduler  │────▶│ AWS Textract │
│
│    │   (Queue)    │     │  (Email Job) │     │  (OCR)       │
│
│    └──────────────┘     └──────────────┘     └──────────────┘
│
```

```
    │
    │
    └──────────────────────────────────────────────────────────────────
   ┘
```

Pros:

Advantage	Description
☑ **Performance**	Excellent runtime performance, JVM optimization
☑ **Enterprise Standard**	Widely adopted in large enterprises, banks, financial institutions
☑ **Strong Typing**	Compile-time error detection, IDE support
☑ **Mature Ecosystem**	Spring Data, Spring Security, Spring Cloud
☑ **Scalability**	Proven for high-load systems
☑ **Talent Pool**	Large pool of experienced Java developers
☑ **Long-term Support**	LTS versions, stable API changes

Cons:

Disadvantage	Description
❌ **Verbose Code**	More boilerplate compared to Python
❌ **Slower Development**	Takes longer to implement features
❌ **Memory Footprint**	JVM requires more memory
❌ **Startup Time**	Slower cold start (mitigated by GraalVM native)
❌ **AI/ML Integration**	Fewer native ML libraries than Python

Key Dependencies (pom.xml):

```xml
<dependencies>
    <!-- Core -->
    <dependency>spring-boot-starter-web</dependency>
    <dependency>spring-boot-starter-data-jpa</dependency>
    <dependency>spring-boot-starter-security</dependency>
    <dependency>spring-boot-starter-oauth2-client</dependency>

    <!-- Email -->
    <dependency>microsoft-graph</dependency>
    <dependency>azure-identity</dependency>

    <!-- PDF Processing -->
    <dependency>pdfbox</dependency>
    <dependency>tess4j</dependency> <!-- Tesseract wrapper -->
```

```
    <!-- Queue -->
    <dependency>spring-kafka</dependency>

    <!-- Database -->
    <dependency>postgresql</dependency>
</dependencies>
```

Implementation Complexity: MEDIUM-HIGH

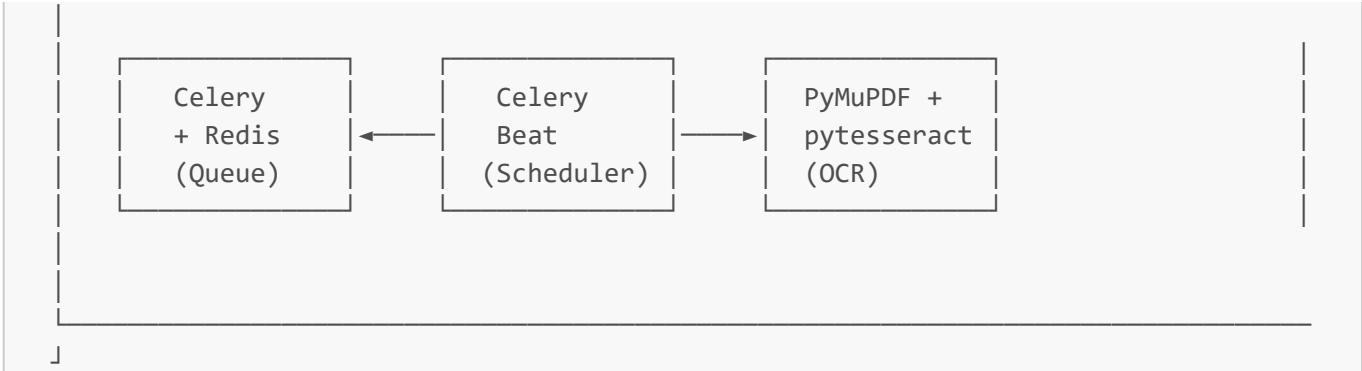
Component	Complexity	Effort (Hours)	Notes
Project Setup	Medium	12	Spring Initializr, dependencies
REST API	Low	16	Spring MVC, annotations
Database Layer	Low	12	Spring Data JPA
Email Integration	Medium	24	Microsoft Graph SDK
PDF Processing	High	48	PDFBox + Tesseract setup
Authentication	Medium	16	Spring Security OAuth2
Background Jobs	Medium	16	Spring Scheduler + Kafka
Total Additional		**+40 hours**	vs baseline estimate

7.3.2 Python Stack - Detailed Analysis

```
┌─────────────────────────────────────────────────────────────────────────────┐
│                                                                             │
│                          PYTHON STACK ANALYSIS                              │
│                                                                             │
└─────────────────────────────────────────────────────────────────────────────┘
```

Architecture:

```
┌─────────────────────────────────────────────────────────────────────────────┐
│                                                                             │
│                                                                             │
│   ┌──────────────┐      ┌──────────────┐      ┌──────────────┐              │
│   │   React/     │      │   FastAPI    │      │  PostgreSQL  │              │
│   │   Vue.js     │─────►│   (Python    │─────►│  Database    │              │
│   │   Frontend   │      │    3.12)     │      │              │              │
│   └──────────────┘      └──────┬───────┘      └──────────────┘              │
│                                │                                            │
│                                │                                            │
│                                ▼                                            │
```



Pros:

Advantage	Description
☑ **Rapid Development**	30-40% faster development than Java
☑ **AI/ML Best-in-Class**	NumPy, pandas, scikit-learn, transformers
☑ **PDF Processing**	Excellent libraries: PyMuPDF, pdfplumber, pdf2image
☑ **OCR Integration**	Native pytesseract, easy AI model integration
☑ **Concise Code**	Less boilerplate, readable syntax
☑ **FastAPI Performance**	Async support, comparable to Node.js
☑ **Data Processing**	Best for invoice data extraction and transformation

Cons:

Disadvantage	Description
❌ **Runtime Performance**	Slower than Java/.NET for CPU-intensive tasks
❌ **Concurrency Model**	GIL limitation (mitigated by async/multiprocessing)
❌ **Type Safety**	Dynamic typing can lead to runtime errors
❌ **Enterprise Perception**	Some enterprises prefer Java/.NET
❌ **Dependency Management**	Virtual environments can be complex

Key Dependencies (requirements.txt):

```
# Core Framework
fastapi==0.109.0
uvicorn[standard]==0.27.0
pydantic==2.5.0
sqlalchemy==2.0.25
alembic==1.13.0

# Email Integration
msgraph-sdk==1.0.0
```

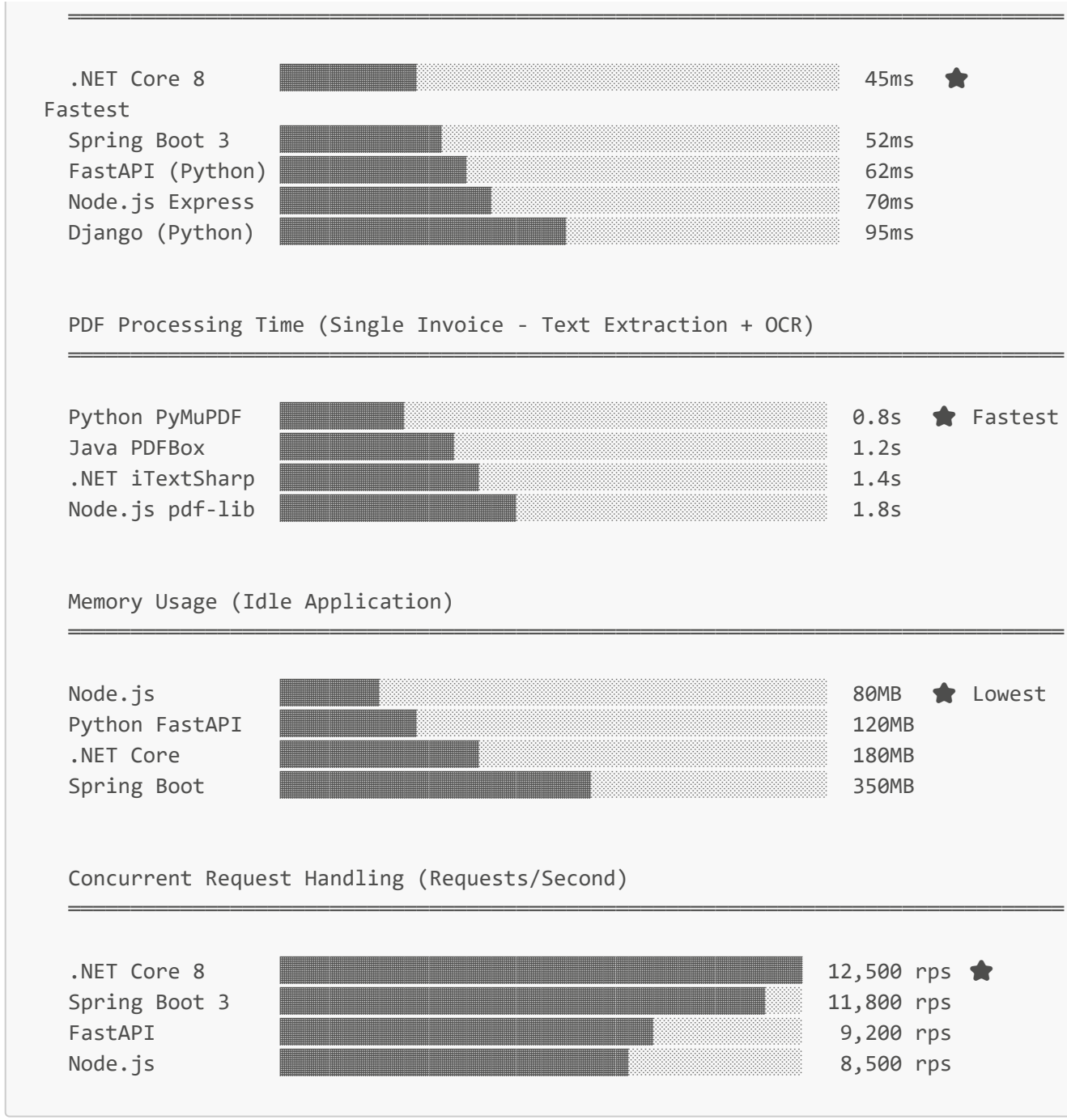
```
azure-identity==1.15.0

# PDF Processing (EXCELLENT for this project)
PyMuPDF==1.23.0          # Fast PDF text extraction
pdfplumber==0.10.0       # Table extraction
pytesseract==0.3.10      # OCR
pdf2image==1.16.0        # PDF to image conversion
Pillow==10.2.0           # Image processing

# AI/ML for Smart Extraction
transformers==4.36.0     # For NLP-based field extraction
spacy==3.7.0             # Named entity recognition
scikit-learn==1.4.0      # ML models

# Async Task Queue
celery==5.3.0
redis==5.0.0

# Database
psycopg2-binary==2.9.9
asyncpg==0.29.0
```

Implementation Complexity: LOW-MEDIUM

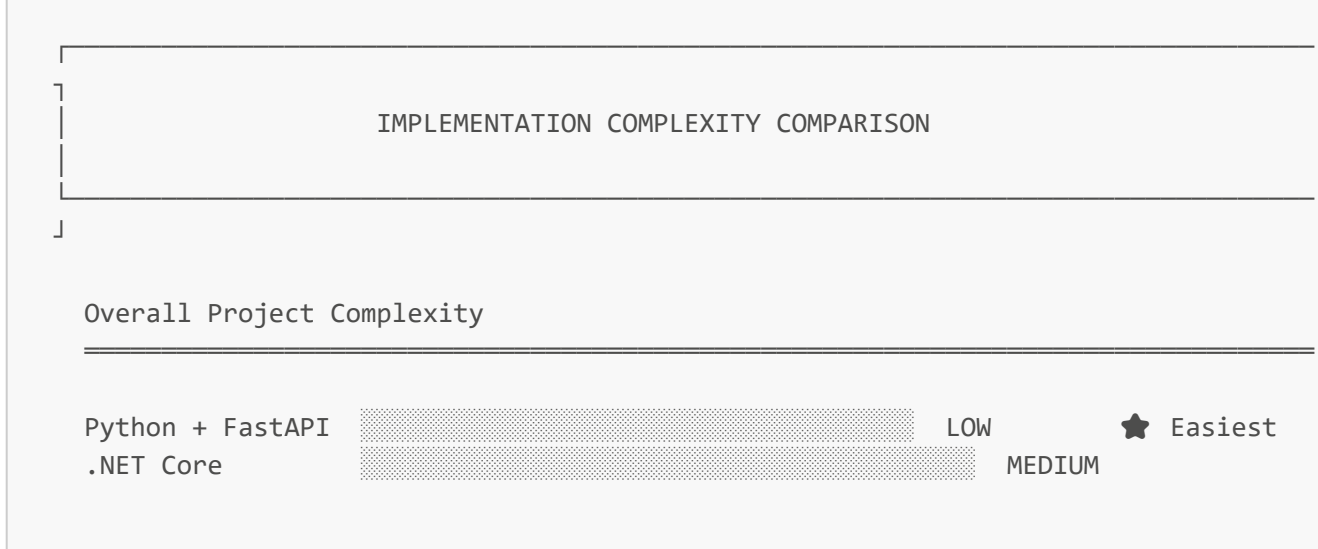
Component	Complexity	Effort (Hours)	Notes
Project Setup	Low	8	FastAPI scaffold
REST API	Low	12	FastAPI auto-docs
Database Layer	Low	10	SQLAlchemy ORM
Email Integration	Medium	20	Microsoft Graph SDK
PDF Processing	**Low**	**32**	**Excellent Python libraries**
Authentication	Medium	14	OAuth2 + JWT
Background Jobs	Low	12	Celery + Redis
Total Savings		**-60 hours**	vs baseline estimate

7.4 Performance Comparison

```
┌─────────────────────────────────────────────────────────────────────────┐
│                        PERFORMANCE BENCHMARKS                           │
└─────────────────────────────────────────────────────────────────────────┘

  API Response Time (GET /invoices - 1000 records)
```



7.5 Implementation Complexity by Stack



```
    Node.js            ░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░  MEDIUM
    Spring Boot        ░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░  MEDIUM-HIGH
```

Component	.NET Core	Spring Boot	Python FastAPI	Node.js
Project Setup	Medium	Medium-High	Low	Low
REST API	Low	Low	Very Low	Low
Database ORM	Medium	Medium	Low	Medium
Authentication	Medium	Medium-High	Medium	Medium
Email (Graph API)	Low	Medium	Medium	Medium
PDF Processing	Medium	Medium-High	**Very Low**	High
OCR Integration	Medium	High	**Very Low**	High
AI/ML Fields Extraction	High	High	**Low**	High
Background Jobs	Medium	Medium	Low	Medium
Sage Integration	Medium	Medium	Medium	Medium

7.6 Effort Estimation by Stack

```
┌─────────────────────────────────────────────────────────────────────────────┐
│                    ESTIMATED HOURS BY TECHNOLOGY STACK                      │
│                                                                             │
└─────────────────────────────────────────────────────────────────────────────┘

                          Base      PDF Module    Total      vs Baseline
    ════════════════════════════════════════════════════════════════════════

    Python + FastAPI      624 hrs   120 hrs       740 hrs    -60 hrs (8% less)
    .NET Core (Baseline)  624 hrs   176 hrs       800 hrs    Baseline
    Node.js + Express     624 hrs   200 hrs       824 hrs    +24 hrs (3% more)
    Java Spring Boot      624 hrs   216 hrs       840 hrs    +40 hrs (5% more)
```

Stack	Total Hours	Duration	Est. Cost	Best For
Python + FastAPI	**740 hrs**	**14-16 weeks**	**$55,000 - $75,000**	**This project (PDF/AI)**
.NET Core	800 hrs	16-18 weeks	$60,000 - $85,000	Microsoft shops
Node.js	824 hrs	16-18 weeks	$62,000 - $88,000	JavaScript teams
Spring Boot	840 hrs	17-19 weeks	$65,000 - $90,000	Enterprise/Banking

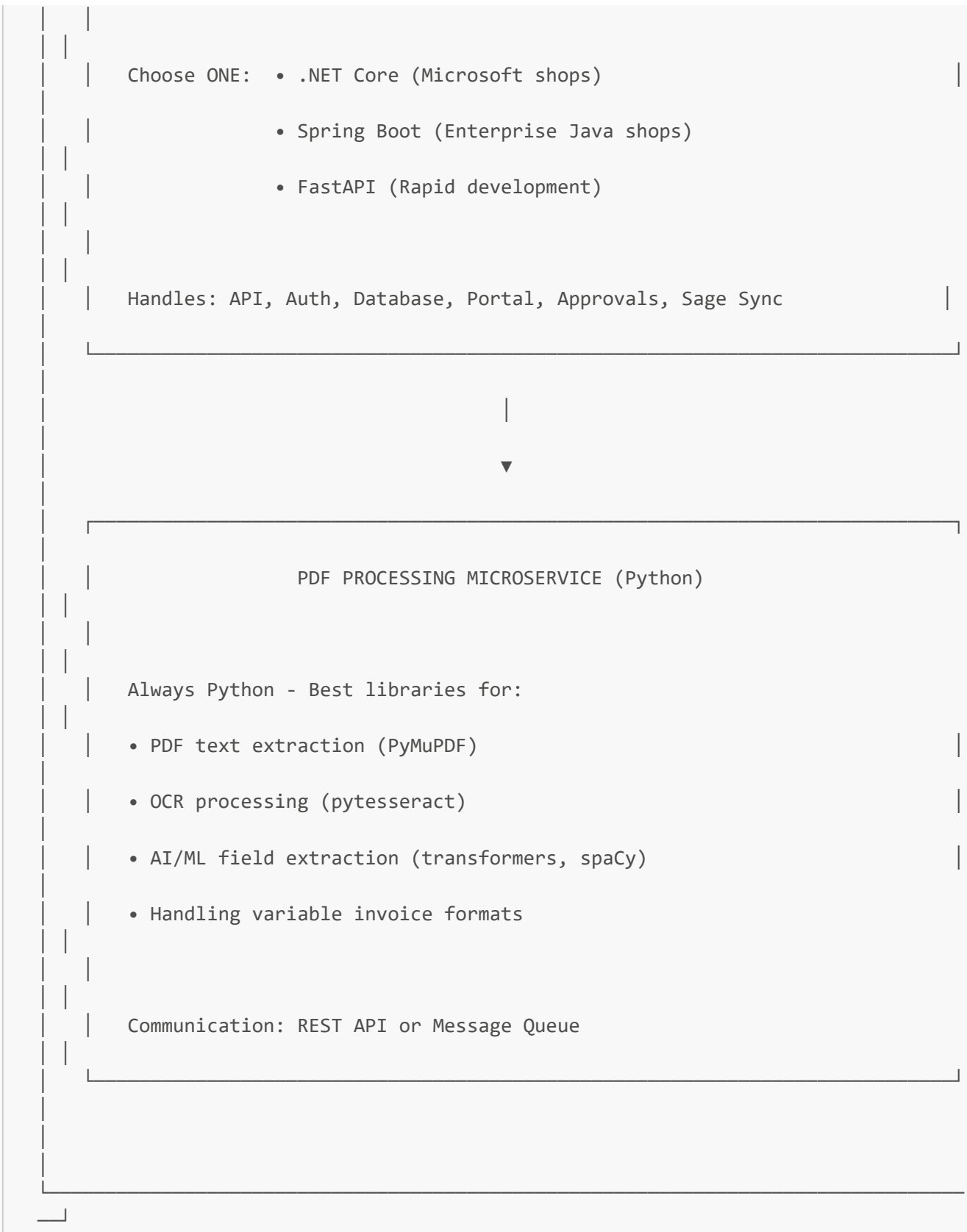
7.7 Recommendation for This Project

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                          STACK RECOMMENDATION                                │
└──────────────────────────────────────────────────────────────────────────────┘

  FOR THIS INVOICE PROCESSING PROJECT:

  🥇 RECOMMENDED: Python + FastAPI
  ──────────────────────────────────────────────────────────────────────────────
     • Best PDF processing libraries (PyMuPDF, pdfplumber)
     • Superior AI/ML integration for intelligent field extraction
     • 8% less development effort
     • Fastest time-to-market
     • Excellent for handling variable invoice formats

  🥈 ALTERNATIVE 1: .NET Core
  ──────────────────────────────────────────────────────────────────────────────
     • Best choice if organization is Microsoft-centric
     • Excellent Azure integration
     • Best runtime performance
     • Strong enterprise support

  🥉 ALTERNATIVE 2: Java Spring Boot
  ──────────────────────────────────────────────────────────────────────────────
     • Best choice for large enterprises with Java expertise
     • Proven scalability
     • Long-term maintainability
     • Higher initial investment but stable long-term
```

7.8 Hybrid Architecture Option (Recommended)

For optimal results, consider a **hybrid approach**:

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                   HYBRID ARCHITECTURE (BEST OF BOTH WORLDS)                  │
└──────────────────────────────────────────────────────────────────────────────┘

┌──────────────────────────────────────────────────────────────────────────────┐
┐
│
│
│    ┌──────────────────────────────────────────────────────────────────────┐
│
│    │                      MAIN APPLICATION
│ │
```



7.9 Final Technology Decision Matrix

If Your Organization...	Recommended Stack	Reason
Uses Microsoft 365 & Azure heavily	.NET Core	Native integration
Has strong Java expertise	Spring Boot	Team familiarity
Wants fastest development	Python FastAPI	8% faster, best PDF libs
Needs best PDF/AI processing	Python FastAPI	Superior libraries
Has mixed expertise	Hybrid (Any + Python PDF service)	Best of both
Is a startup/small team	Python FastAPI	Rapid development
Is a large enterprise/bank	Spring Boot	Enterprise standard
Prioritizes raw performance	.NET Core	Fastest runtime

7.2 Technology Stack Comparison

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                    TECHNOLOGY DECISION MATRIX                                │
└──────────────────────────────────────────────────────────────────────────────┘

                    │ Microsoft Stack │ Open Source Stack │
────────────────────┼─────────────────┼───────────────────┤
 Initial Cost       │     Medium      │       Low         │
 License Cost       │      High       │       Free        │
 Microsoft Integration │   Excellent   │      Good         │
 Team Expertise     │   Depends       │    Depends        │
 Enterprise Support │    Excellent    │     Community     │
 Scalability        │    Excellent    │     Excellent     │
 Vendor Lock-in     │      High       │       Low         │
────────────────────┼─────────────────┼───────────────────┤
```

7.3 Selected Stack (This Document Assumes)

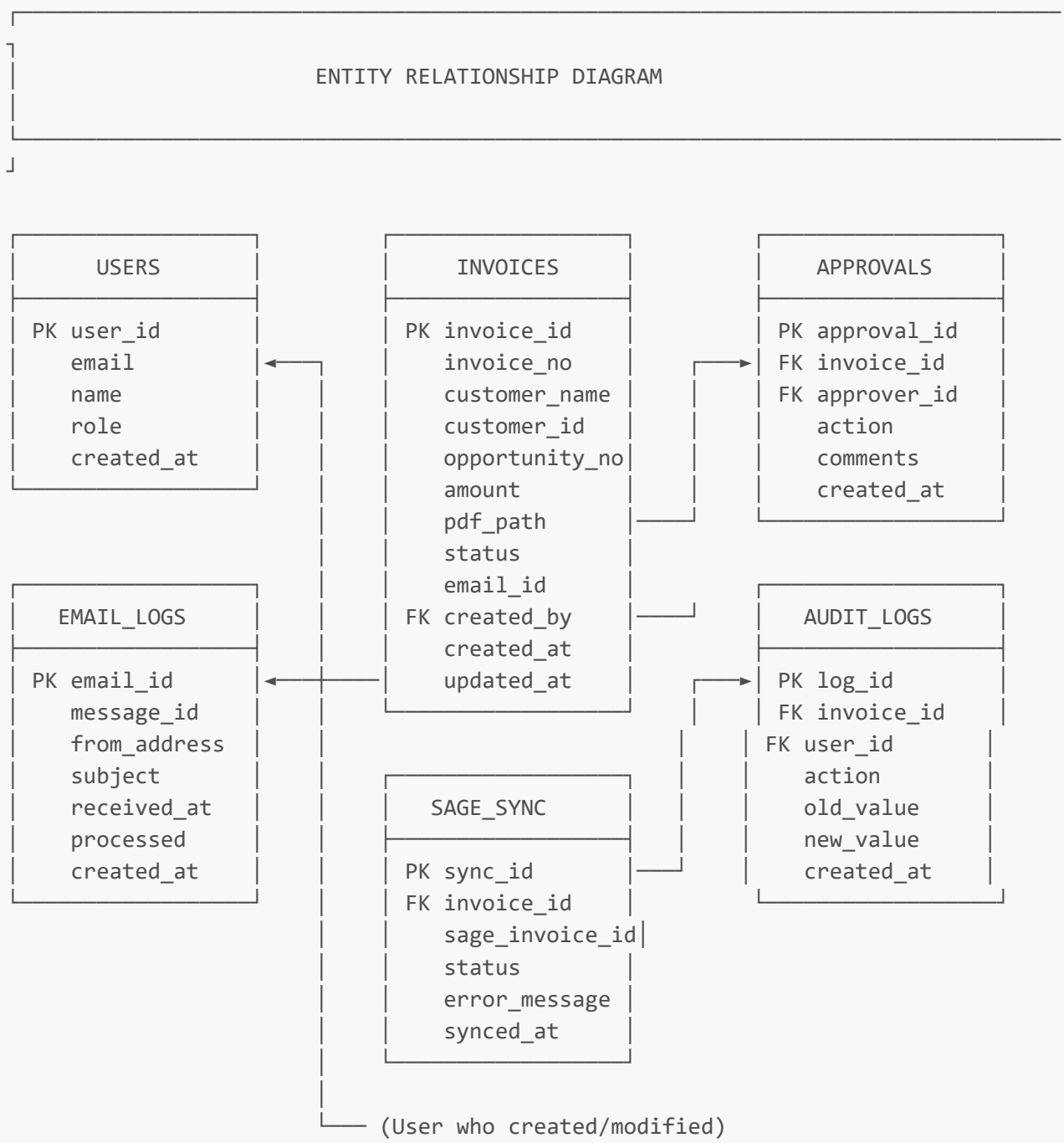
For this documentation, we assume **Option A (Microsoft Stack)** due to:

- Native Microsoft 365 integration requirement
- Enterprise support needs
- Existing Microsoft infrastructure

---

8. Database Design

8.1 Entity Relationship Diagram



8.2 Table Definitions

8.2.1 Users Table

```
CREATE TABLE Users (
    user_id         UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
    email           NVARCHAR(255) NOT NULL UNIQUE,
    name            NVARCHAR(255) NOT NULL,
    role            NVARCHAR(50) NOT NULL,  -- 'Admin', 'Approver', 'Viewer'
    is_active       BIT DEFAULT 1,
    created_at      DATETIME2 DEFAULT GETUTCDATE(),
    updated_at      DATETIME2 DEFAULT GETUTCDATE()
);
```

8.2.2 Invoices Table

```
CREATE TABLE Invoices (
    invoice_id      UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
    invoice_number  NVARCHAR(100) NULL,          -- Extracted from PDF
    customer_name   NVARCHAR(255) NULL,          -- Extracted from PDF
    customer_id     NVARCHAR(100) NULL,          -- Extracted from PDF
    opportunity_no  NVARCHAR(100) NULL,          -- Extracted from PDF
    amount          DECIMAL(18,2) NULL,          -- Extracted from PDF
    currency        NVARCHAR(10) DEFAULT 'USD',
    pdf_path        NVARCHAR(500) NOT NULL,      -- Path to stored PDF
    pdf_filename    NVARCHAR(255) NOT NULL,
    status          NVARCHAR(50) NOT NULL,       -- See status enum below
    email_id        UNIQUEIDENTIFIER NOT NULL,   -- FK to EmailLogs
    extraction_confidence DECIMAL(5,2) NULL,     -- OCR confidence score
    needs_review    BIT DEFAULT 0,
    created_by      UNIQUEIDENTIFIER NULL,       -- FK to Users (for manual)
    created_at      DATETIME2 DEFAULT GETUTCDATE(),
    updated_at      DATETIME2 DEFAULT GETUTCDATE(),

    FOREIGN KEY (email_id) REFERENCES EmailLogs(email_id),
    FOREIGN KEY (created_by) REFERENCES Users(user_id)
);

-- Status Values:
-- 'RECEIVED'          - Email received, not yet processed
-- 'PROCESSING'        - Currently extracting data
-- 'PROCESSED'         - Data extracted successfully
-- 'EXTRACTION_FAILED'- OCR/extraction failed
-- 'PENDING_REVIEW'    - Needs manual review
-- 'PENDING_APPROVAL' - Submitted for approval
-- 'APPROVED'          - Approved by approver
-- 'REJECTED'          - Rejected by approver
-- 'SYNCING'           - Being sent to Sage
-- 'SYNCED'            - Successfully sent to Sage
-- 'SYNC_FAILED'       - Sage sync failed
```

8.2.3 EmailLogs Table

```
CREATE TABLE EmailLogs (
    email_id        UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
    message_id      NVARCHAR(500) NOT NULL UNIQUE, -- Outlook message ID
    from_address    NVARCHAR(255) NOT NULL,
    subject         NVARCHAR(500) NULL,
    received_at     DATETIME2 NOT NULL,
    processed       BIT DEFAULT 0,
    attachment_count INT DEFAULT 0,
```

```
    created_at      DATETIME2 DEFAULT GETUTCDATE()
);
```

8.2.4 Approvals Table

```
CREATE TABLE Approvals (
    approval_id     UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
    invoice_id      UNIQUEIDENTIFIER NOT NULL,
    approver_id     UNIQUEIDENTIFIER NOT NULL,
    action          NVARCHAR(50) NOT NULL,  -- 'APPROVED', 'REJECTED'
    comments        NVARCHAR(MAX) NULL,
    created_at      DATETIME2 DEFAULT GETUTCDATE(),

    FOREIGN KEY (invoice_id) REFERENCES Invoices(invoice_id),
    FOREIGN KEY (approver_id) REFERENCES Users(user_id)
);
```

8.2.5 SageSync Table

```
CREATE TABLE SageSync (
    sync_id         UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
    invoice_id      UNIQUEIDENTIFIER NOT NULL,
    sage_invoice_id NVARCHAR(100) NULL,      -- ID returned by Sage
    status          NVARCHAR(50) NOT NULL,   -- 'PENDING','SUCCESS','FAILED'
    request_payload NVARCHAR(MAX) NULL,      -- JSON sent to Sage
    response_payload NVARCHAR(MAX) NULL,     -- JSON response from Sage
    error_message   NVARCHAR(MAX) NULL,
    retry_count     INT DEFAULT 0,
    synced_at       DATETIME2 NULL,
    created_at      DATETIME2 DEFAULT GETUTCDATE(),

    FOREIGN KEY (invoice_id) REFERENCES Invoices(invoice_id)
);
```

8.2.6 AuditLogs Table

```
CREATE TABLE AuditLogs (
    log_id          UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
    invoice_id      UNIQUEIDENTIFIER NULL,
    user_id         UNIQUEIDENTIFIER NULL,
    action          NVARCHAR(100) NOT NULL,  --
'CREATE','UPDATE','DELETE','APPROVE','REJECT','SYNC'
    entity_type     NVARCHAR(100) NOT NULL,  -- 'INVOICE','USER','APPROVAL'
    old_value       NVARCHAR(MAX) NULL,      -- JSON of old values
    new_value       NVARCHAR(MAX) NULL,      -- JSON of new values
    ip_address      NVARCHAR(50) NULL,
```

```
    user_agent      NVARCHAR(500) NULL,
    created_at      DATETIME2 DEFAULT GETUTCDATE(),

    FOREIGN KEY (invoice_id) REFERENCES Invoices(invoice_id),
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
```

8.3 Database Indexes

```
-- Performance indexes
CREATE INDEX IX_Invoices_Status ON Invoices(status);
CREATE INDEX IX_Invoices_CreatedAt ON Invoices(created_at DESC);
CREATE INDEX IX_Invoices_CustomerName ON Invoices(customer_name);
CREATE INDEX IX_EmailLogs_MessageId ON EmailLogs(message_id);
CREATE INDEX IX_EmailLogs_Processed ON EmailLogs(processed);
CREATE INDEX IX_Approvals_InvoiceId ON Approvals(invoice_id);
CREATE INDEX IX_AuditLogs_InvoiceId ON AuditLogs(invoice_id);
CREATE INDEX IX_AuditLogs_CreatedAt ON AuditLogs(created_at DESC);
```

---

9. API Specifications

9.1 API Overview

Method	Endpoint	Description
GET	`/api/invoices`	List all invoices
GET	`/api/invoices/{id}`	Get invoice details
PUT	`/api/invoices/{id}`	Update invoice
POST	`/api/invoices/{id}/submit`	Submit for approval
POST	`/api/invoices/{id}/approve`	Approve invoice
POST	`/api/invoices/{id}/reject`	Reject invoice
GET	`/api/invoices/{id}/pdf`	Download invoice PDF
GET	`/api/dashboard/stats`	Get dashboard statistics
POST	`/api/invoices/{id}/sync`	Manual sync to Sage

9.2 API Endpoint Details

9.2.1 List Invoices

```
GET /api/invoices
```

Query Parameters:

Parameter	Type	Description
page	int	Page number (default: 1)
pageSize	int	Items per page (default: 20, max: 100)
status	string	Filter by status
search	string	Search in customer name, invoice number
sortBy	string	Field to sort by
sortOrder	string	'asc' or 'desc'

Response:

```
{
  "data": [
    {
      "invoiceId": "guid",
      "invoiceNumber": "INV-2024-001",
      "customerName": "Acme Corp",
      "customerId": "CUST-001",
      "opportunityNumber": "OPP-2024-100",
      "amount": 15000.00,
      "currency": "USD",
      "status": "PENDING_APPROVAL",
      "createdAt": "2024-01-15T10:30:00Z"
    }
  ],
  "pagination": {
    "currentPage": 1,
    "pageSize": 20,
    "totalItems": 150,
    "totalPages": 8
  }
}
```

9.2.2 Get Invoice Details

```
GET /api/invoices/{id}
```

Response:

```
{
  "invoiceId": "guid",
  "invoiceNumber": "INV-2024-001",
```

```
  "customerName": "Acme Corp",
  "customerId": "CUST-001",
  "opportunityNumber": "OPP-2024-100",
  "amount": 15000.00,
  "currency": "USD",
  "status": "PENDING_APPROVAL",
  "pdfUrl": "/api/invoices/{id}/pdf",
  "needsReview": false,
  "extractionConfidence": 95.5,
  "email": {
    "from": "vendor@example.com",
    "subject": "Invoice INV-2024-001",
    "receivedAt": "2024-01-15T10:00:00Z"
  },
  "approvals": [
    {
      "approver": "John Smith",
      "action": "APPROVED",
      "comments": "Looks good",
      "createdAt": "2024-01-15T14:00:00Z"
    }
  ],
  "sageSync": {
    "status": "SUCCESS",
    "sageInvoiceId": "SAGE-001",
    "syncedAt": "2024-01-15T15:00:00Z"
  },
  "auditTrail": [
    {
      "action": "CREATED",
      "user": "System",
      "createdAt": "2024-01-15T10:30:00Z"
    }
  ],
  "createdAt": "2024-01-15T10:30:00Z",
  "updatedAt": "2024-01-15T15:00:00Z"
}
```


9.2.3 Update Invoice

```
PUT /api/invoices/{id}
```

Request Body:


```
{
  "customerName": "Acme Corporation",
  "customerId": "CUST-001",
  "opportunityNumber": "OPP-2024-100",
```

```
  "amount": 15500.00
}
```

9.2.4 Approve Invoice

```
POST /api/invoices/{id}/approve
```

Request Body:

```
{
  "comments": "Approved - amount verified"
}
```

9.2.5 Reject Invoice

```
POST /api/invoices/{id}/reject
```

Request Body:

```
{
  "comments": "Amount does not match PO"
}
```

9.3 Error Responses

```
{
  "error": {
    "code": "INVOICE_NOT_FOUND",
    "message": "Invoice with ID {id} not found",
    "details": null
  }
}
```

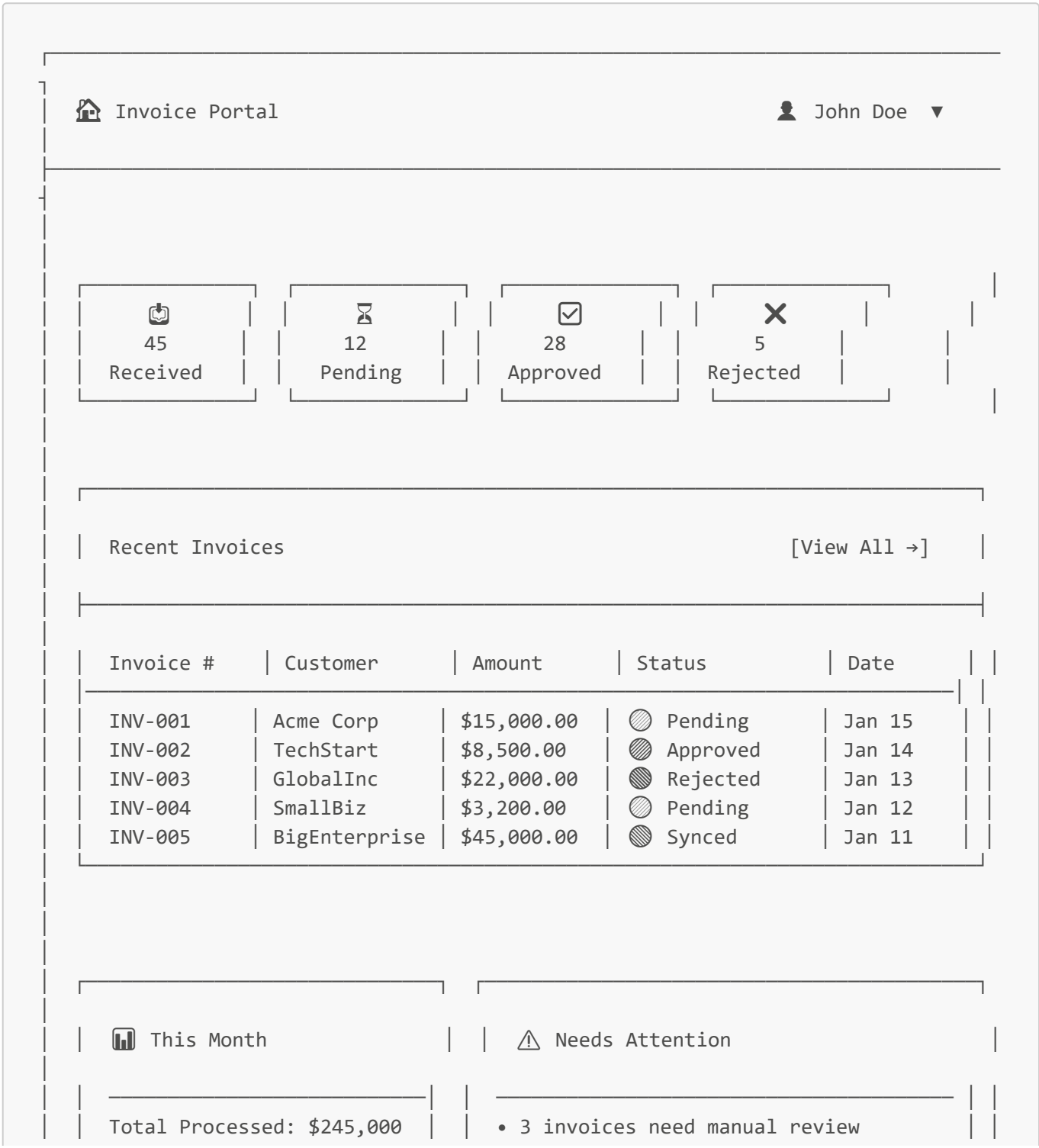
HTTP Code	Error Code	Description
400	VALIDATION_ERROR	Invalid request data
401	UNAUTHORIZED	Authentication required
403	FORBIDDEN	Insufficient permissions

HTTP Code	Error Code	Description
404	NOT_FOUND	Resource not found
409	CONFLICT	State conflict (e.g., already approved)
500	INTERNAL_ERROR	Server error

10. User Interface Design

10.1 Portal Wireframes

10.1.1 Dashboard

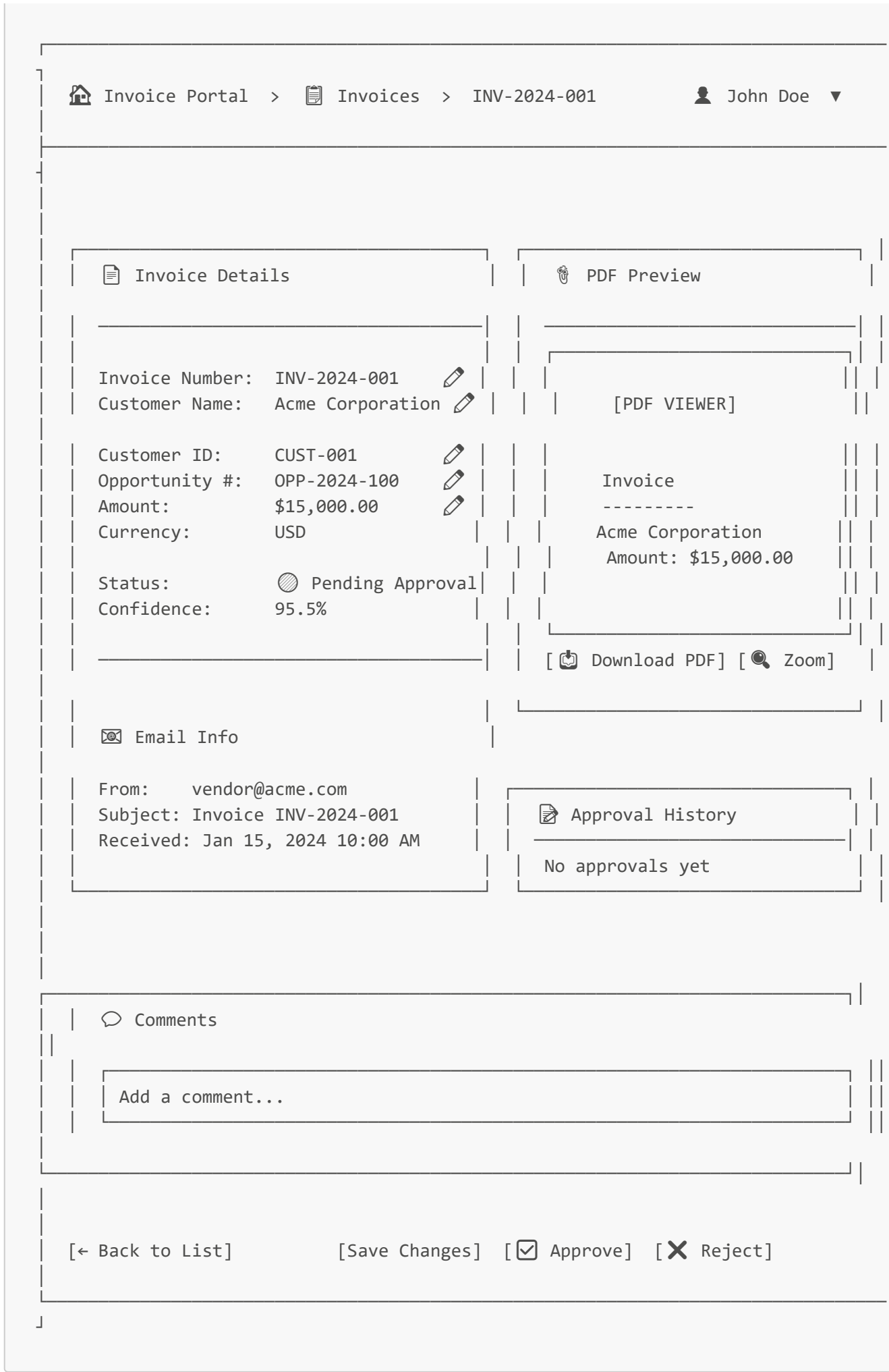


```
│  │  Average: $5,450            │  │  • 2 Sage sync failures                  │ │
│  │  Processing Time: 2.3 hrs   │  │  • 5 approvals pending > 48 hrs          │ │
│  └─────────────────────────────┘  └──────────────────────────────────────────┘ │
│                                                                                │
└────────────────────────────────────────────────────────────────────────────────┘
```

10.1.2 Invoice List View

```
┌──────────────────────────────────────────────────────────────────────────────────┐
│  🏠 Invoice Portal  >  📋 Invoices                              👤 John Doe  ▼   │
├──────────────────────────────────────────────────────────────────────────────────┤
│                                                                                  │
│  ┌──────────────────────────────────────────────────────────────────────────┐    │
│  │ 🔍 Search invoices...                  │ Status: [All ▼] │ Date: [Any ▼] │    │
│  └──────────────────────────────────────────────────────────────────────────┘    │
│                                                                                  │
│  ┌──────────────────────────────────────────────────────────────────────────┐    │
│  │ □ │ Invoice #     │ Customer          │ Opp #      │ Amount       │ Status │    │
│  │───┼───────────────┼───────────────────┼────────────┼──────────────┼────────│    │
│  │ □ │ INV-2024-001  │ Acme Corp         │ OPP-100    │ $15,000.00   │ 🟡 Pending │    │
│  │ □ │ INV-2024-002  │ TechStart Inc     │ OPP-101    │ $8,500.00    │ 🟢 Approved │    │
│  │ □ │ INV-2024-003  │ Global Industries │ OPP-102    │ $22,000.00   │ 🔴 Rejected │    │
│  │ □ │ INV-2024-004  │ SmallBiz LLC      │ OPP-103    │ $3,200.00    │ 🟡 Pending │    │
│  │ □ │ INV-2024-005  │ BigEnterprise     │ OPP-104    │ $45,000.00   │ 🔵 Synced  │    │
│  │ □ │ INV-2024-006  │ StartupXYZ        │ OPP-105    │ $12,750.00   │ ⚠ Review  │    │
│  └──────────────────────────────────────────────────────────────────────────┘    │
│                                                                                  │
│  [◀ Previous]  Page 1 of 8  [Next ▶]                    Showing 1-20 of 150      │
│                                                                                  │
│  Selected: 0  │  [Bulk Approve]  [Bulk Submit]  [Export CSV]                    │
│                                                                                  │
└──────────────────────────────────────────────────────────────────────────────────┘
```

10.1.3 Invoice Detail View


Invoice Portal > Invoices > INV-2024-001 John Doe
Invoice Details
Invoice Number: INV-2024-001
Customer Name: Acme Corporation
Customer ID: CUST-001
Opportunity #: OPP-2024-100
Amount: $15,000.00
Currency: USD
Status: Pending Approval
Confidence: 95.5%
Email Info
From: vendor@acme.com
Subject: Invoice INV-2024-001
Received: Jan 15, 2024 10:00 AM
PDF Preview
[PDF VIEWER]
Invoice
Acme Corporation
Amount: $15,000.00
[Download PDF] [Zoom]
Approval History
No approvals yet
Comments
Add a comment...
[← Back to List]
[Save Changes]
[Approve]
[Reject]

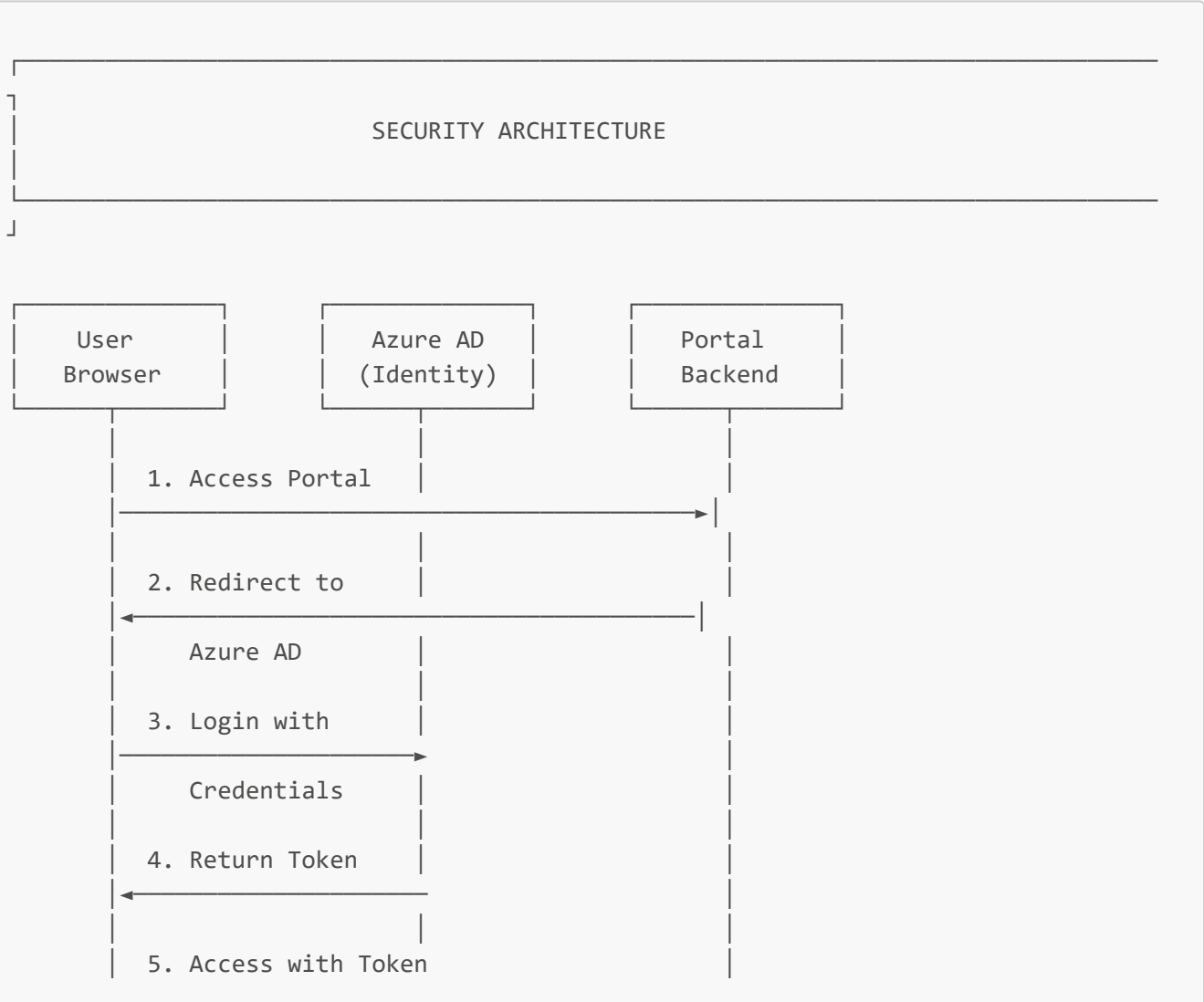
10.2 User Roles and Permissions

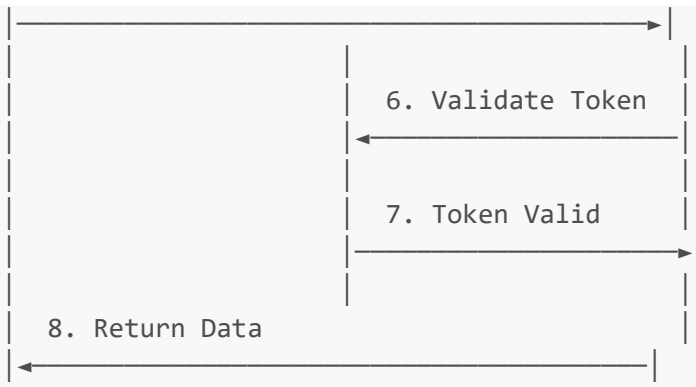
Feature	Admin	Approver	Viewer
View Dashboard	☑	☑	☑
View Invoice List	☑	☑	☑
View Invoice Details	☑	☑	☑
Edit Invoice Data	☑	☑	✗
Submit for Approval	☑	☑	✗
Approve/Reject	☑	☑	✗
Manual Sage Sync	☑	✗	✗
User Management	☑	✗	✗
System Settings	☑	✗	✗

---

11. Security & Compliance

11.1 Authentication & Authorization





11.2 Security Measures

Area	Measure	Implementation
Authentication	SSO with Azure AD	OAuth 2.0 / OpenID Connect
Authorization	Role-based access	Claims-based permissions
Data in Transit	Encryption	TLS 1.3
Data at Rest	Encryption	AES-256
API Security	Rate limiting	100 requests/minute
File Security	Secure storage	Azure Blob with SAS tokens
Audit Logging	Complete trail	All actions logged
Session Management	Secure tokens	JWT with 1-hour expiry

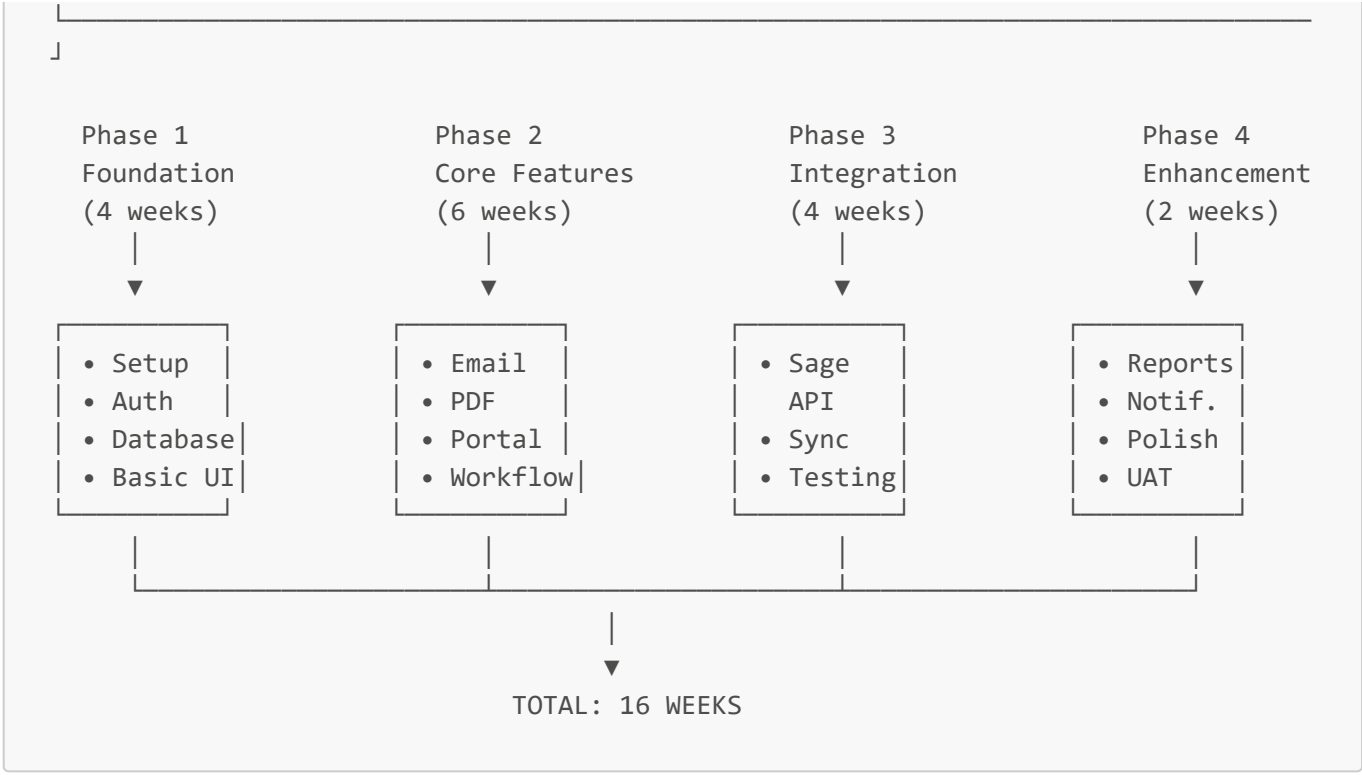
11.3 Compliance Considerations

Requirement	How Addressed
Data Privacy	Only authorized users can access invoices
Audit Trail	Complete history of all actions
Data Retention	Configurable retention policies
Access Control	Role-based permissions
Encryption	All data encrypted in transit and at rest

---

12. Implementation Phases

12.1 Phase Overview



12.2 Phase 1: Foundation (Weeks 1-4)

Week	Tasks	Deliverables
Week 1	Project setup, environment configuration	Dev environment ready
	Database design and creation	Database deployed
	Azure AD integration setup	Authentication working
Week 2	API scaffolding	Basic API structure
	User management	User CRUD operations
	Basic frontend setup	React app skeleton
Week 3	Dashboard UI	Dashboard page
	Invoice list UI	List view with mock data
	Navigation and layout	Complete UI shell
Week 4	Integration testing	All components connected
	Bug fixes	Stable foundation
	Documentation	Technical docs updated

Phase 1 Exit Criteria:

- ☑ Users can log in with Azure AD
- ☑ Basic portal UI is navigable
- ☑ Database is deployed and accessible
- ☑ API endpoints respond correctly

12.3 Phase 2: Core Features (Weeks 5-10)

Week	Tasks	Deliverables
Week 5	Microsoft Graph API integration	Email reading capability
	Email monitoring service	Background job running
	Email logging	Emails tracked in database
Week 6	PDF download and storage	PDFs stored in blob storage
	Azure Document Intelligence setup	OCR service configured
	Text extraction pipeline	Raw text extracted
Week 7	Field extraction logic	Structured data extracted
	Confidence scoring	Extraction quality measured
	Manual review flagging	Problem invoices identified
Week 8	Invoice detail view	Complete detail UI
	PDF viewer integration	View PDF in browser
	Edit functionality	Users can modify data
Week 9	Approval workflow	Submit/Approve/Reject flows
	Email notifications	Notification service
	Status management	Full status lifecycle
Week 10	Integration testing	End-to-end flows working
	Performance optimization	Acceptable response times
	Bug fixes	Stable core features

Phase 2 Exit Criteria:

- ☑ Emails are automatically processed
- ☑ PDFs are parsed and data extracted
- ☑ Users can review and edit invoices
- ☑ Approval workflow is functional

12.4 Phase 3: Integration (Weeks 11-14)

Week	Tasks	Deliverables
Week 11	Sage API analysis	Integration spec finalized
	Sage data mapping	Field mapping documented
	Sage test environment	Test account ready
Week 12	Sage sync service	Auto-sync on approval

Week	Tasks	Deliverables
	Error handling	Retry logic, alerts
	Sync status tracking	Status visible in portal
Week 13	Full integration testing	End-to-end with Sage
	Error scenarios	All edge cases handled
	Performance testing	Load testing completed
Week 14	Security review	Vulnerabilities addressed
	Bug fixes	All critical bugs fixed
	Documentation	Integration docs complete

Phase 3 Exit Criteria:

- ☑ Approved invoices sync to Sage
- ☑ Sync failures are handled gracefully
- ☑ Full audit trail is available

12.5 Phase 4: Enhancement (Weeks 15-16)

Week	Tasks	Deliverables
Week 15	Reporting dashboard	Reports and analytics
	Advanced notifications	Configurable alerts
	UI polish	Final design refinements
Week 16	User acceptance testing	UAT completed
	Training materials	User guide, videos
	Production deployment	Go-live

Phase 4 Exit Criteria:

- ☑ All UAT test cases pass
- ☑ Users are trained
- ☑ Production deployment successful

12.6 Resource Requirements

Role	Count	Phase 1-2	Phase 3-4
Project Manager	1	Part-time	Part-time
Backend Developer	2	Full-time	Full-time
Frontend Developer	1	Full-time	Full-time

Role	Count	Phase 1-2	Phase 3-4
QA Engineer	1	Part-time	Full-time
DevOps Engineer	1	Part-time	Part-time
Business Analyst	1	Full-time	Part-time

12A. Development Effort Estimation

12A.1 Effort Estimation Summary

```
┌─────────────────────────────────────────────────────────────────────────────┐
│                     TOTAL PROJECT EFFORT ESTIMATION                         │
└─────────────────────────────────────────────────────────────────────────────┘

    ┌─────────────────────────────────────────────────────────────────────┐
    │                                                                     │
    │   TOTAL EFFORT:        640 - 800 Person-Hours                       │
    │   TOTAL DURATION:      16 Weeks (4 Months)                          │
    │   TEAM SIZE:           4-5 Developers                               │
    │                                                                     │
    │   ESTIMATED COST:      $64,000 - $120,000 USD                       │
    │   (Based on $100-150/hour blended rate)                             │
    │                                                                     │
    └─────────────────────────────────────────────────────────────────────┘
```

12A.2 Detailed Module-wise Effort Breakdown

Module 1: Project Setup & Infrastructure

Task	Effort (Hours)	Complexity	Notes
Development environment setup	8	Low	Docker, local dev
Azure infrastructure provisioning	16	Medium	Terraform/ARM templates
CI/CD pipeline setup	16	Medium	GitHub Actions/Azure DevOps
Database setup & migrations	8	Low	SQL Server/PostgreSQL
Module Total	**48**		

Module 2: Authentication & User Management

Task	Effort (Hours)	Complexity	Notes
Azure AD integration	16	Medium	OAuth 2.0, MSAL

Task	Effort (Hours)	Complexity	Notes
User roles & permissions	12	Medium	RBAC implementation
User management UI	8	Low	Admin screens
Session management	4	Low	Token refresh
Module Total	**40**		

Module 3: Email Processing Service

Task	Effort (Hours)	Complexity	Notes
Microsoft Graph API integration	24	Medium	Email reading, OAuth
Email monitoring background service	16	Medium	Scheduled job
Attachment download & storage	12	Low	Blob storage integration
Email logging & tracking	8	Low	Database operations
Duplicate email detection	8	Low	Message ID tracking
Error handling & retries	8	Medium	Resilience patterns
Module Total	**76**		

Module 4: PDF Processing & Data Extraction (CRITICAL)

Task	Effort (Hours)	Complexity	Notes
Azure Document Intelligence setup	8	Low	Service configuration
PDF text extraction (digital)	16	Medium	Native PDF parsing
OCR for scanned PDFs	16	Medium	Image-based extraction
AI-based field extraction	**40**	**High**	**Handle variable formats**
Field label recognition	24	High	Multiple label variations
Value extraction & parsing	24	High	Amount, dates, IDs
Confidence scoring algorithm	16	Medium	Quality assessment
Low-confidence flagging	8	Low	Review queue
Testing with diverse formats	24	High	Multiple vendor formats
Module Total	**176**		**Most complex module**

Module 5: Web Portal - Frontend

Task	Effort (Hours)	Complexity	Notes

Task	Effort (Hours)	Complexity	Notes
Project setup (React/Angular)	8	Low	Boilerplate, routing
Dashboard component	16	Medium	Stats, charts
Invoice list view	16	Medium	Table, filters, search
Invoice detail view	24	Medium	Form, PDF viewer
PDF viewer integration	12	Medium	In-browser viewing
Manual data editing	12	Low	Form validation
Approval workflow UI	16	Medium	Actions, comments
Notifications UI	8	Low	Toast, alerts
Responsive design	12	Medium	Mobile-friendly
Module Total	**124**		

Module 6: Web Portal - Backend API

Task	Effort (Hours)	Complexity	Notes
API project setup	8	Low	.NET Core / Node.js
Invoice CRUD endpoints	16	Low	Standard REST
Search & filtering	12	Medium	Query optimization
Approval workflow logic	16	Medium	State machine
File download endpoints	8	Low	Secure PDF access
Dashboard statistics	12	Medium	Aggregation queries
Audit logging	12	Medium	All actions tracked
Error handling & validation	8	Low	Global handlers
Module Total	**92**		

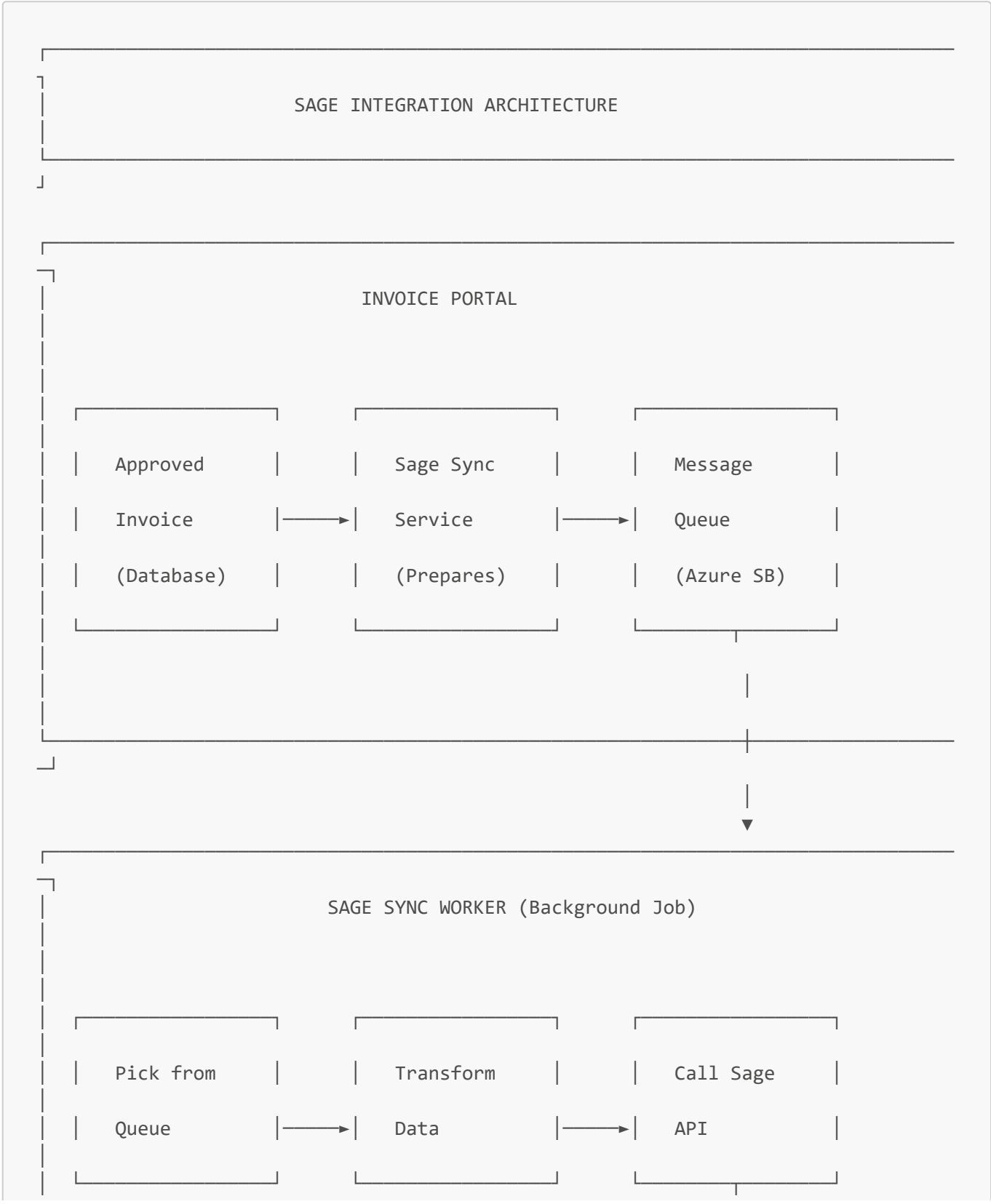
Module 7: Sage Integration

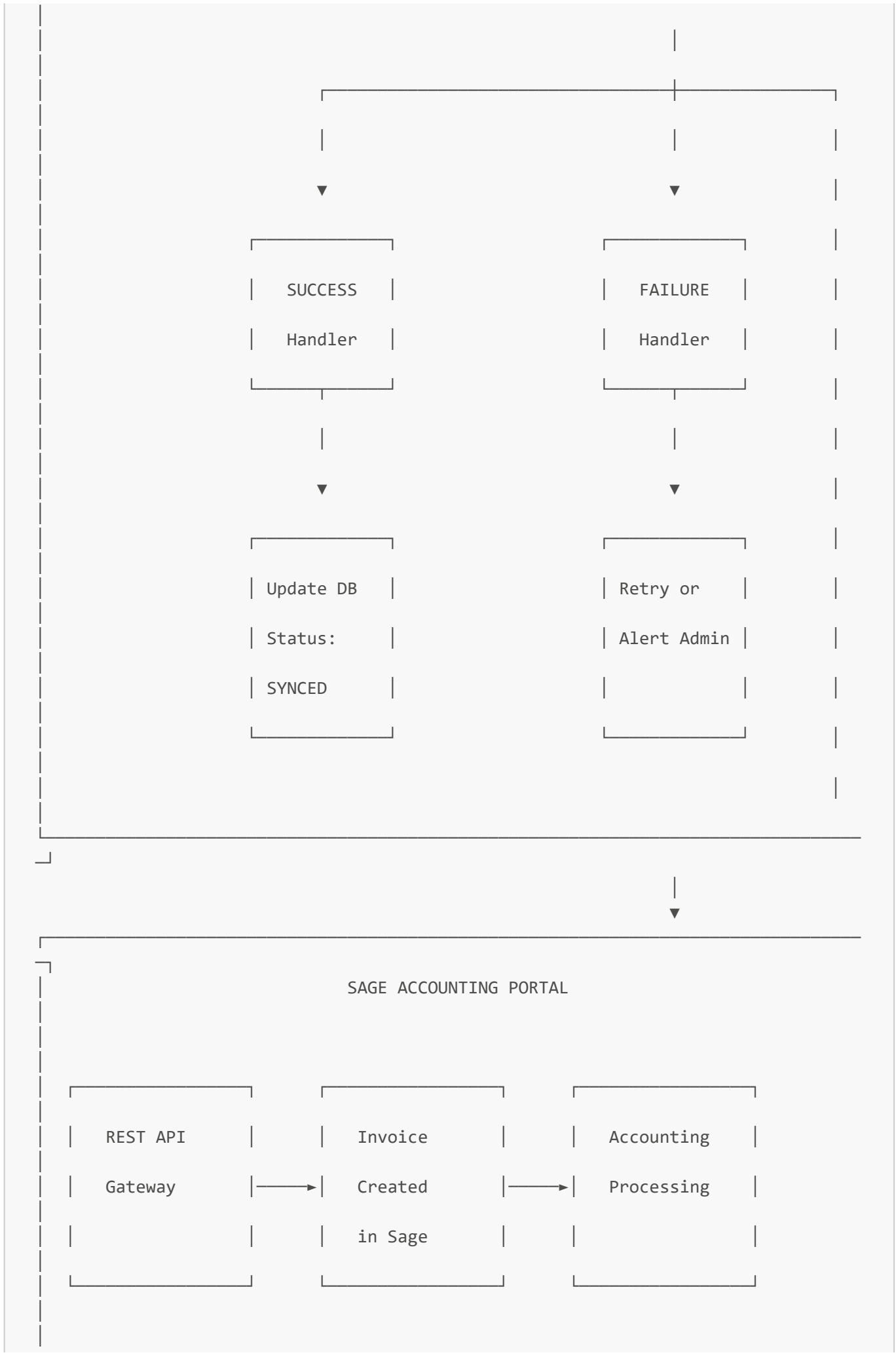
Task	Effort (Hours)	Complexity	Notes
Sage API research & analysis	16	Medium	API documentation
Data mapping implementation	12	Medium	Field transformations
Sync service development	24	Medium	Queue-based processing
Error handling & retry logic	12	Medium	Resilience
Sync status tracking	8	Low	Status updates

Task	Effort (Hours)	Complexity	Notes
Manual retry functionality	8	Low	Admin feature
Module Total	**80**		

12B. Sage Integration - Detailed Data Flow

12B.1 Sage Integration Architecture




SUCCESS
Handler
FAILURE
Handler
Update DB
Status:
SYNCED
Retry or
Alert Admin
SAGE ACCOUNTING PORTAL
REST API
Gateway
Invoice
Created
in Sage
Accounting
Processing

```
  │
  └──────────────────────────────────────────────────────────────────────────────
```

12B.2 Data Flow - Step by Step

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                                                                              │
│                    SAGE SYNC - DETAILED DATA FLOW                            │
│                                                                              │
└──────────────────────────────────────────────────────────────────────────────┘

  STEP 1: APPROVAL TRIGGERS SYNC
  ═══════════════════════════════════════════════════════════════════════════

  ┌─────────────────┐          ┌─────────────────┐          ┌─────────────────┐
  │   Approver      │          │   Invoice       │          │   Sync Queue    │
  │   Clicks        │─────────►│   Status =      │─────────►│   Message       │
  │   "APPROVE"     │          │   "APPROVED"    │          │   Created       │
  └─────────────────┘          └─────────────────┘          └─────────────────┘

  Queue Message Payload:
  ┌──────────────────────────────────────────────────────────────────────────┐
  │ {                                                                        │
  │   "messageId": "msg-uuid-12345",                                         │
  │   "invoiceId": "inv-uuid-67890",                                         │
  │   "action": "SYNC_TO_SAGE",                                              │
  │   "priority": "normal",                                                  │
  │   "createdAt": "2026-02-05T10:30:00Z",                                   │
  │   "retryCount": 0                                                        │
  │ }                                                                        │
  └──────────────────────────────────────────────────────────────────────────┘

  STEP 2: WORKER PICKS UP MESSAGE
  ═══════════════════════════════════════════════════════════════════════════

  ┌─────────────────┐          ┌─────────────────┐          ┌─────────────────┐
  │   Sage Sync     │          │   Load Invoice  │          │   Validate      │
  │   Worker        │─────────►│   from          │─────────►│   Required      │
  │   (Listener)    │          │   Database      │          │   Fields        │
  └─────────────────┘          └─────────────────┘          └─────────────────┘

  STEP 3: DATA TRANSFORMATION
  ═══════════════════════════════════════════════════════════════════════════

  ┌──────────────────────────────────┐          ┌──────────────────────────────────┐
  │      PORTAL DATA (Source)        │          │       SAGE DATA (Target)         │
  ├──────────────────────────────────┤          ├──────────────────────────────────┤
  │ invoice_number: "INV-2024-001"   │─────────►│ InvoiceReference: "INV-2024-001" │
  │ customer_name: "Acme Corp"       │─────────►│ CustomerName: "Acme Corp"        │
```

```
│ customer_id: "CUST-001"          │──────────►│ CustomerCode: "CUST-001"         │
│ opportunity_no: "OPP-100"        │──────────►│ Reference: "OPP-100"             │
│ amount: 15000.00                 │──────────►│ TotalAmount: 15000.00            │
│ currency: "USD"                  │──────────►│ CurrencyCode: "USD"              │
│ created_at: "2026-02-05"         │──────────►│ InvoiceDate: "2026-02-05"        │
└──────────────────────────────────┘           └──────────────────────────────────┘

STEP 4: API CALL TO SAGE
════════════════════════════════════════════════════════════════════════════════

┌─────────────────┐                      ┌─────────────────┐
│   Sync Worker   │ ──── HTTPS POST ───► │   Sage API      │
│                 │                      │   /invoices     │
└─────────────────┘                      └─────────────────┘

STEP 5: HANDLE RESPONSE
════════════════════════════════════════════════════════════════════════════════

                         ┌─────────────────┐
                         │  Sage Response  │
                         └────────┬────────┘
                                  │
               ┌──────────────────┼──────────────────┐
               │                  │                  │
               ▼                  ▼                  ▼
        ┌─────────────┐    ┌─────────────┐    ┌─────────────┐
        │ 200/201 OK  │    │ 4xx Error   │    │ 5xx Error   │
        │ SUCCESS     │    │ BAD REQUEST │    │ SERVER ERR  │
        └──────┬──────┘    └──────┬──────┘    └──────┬──────┘
               │                  │                  │
               ▼                  ▼                  ▼
        ┌─────────────┐    ┌─────────────┐    ┌─────────────┐
        │ Status:     │    │ Status:     │    │ Retry up to │
        │ SYNCED      │    │ SYNC_FAILED │    │ 3 times     │
        │ Store Sage  │    │ Log Error   │    │ Then FAIL   │
        │ Invoice ID  │    │ Alert Admin │    │ Alert Admin │
        └─────────────┘    └─────────────┘    └─────────────┘
```

12B.3 Sage API Request/Response Specification

Authentication

```
┌──────────────────────────────────────────────────────────────────────────────────┐
│                          SAGE API AUTHENTICATION                                 │
│                                                                                  │
└──────────────────────────────────────────────────────────────────────────────────┘

  1. OAuth 2.0 Client Credentials Flow (Recommended for Server-to-Server)
```

Token Request:

```
POST https://api.sage.com/oauth/token
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
&client_id={CLIENT_ID}
&client_secret={CLIENT_SECRET}
&scope=invoices.write
```

Token Response:

```
{
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6...",
  "token_type": "Bearer",
  "expires_in": 3600,
  "scope": "invoices.write"
}
```

Create Invoice Request

```
POST https://api.sage.com/v3.1/purchase_invoices
Authorization: Bearer {access_token}
Content-Type: application/json
X-Request-Id: {unique-request-id}
```

Request Payload:

```
{
  "purchase_invoice": {
    "contact_id": "CUST-001",
    "contact_name": "Acme Corporation",
    "date": "2026-02-05",
    "due_date": "2026-03-05",
    "reference": "INV-2024-001",
    "vendor_reference": "OPP-2024-100",
    "notes": "Synced from Invoice Portal",
    "currency_id": "USD",
    "invoice_lines": [
```

```
      {
        "description": "Invoice INV-2024-001 from Acme Corporation",
        "quantity": 1,
        "unit_price": 15000.00,
        "tax_rate_id": "TAX-STANDARD",
        "ledger_account_id": "PURCHASES"
      }
    ],
    "total_amount": 15000.00,
    "external_reference": "PORTAL-inv-uuid-67890"
  }
}
```

Success Response (201 Created)

```
{
  "purchase_invoice": {
    "id": "SAGE-PI-12345",
    "displayed_as": "PI-00123",
    "reference": "INV-2024-001",
    "contact_id": "CUST-001",
    "contact_name": "Acme Corporation",
    "date": "2026-02-05",
    "due_date": "2026-03-05",
    "total_amount": 15000.00,
    "outstanding_amount": 15000.00,
    "currency": {
      "id": "USD",
      "symbol": "$"
    },
    "status": {
      "id": "UNPAID",
      "displayed_as": "Unpaid"
    },
    "created_at": "2026-02-05T10:35:00Z",
    "updated_at": "2026-02-05T10:35:00Z"
  }
}
```

Error Response Examples

400 Bad Request - Validation Error:

```
{
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Validation failed",
    "details": [
```

```
      {
        "field": "contact_id",
        "message": "Contact not found with ID: CUST-001"
      },
      {
        "field": "total_amount",
        "message": "Total amount must be greater than 0"
      }
    ]
  }
}
```

401 Unauthorized:

```
{
  "error": {
    "code": "UNAUTHORIZED",
    "message": "Invalid or expired access token"
  }
}
```

429 Rate Limited:

```
{
  "error": {
    "code": "RATE_LIMITED",
    "message": "Too many requests. Please retry after 60 seconds",
    "retry_after": 60
  }
}
```

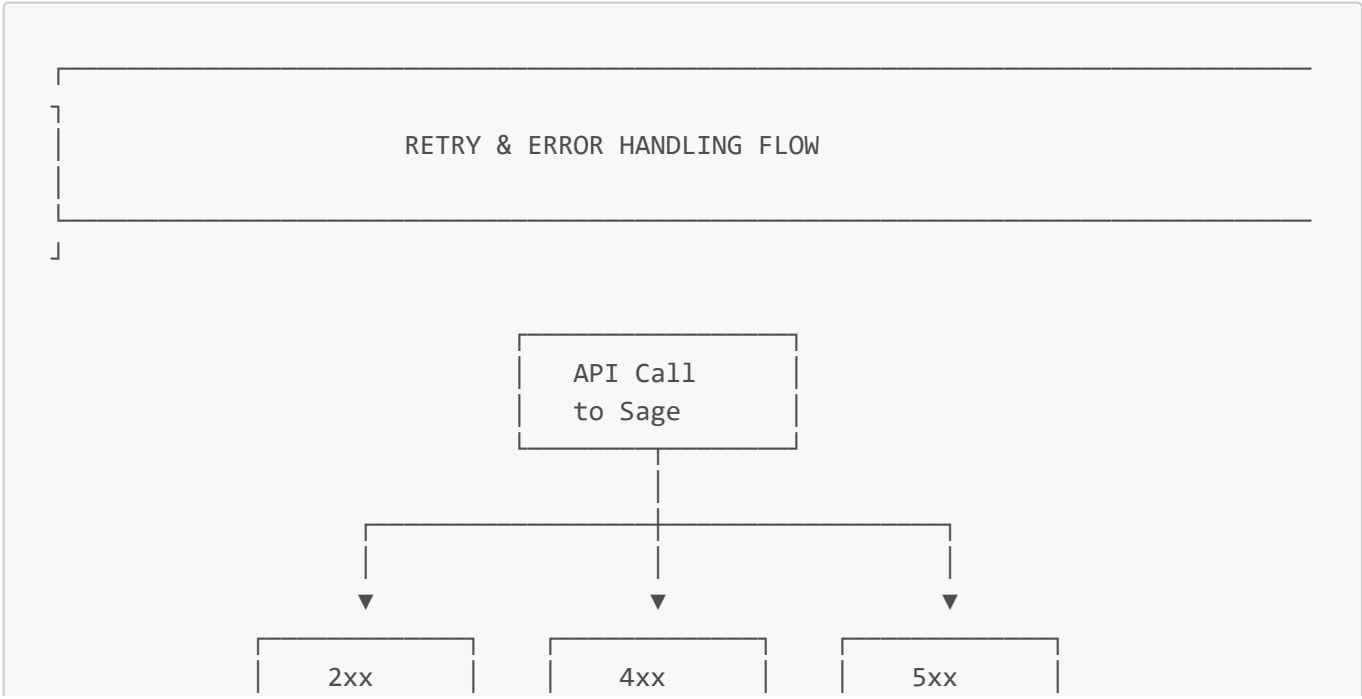
500 Server Error:

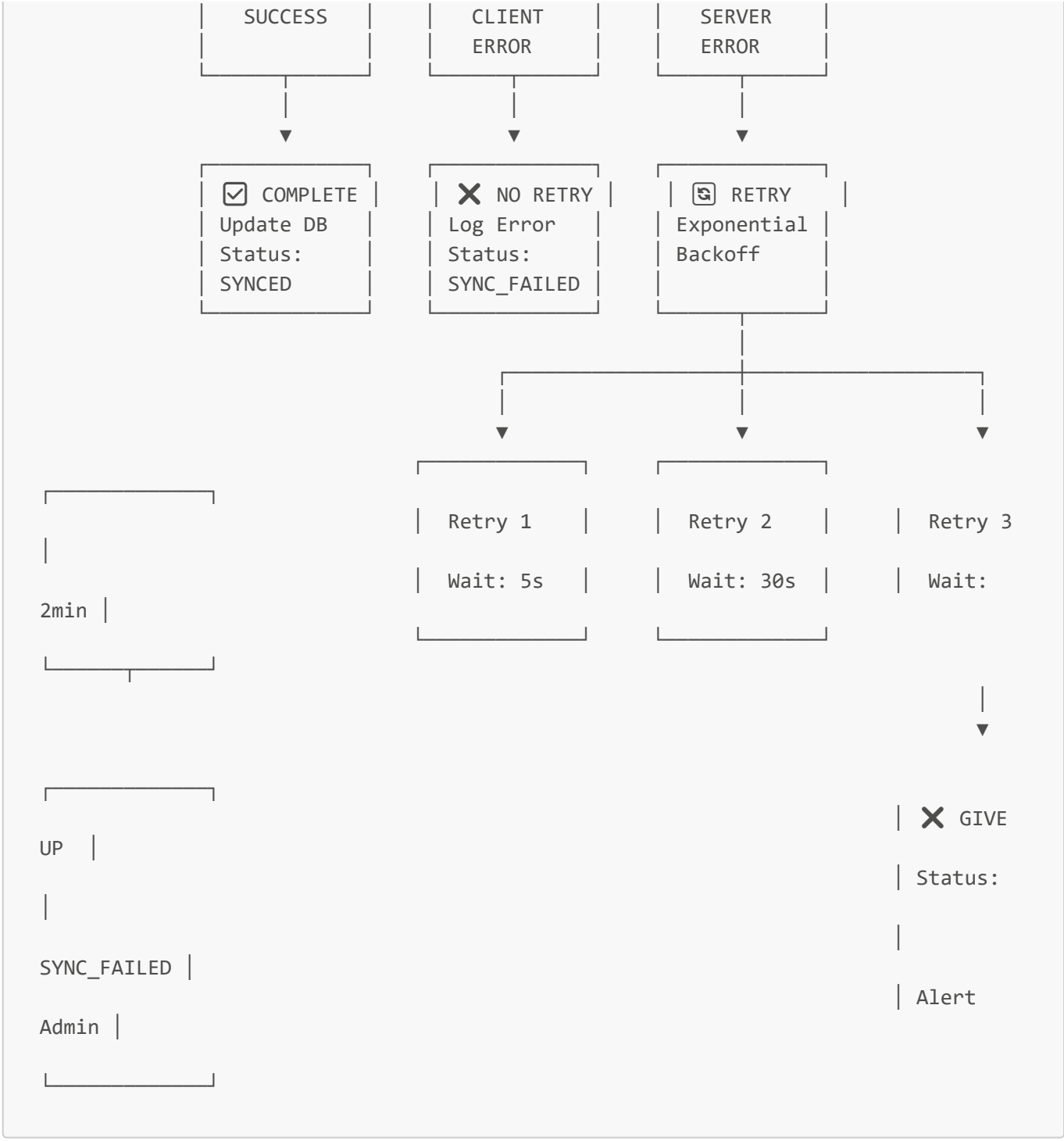
```
{
  "error": {
    "code": "INTERNAL_ERROR",
    "message": "An unexpected error occurred. Please try again later",
    "request_id": "req-12345"
  }
}
```

12B.4 Complete Data Mapping Table

Portal Field	Portal DB Column	Transformation	Sage API Field	Sage Data Type
Invoice Number	`invoice_number`	Direct map	`reference`	String (50)
Customer Name	`customer_name`	Direct map	`contact_name`	String (200)
Customer ID	`customer_id`	Direct map	`contact_id`	String (50)
Opportunity No	`opportunity_no`	Direct map	`vendor_reference`	String (50)
Amount	`amount`	Direct map	`total_amount`	Decimal (18,2)
Currency	`currency`	Map to Sage currency	`currency_id`	String (3)
Invoice Date	`created_at`	Format: YYYY-MM-DD	`date`	Date
Due Date	Calculated	+30 days from date	`due_date`	Date
Portal Invoice ID	`invoice_id`	Direct map	`external_reference`	String (100)
Line Description	Computed	Invoice # + Customer	`invoice_lines[0].description`	String (500)

12B.5 Retry & Error Handling Strategy





Retry Policy Configuration:

Error Type	Retry?	Max Retries	Backoff Strategy
400 Bad Request	❌ No	0	N/A - Fix data and re-submit
401 Unauthorized	🔄 Yes	1	Refresh token, then retry
403 Forbidden	❌ No	0	N/A - Permission issue
404 Not Found	❌ No	0	N/A - Invalid endpoint
429 Rate Limited	🔄 Yes	3	Use `retry_after` header
500 Server Error	🔄 Yes	3	Exponential: 5s, 30s, 2min

Error Type	Retry?	Max Retries	Backoff Strategy
502 Bad Gateway	🔄 Yes	3	Exponential backoff
503 Unavailable	🔄 Yes	3	Exponential backoff
Network Timeout	🔄 Yes	3	Exponential backoff

12B.6 Sync Status Tracking

Database Record (SageSync Table):

```
{
  "sync_id": "sync-uuid-001",
  "invoice_id": "inv-uuid-67890",
  "sage_invoice_id": "SAGE-PI-12345",
  "status": "SUCCESS",
  "request_payload": "{...full request JSON...}",
  "response_payload": "{...full response JSON...}",
  "error_message": null,
  "retry_count": 0,
  "synced_at": "2026-02-05T10:35:00Z",
  "created_at": "2026-02-05T10:30:00Z"
}
```

Failed Sync Record:

```
{
  "sync_id": "sync-uuid-002",
  "invoice_id": "inv-uuid-99999",
  "sage_invoice_id": null,
  "status": "FAILED",
  "request_payload": "{...full request JSON...}",
  "response_payload": "{\"error\": {\"code\": \"VALIDATION_ERROR\"...}}",
  "error_message": "Contact not found with ID: CUST-INVALID",
  "retry_count": 3,
  "synced_at": null,
  "created_at": "2026-02-05T11:00:00Z"
}
```

12B.7 Monitoring & Alerts

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                                                                              │
│                          SAGE SYNC MONITORING                                │
│                                                                              │
└──────────────────────────────────────────────────────────────────────────────┘
```

```
┌─────────────────────────────────────────────────────────────────────────┐
│                          ADMIN DASHBOARD                                │
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐  │
│  │   Pending    │  │    Synced    │  │    Failed    │  │   Avg Time   │  │
│  │      12      │  │     145      │  │       3      │  │     2.3s     │  │
│  │   in queue   │  │   today      │  │   today      │  │   response   │  │
│  └──────────────┘  └──────────────┘  └──────────────┘  └──────────────┘  │
└─────────────────────────────────────────────────────────────────────────┘

ALERTS TRIGGERED:
═══════════════════════════════════════════════════════════════════════════

⚠  ALERT: Sync failure after 3 retries
   Invoice: INV-2024-099 | Error: Contact not found
   Action: Manual review required

⚠  ALERT: High failure rate detected
   Failure rate: 15% (threshold: 5%)
   Action: Check Sage API status

⚠  ALERT: Sync queue backlog
   Pending items: 50 (threshold: 20)
   Action: Scale up workers
```

Alert Configuration:

Alert Type	Threshold	Notification
Sync Failure	Any failure after max retries	Email to Admin
High Failure Rate	> 5% failures in 1 hour	Email + Slack
Queue Backlog	> 20 pending items	Email
API Downtime	> 5 consecutive failures	Email + SMS
Response Time	> 10 seconds average	Email

Module 8: Testing & Quality Assurance

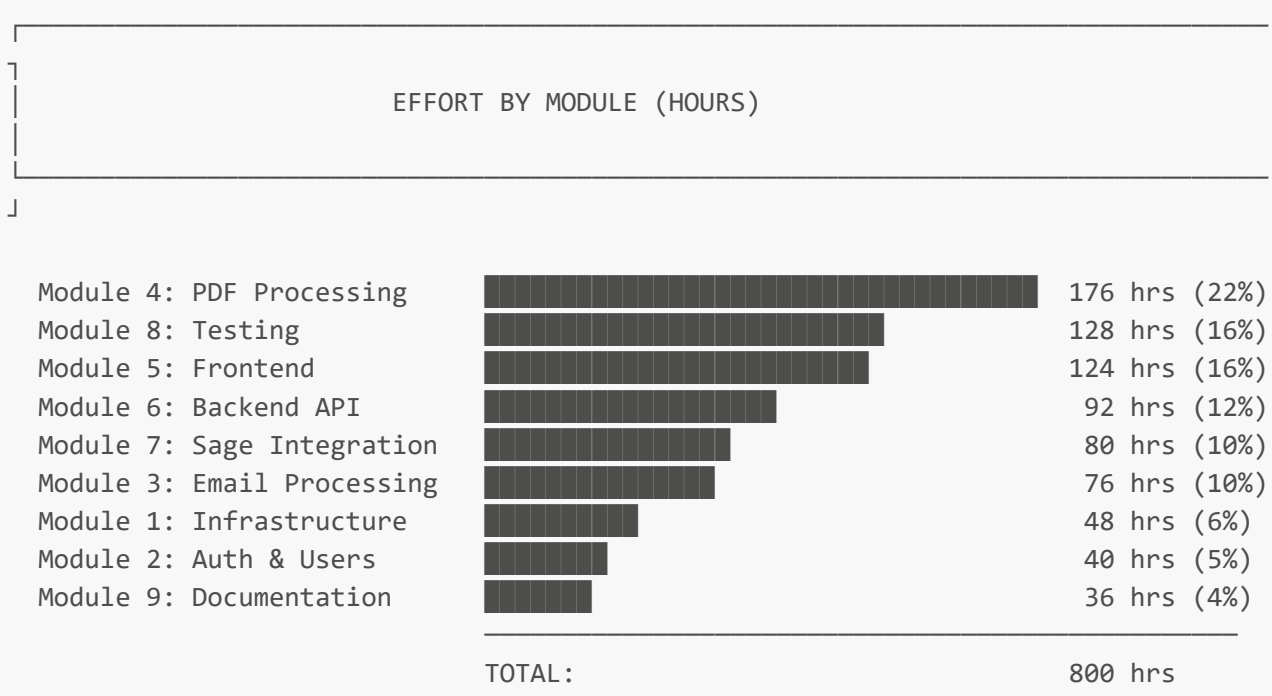
Task	Effort (Hours)	Complexity	Notes
Unit test development	24	Medium	70%+ coverage target
Integration testing	24	Medium	API, database
End-to-end testing	16	Medium	Playwright/Selenium
PDF extraction testing	24	High	Multiple formats
Performance testing	12	Medium	Load testing
Security testing	12	Medium	OWASP checks

Task	Effort (Hours)	Complexity	Notes
UAT support	16	Low	User testing
Module Total	**128**		

Module 9: Documentation & Training

Task	Effort (Hours)	Complexity	Notes
Technical documentation	16	Low	API docs, architecture
User guide	12	Low	How-to guides
Training materials	8	Low	Videos, walkthroughs
Module Total	**36**		

12A.3 Effort Summary by Module



12A.4 Effort by Phase

Phase	Duration	Effort (Hours)	Percentage
Phase 1: Foundation	4 weeks	136	17%
Phase 2: Core Features	6 weeks	400	50%
Phase 3: Integration	4 weeks	172	22%
Phase 4: Enhancement	2 weeks	92	11%
Total	**16 weeks**	**800**	**100%**

12A.5 Cost Estimation

Option A: In-House Development

Role	Hours	Rate ($/hr)	Cost
Senior Backend Developer	280	$80	$22,400
Senior Frontend Developer	180	$75	$13,500
Full Stack Developer	160	$70	$11,200
QA Engineer	128	$60	$7,680
DevOps Engineer	52	$85	$4,420
Total Labor	**800**		**$59,200**
Azure Infrastructure (4 months)			$1,660
Tools & Licenses			$1,000
Grand Total			**$61,860**

Option B: Outsourced Development

Region	Rate Range ($/hr)	Estimated Cost
US/UK/Australia	$120-180	$96,000 - $144,000
Western Europe	$80-120	$64,000 - $96,000
Eastern Europe	$50-80	$40,000 - $64,000
India/Southeast Asia	$30-50	$24,000 - $40,000

Option C: Hybrid (Recommended)

Component	Approach	Cost
Architecture & Design	In-house/Senior	$8,000
Core Development	Outsourced (Eastern Europe)	$40,000
QA & Testing	Outsourced	$8,000
Project Management	In-house	$6,000
Infrastructure	Azure	$1,660
Total		**$63,660**

12A.6 Risk-Based Estimation Adjustment

Risk Factor	Impact	Adjustment

Risk Factor	Impact	Adjustment
PDF format variations more complex than expected	High	+20%
Sage API integration challenges	Medium	+10%
Scope creep	Medium	+15%
Resource availability	Low	+5%

Recommended Buffer: 20-30%

```
┌─────────────────────────────────────────────────────────────────────────────┐
│                     FINAL ESTIMATION WITH BUFFER                            │
└─────────────────────────────────────────────────────────────────────────────┘

  Base Estimate:          800 hours        │  $60,000 - $65,000
  With 25% Buffer:        1,000 hours      │  $75,000 - $85,000

  Timeline:               16-20 weeks      │  4-5 months
```

12A.7 Estimation Assumptions

1. Team has experience with chosen technology stack
2. Azure/Cloud infrastructure is available
3. Sage API documentation and test environment are accessible
4. Business stakeholders are available for clarifications
5. No major scope changes during development
6. PDF samples from various vendors are available for testing

---

13. Testing Strategy

13.1 Testing Levels

```
┌─────────────────────────────────────────────────────────────────────────────┐
│                          TESTING PYRAMID                                    │
└─────────────────────────────────────────────────────────────────────────────┘

                            /\
                           /  \
                          / E2E\        End-to-End Tests
                         /Tests \       (Selenium/Playwright)
                        /────────\
```

```
                    /Integration\    Integration Tests
                   /    Tests    \   (API, Database)
                  /————————————————\
                 /    Unit Tests    \    Unit Tests
                /                    \   (xUnit, Jest)
               ————————————————————————
```

13.2 Test Cases Summary

Category	Test Cases	Priority
Email Processing	Email detection, attachment download, duplicate handling	High
PDF Extraction	Text extraction, field parsing, confidence scoring	High
Invoice CRUD	Create, read, update, status changes	High
Approval Workflow	Submit, approve, reject, notifications	High
Sage Integration	Sync success, sync failure, retry logic	High
Authentication	Login, logout, session management	High
Authorization	Role permissions, access control	Medium
UI/UX	Responsive design, accessibility	Medium
Performance	Load testing, response times	Medium

13.3 Sample Test Scenarios

Scenario 1: Happy Path - Invoice Processing

```
Given: A vendor sends an email with PDF invoice to invoices@company.com
When: The system processes the email
Then: Invoice data is extracted and stored
And: Invoice appears on the portal with status "Pending Review"
```

Scenario 2: Approval Flow

```
Given: An invoice is in "Pending Approval" status
When: An approver clicks "Approve" with comments
Then: Invoice status changes to "Approved"
And: Approver and submitter receive email notification
And: Invoice is queued for Sage sync
```

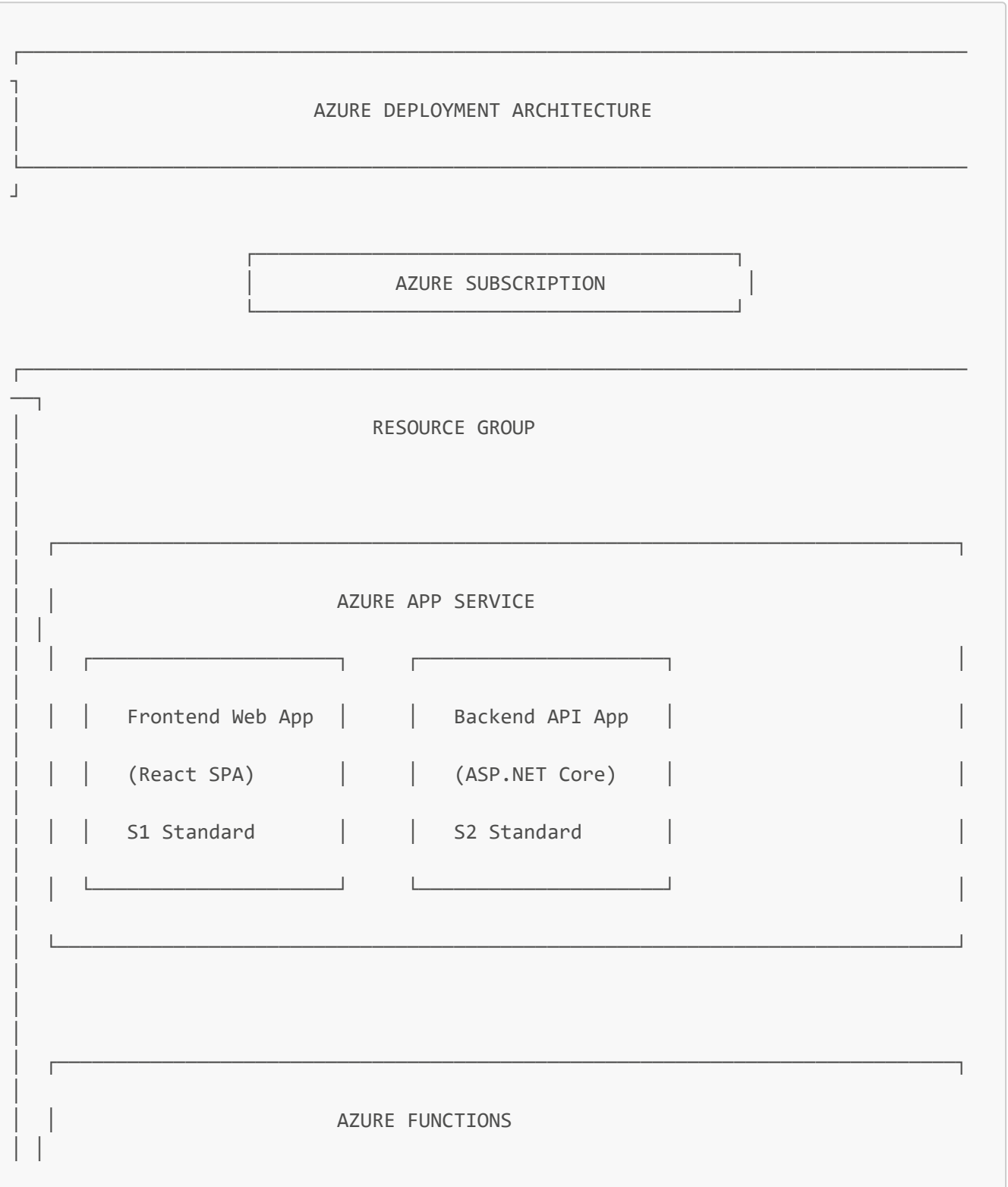
Scenario 3: Sage Sync Failure

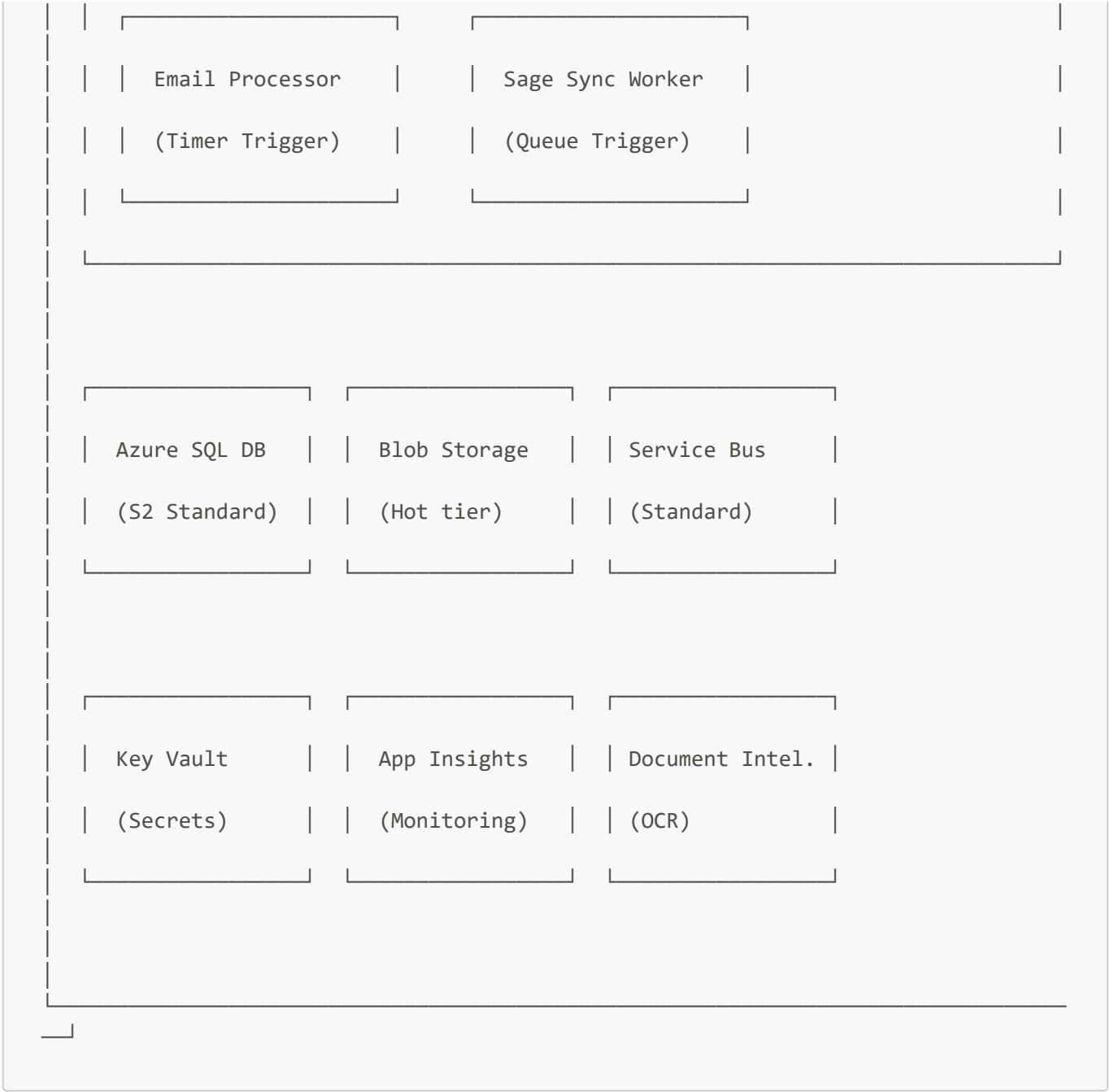
```
Given: An approved invoice fails to sync to Sage
When: The sync fails 3 times
Then: Invoice status shows "Sync Failed"
And: Admin receives alert notification
And: Manual retry option is available
```

---

14. Deployment & Infrastructure

14.1 Deployment Architecture (Azure)





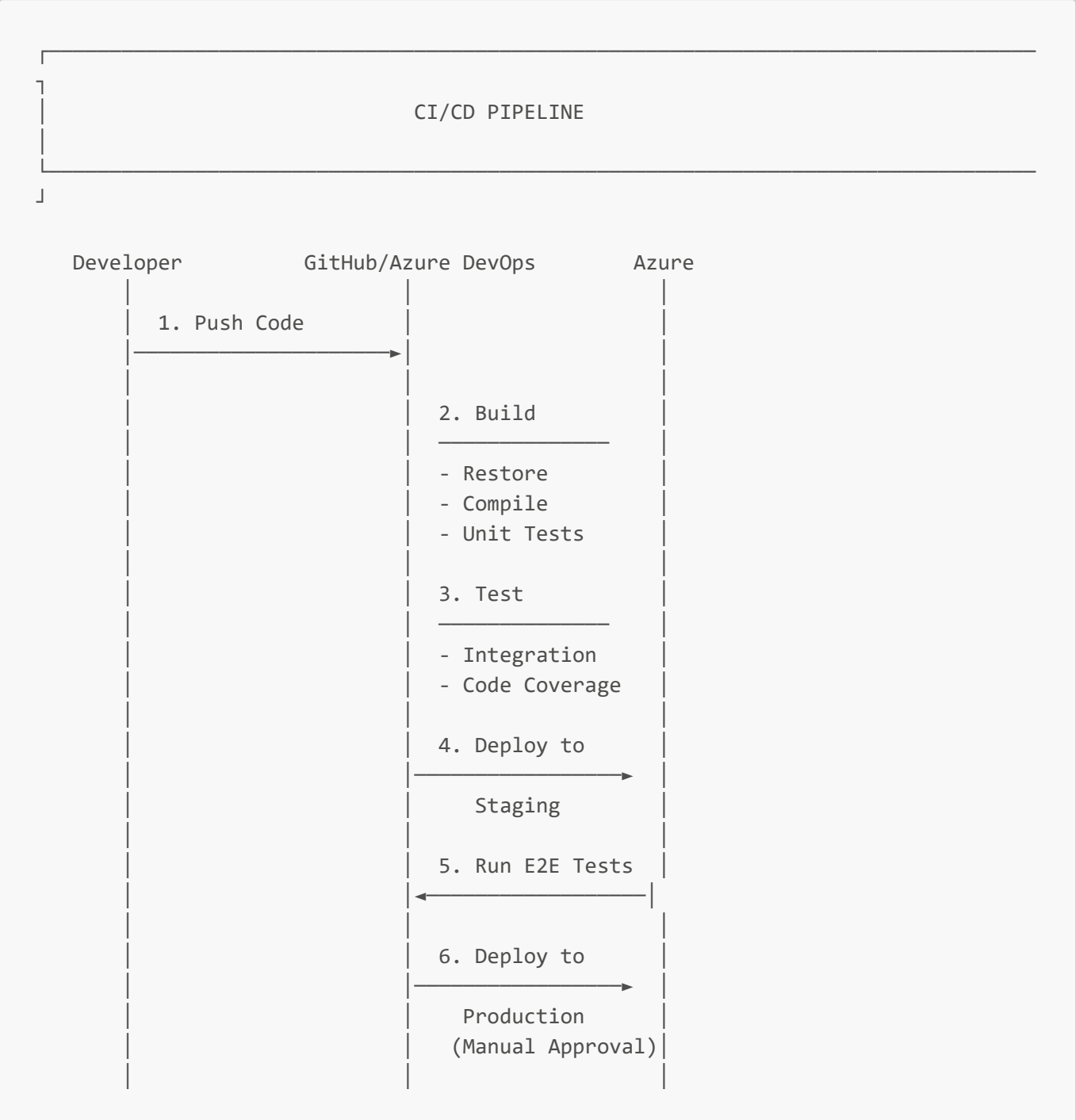
14.2 Estimated Azure Costs

Resource	SKU	Monthly Cost (Est.)
App Service (Frontend)	S1 Standard	$70
App Service (Backend)	S2 Standard	$140
Azure Functions	Consumption	$20
Azure SQL Database	S2 Standard	$75
Blob Storage	Hot (100GB)	$20
Service Bus	Standard	$10
Document Intelligence	S0	$50
Key Vault	Standard	$5

Resource	SKU	Monthly Cost (Est.)
Application Insights	Basic	$25
Total		**~$415/month**

Note: Costs are estimates and may vary based on usage.

14.3 CI/CD Pipeline



15. Glossary

15.1 Business Terms

Term	Definition
Invoice	A commercial document issued by a seller to a buyer
Approval Workflow	Process where invoices are reviewed and approved/rejected
Sage	Accounting software used for financial management
Opportunity Number	Reference number linking invoice to a sales opportunity
Customer ID	Unique identifier for a customer in the system

15.2 Technical Terms

Term	Definition
API	Application Programming Interface - how systems communicate
OCR	Optical Character Recognition - converts images to text
Azure AD	Microsoft's cloud identity service
Blob Storage	Cloud storage for unstructured data (files)
JWT	JSON Web Token - secure token for authentication
REST	Representational State Transfer - API architecture style
SPA	Single Page Application - modern web app architecture
SSO	Single Sign-On - one login for multiple applications

15.3 Status Definitions

Status	Meaning	Next Actions
RECEIVED	Email received, not yet processed	Automatic processing
PROCESSING	Currently extracting data	Wait for completion
PROCESSED	Data extracted successfully	Review and submit
EXTRACTION_FAILED	OCR/extraction failed	Manual data entry
PENDING_REVIEW	Needs manual review	Edit and verify data
PENDING_APPROVAL	Submitted for approval	Approve or reject
APPROVED	Approved by approver	Auto-sync to Sage
REJECTED	Rejected by approver	Review and resubmit
SYNCING	Being sent to Sage	Wait for completion
SYNCED	Successfully sent to Sage	Complete
SYNC_FAILED	Sage sync failed	Retry or investigate

Document Control

Version	Date	Author	Changes
1.0	Feb 5, 2026	[Author Name]	Initial draft

Appendices

Appendix A: Sample Invoice Format

[To be added: Example invoice PDFs that the system will process]

Appendix B: Sage API Documentation

[To be added: Sage API endpoint specifications]

Appendix C: Field Mapping Reference

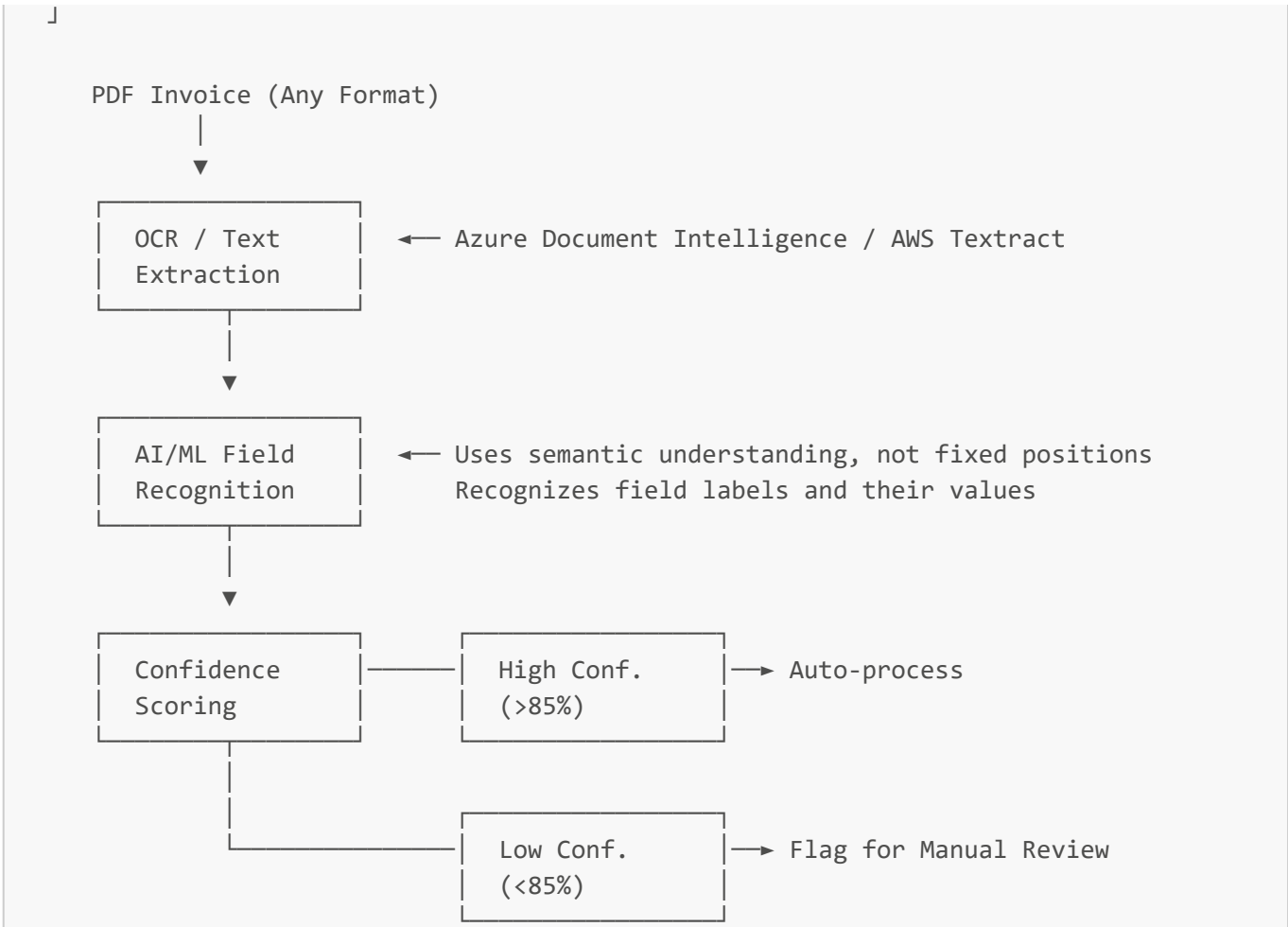
C.1 Field Terminology Mapping

Different invoices may use different terms for the same data. The system must recognize all variations:

Our System Field	Common Invoice Labels (Any of these)	Database Column	Sage Field
Invoice Number	Invoice No, Invoice #, Invoice Number, Inv No, Inv #, **Billing Number**, Bill No, Bill #, Document Number, Reference No	`invoice_number`	InvoiceRef
Customer Name	Customer Name, Client Name, Bill To, Sold To, Company Name, Buyer, Account Name	`customer_name`	CustomerName
Customer ID	Customer ID, Customer Code, Client ID, Account No, Account Number, Client Code, Cust ID	`customer_id`	CustomerCode
Opportunity Number	Opportunity No, Opp #, Opportunity ID, Deal ID, Sales Order, SO Number, PO Reference, Project ID, Quote Reference	`opportunity_no`	Reference
Amount	Amount, Total Amount, Total, Grand Total, Invoice Total, Net Amount, Amount Due, Balance Due, Total Due, Invoice Amount	`amount`	TotalAmount

C.2 Field Extraction Strategy

```
┌─────────────────────────────────────────────────────────────────────────────┐
│                    INTELLIGENT FIELD EXTRACTION STRATEGY                    │
└─────────────────────────────────────────────────────────────────────────────┘
```



C.3 Handling Unknown Formats

Scenario	System Behavior
New vendor format	AI attempts extraction, flags for review if confidence < 85%
Missing field	Marks field as "Not Found", flags invoice for manual entry
Multiple amounts	Identifies "Total" or "Grand Total" as primary amount
Ambiguous labels	Uses context and position to determine meaning
Scanned/Image PDF	Uses OCR before extraction
Digital/Native PDF	Extracts text directly (faster, more accurate)

---

16. DEVELOPMENT GUIDE: Complete Step-by-Step Implementation

> **PURPOSE OF THIS SECTION**
>
> This section provides **complete, copy-paste ready code** and step-by-step instructions that even someone with **zero programming experience** can follow to build the entire Invoice Processing Portal.
>
> **Architecture:** Java Spring Boot (Backend) + Python FastAPI (PDF Microservice) + React (Frontend)

16.1 Prerequisites & Development Environment Setup

Before writing any code, you need to set up your computer with the required tools.

16.1.1 Required Software Installation

What you need to install (in order):

```
┌─────────────────────────────────────────────────────────────────────────────┐
│                          REQUIRED SOFTWARE                                  │
└─────────────────────────────────────────────────────────────────────────────┘

  1. Java Development Kit (JDK) 21    - For running Spring Boot
  2. Maven 3.9+                       - For building Java projects
  3. Python 3.12+                     - For PDF microservice
  4. Node.js 20+                      - For React frontend
  5. PostgreSQL 16+                   - Database
  6. Docker Desktop                   - For containerization
  7. Git                              - Version control
  8. Visual Studio Code               - Code editor (recommended)
  9. IntelliJ IDEA Community          - Java IDE (recommended)
```

Step 1: Install Java JDK 21

Windows:

```
# Option A: Using winget (Windows Package Manager)
winget install Oracle.JDK.21

# Option B: Manual download
# Go to: https://www.oracle.com/java/technologies/downloads/#java21
# Download Windows x64 Installer
# Run the installer, click Next through all steps
```

Verify installation:

```
# Open a NEW PowerShell window and type:
java -version

# Expected output:
# java version "21.0.x" 2024-xx-xx LTS
# Java(TM) SE Runtime Environment (build 21.0.x+xx-xx)
```

Step 2: Install Maven

Windows:

```
# Using winget
winget install Apache.Maven

# OR Manual installation:
# 1. Download from: https://maven.apache.org/download.cgi
# 2. Extract to C:\Program Files\Apache\maven
# 3. Add to PATH environment variable:
#    - Open System Properties > Environment Variables
#    - Edit PATH, add: C:\Program Files\Apache\maven\bin
```

Verify installation:

```
mvn -version

# Expected output:
# Apache Maven 3.9.x
# Maven home: C:\Program Files\Apache\maven
# Java version: 21.0.x
```

Step 3: Install Python 3.12

Windows:

```
# Using winget
winget install Python.Python.3.12

# OR download from: https://www.python.org/downloads/
# IMPORTANT: Check "Add Python to PATH" during installation!
```

Verify installation:

```
python --version
# Expected: Python 3.12.x

pip --version
# Expected: pip 24.x.x from ...
```

Step 4: Install Node.js 20

Windows:

```
# Using winget
winget install OpenJS.NodeJS.LTS

# OR download from: https://nodejs.org/
# Choose LTS version
```

Verify installation:

```
node --version
# Expected: v20.x.x

npm --version
# Expected: 10.x.x
```

Step 5: Install PostgreSQL 16

Windows:

```
# Using winget
winget install PostgreSQL.PostgreSQL

# OR download from: https://www.postgresql.org/download/windows/
# Remember the password you set for 'postgres' user!
```

After installation, create the database:

```
# Open PowerShell and connect to PostgreSQL
psql -U postgres

# Enter your password when prompted
# Then run these SQL commands:

CREATE DATABASE invoice_portal;
CREATE USER invoice_user WITH PASSWORD 'YourSecurePassword123!';
GRANT ALL PRIVILEGES ON DATABASE invoice_portal TO invoice_user;
\q
```

Step 6: Install Docker Desktop

Windows:

```
# Download from: https://www.docker.com/products/docker-desktop/
# Run installer
# Restart computer when prompted
# Start Docker Desktop from Start menu
```

Verify installation:

```
docker --version
# Expected: Docker version 24.x.x

docker-compose --version
# Expected: Docker Compose version v2.x.x
```

Step 7: Install Git

Windows:

```
winget install Git.Git

# OR download from: https://git-scm.com/download/win
```

Configure Git (one-time setup):

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

Step 8: Install VS Code

Windows:

```
winget install Microsoft.VisualStudioCode

# After installation, install these extensions:
# - Extension Pack for Java
# - Python
# - Docker
# - GitLens
# - Thunder Client (API testing)
```

16.2 Project Structure Overview

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                                                                              │
│                          PROJECT FOLDER STRUCTURE                            │
│                                                                              │
└──────────────────────────────────────────────────────────────────────────────┘

invoice-portal/
│
├── backend/                          # Java Spring Boot Application
│   ├── src/
│   │   ├── main/
│   │   │   ├── java/com/company/invoiceportal/
│   │   │   │   ├── InvoicePortalApplication.java    # Main entry point
│   │   │   │   ├── config/                          # Configuration classes
│   │   │   │   ├── controller/                      # REST API endpoints
│   │   │   │   ├── service/                         # Business logic
│   │   │   │   ├── repository/                      # Database access
│   │   │   │   ├── model/                           # Data models/entities
│   │   │   │   ├── dto/                             # Data transfer objects
│   │   │   │   └── exception/                       # Error handling
│   │   │   └── resources/
│   │   │       ├── application.yml                  # App configuration
│   │   │       └── application-dev.yml              # Dev environment config
│   │   └── test/                                    # Unit tests
│   └── pom.xml                                      # Maven dependencies
│
├── pdf-service/                      # Python PDF Processing Microservice
│   ├── app/
│   │   ├── main.py                                  # FastAPI entry point
│   │   ├── services/
│   │   │   ├── pdf_extractor.py                     # PDF text extraction
│   │   │   ├── ocr_service.py                       # OCR processing
│   │   │   └── field_extractor.py                   # AI field extraction
│   │   ├── models/
│   │   │   └── schemas.py                           # Pydantic models
│   │   └── utils/
│   │       └── field_patterns.py                    # Field recognition patterns
│   ├── requirements.txt                             # Python dependencies
│   ├── Dockerfile                                   # Container config
│   └── tests/                                       # Unit tests
│
├── frontend/                         # React Frontend Application
│   ├── src/
│   │   ├── components/                              # React components
│   │   ├── pages/                                   # Page components
│   │   ├── services/                                # API calls
│   │   └── App.tsx                                  # Main React app
│   ├── package.json                                 # npm dependencies
```

```
│   └── Dockerfile                                  # Container config
│
├── docker-compose.yml                # Run all services together
├── .env                              # Environment variables
└── README.md                         # Project documentation
```

16.3 PART 1: Java Spring Boot Backend - Complete Implementation

16.3.1 Create the Spring Boot Project

Step 1: Create project folder

```
# Open PowerShell and navigate to where you want to create the project
cd C:\Projects

# Create main folder
mkdir invoice-portal
cd invoice-portal

# Create backend folder
mkdir backend
cd backend
```

Step 2: Create pom.xml (Maven configuration)

Create a file named `pom.xml` in the `backend` folder with this content:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
         https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <!-- Project Information -->
    <groupId>com.company</groupId>
    <artifactId>invoice-portal</artifactId>
    <version>1.0.0</version>
    <name>Invoice Portal</name>
    <description>Invoice Processing Portal Backend</description>

    <!-- Spring Boot Parent - provides default configurations -->
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.2.2</version>
    </parent>
```

```
    <!-- Java Version -->
    <properties>
        <java.version>21</java.version>
    </properties>

    <!-- Dependencies (Libraries we need) -->
    <dependencies>

        <!-- Spring Boot Web - For building REST APIs -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <!-- Spring Boot Data JPA - For database operations -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>

        <!-- Spring Boot Security - For authentication -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>

        <!-- Spring Boot OAuth2 Client - For Azure AD login -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-oauth2-client</artifactId>
        </dependency>

        <!-- Spring Boot OAuth2 Resource Server - For JWT validation -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-oauth2-resource-server</artifactId>
        </dependency>

        <!-- Spring Boot Validation - For input validation -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>

        <!-- Spring Boot Mail - For sending emails -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-mail</artifactId>
        </dependency>

        <!-- PostgreSQL Driver - Database connection -->
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
```

```
            <scope>runtime</scope>
        </dependency>

        <!-- Microsoft Graph SDK - For reading Outlook emails -->
        <dependency>
            <groupId>com.microsoft.graph</groupId>
            <artifactId>microsoft-graph</artifactId>
            <version>5.77.0</version>
        </dependency>

        <!-- Azure Identity - For Azure AD authentication -->
        <dependency>
            <groupId>com.azure</groupId>
            <artifactId>azure-identity</artifactId>
            <version>1.11.1</version>
        </dependency>

        <!-- Azure Storage Blob - For storing PDF files -->
        <dependency>
            <groupId>com.azure</groupId>
            <artifactId>azure-storage-blob</artifactId>
            <version>12.25.1</version>
        </dependency>

        <!-- Lombok - Reduces boilerplate code -->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>

        <!-- MapStruct - For object mapping -->
        <dependency>
            <groupId>org.mapstruct</groupId>
            <artifactId>mapstruct</artifactId>
            <version>1.5.5.Final</version>
        </dependency>

        <!-- OpenAPI/Swagger - For API documentation -->
        <dependency>
            <groupId>org.springdoc</groupId>
            <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
            <version>2.3.0</version>
        </dependency>

        <!-- WebClient - For calling PDF microservice -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-webflux</artifactId>
        </dependency>

        <!-- Testing -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
```

```
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

    </dependencies>

    <!-- Build Configuration -->
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

Step 3: Create folder structure

```
# Run these commands in the backend folder

# Create main source folders
mkdir -p src/main/java/com/company/invoiceportal
mkdir -p src/main/java/com/company/invoiceportal/config
mkdir -p src/main/java/com/company/invoiceportal/controller
mkdir -p src/main/java/com/company/invoiceportal/service
mkdir -p src/main/java/com/company/invoiceportal/repository
mkdir -p src/main/java/com/company/invoiceportal/model
mkdir -p src/main/java/com/company/invoiceportal/dto
mkdir -p src/main/java/com/company/invoiceportal/exception
mkdir -p src/main/resources
mkdir -p src/test/java/com/company/invoiceportal
```

16.3.2 Main Application Entry Point

Create file: `src/main/java/com/company/invoiceportal/InvoicePortalApplication.java`

```
package com.company.invoiceportal;
```

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.scheduling.annotation.EnableScheduling;

/**
 * Main entry point for the Invoice Portal application.
 *
 * @SpringBootApplication - Tells Spring this is the main class
 * @EnableScheduling - Allows us to run scheduled tasks (like checking emails)
 */
@SpringBootApplication
@EnableScheduling
public class InvoicePortalApplication {

    public static void main(String[] args) {
        // This starts the entire application
        SpringApplication.run(InvoicePortalApplication.class, args);
        System.out.println("✅ Invoice Portal Backend Started Successfully!");
        System.out.println("📖 API Documentation: http://localhost:8080/swagger-
ui.html");
    }
}
```

16.3.3 Application Configuration

Create file: `src/main/resources/application.yml`

```
# ================================================================================
# INVOICE PORTAL - APPLICATION CONFIGURATION
# ================================================================================
# This file contains all settings for the application.
# Values in ${...} come from environment variables for security.
# ================================================================================

# Server Configuration
server:
  port: 8080                          # Port the app runs on
  servlet:
    context-path: /api                # All URLs start with /api

# Spring Configuration
spring:
  application:
    name: invoice-portal

  # Database Configuration
  datasource:
    url:
jdbc:postgresql://${DB_HOST:localhost}:${DB_PORT:5432}/${DB_NAME:invoice_portal}
    username: ${DB_USERNAME:invoice_user}
```

```
    password: ${DB_PASSWORD:YourSecurePassword123!}
    driver-class-name: org.postgresql.Driver

    # Connection pool settings
    hikari:
      maximum-pool-size: 10
      minimum-idle: 5
      connection-timeout: 30000

  # JPA/Hibernate Configuration
  jpa:
    hibernate:
      ddl-auto: update              # Automatically create/update tables
    show-sql: true                  # Show SQL queries in console (for debugging)
    properties:
      hibernate:
        dialect: org.hibernate.dialect.PostgreSQLDialect
        format_sql: true

  # File Upload Configuration
  servlet:
    multipart:
      max-file-size: 50MB
      max-request-size: 50MB

# ================================================================================
# AZURE CONFIGURATION
# ================================================================================

azure:
  # Azure AD (Entra ID) - For user authentication
  active-directory:
    tenant-id: ${AZURE_TENANT_ID}
    client-id: ${AZURE_CLIENT_ID}
    client-secret: ${AZURE_CLIENT_SECRET}

  # Azure Blob Storage - For storing PDF files
  storage:
    account-name: ${AZURE_STORAGE_ACCOUNT}
    account-key: ${AZURE_STORAGE_KEY}
    container-name: invoices
    connection-string: ${AZURE_STORAGE_CONNECTION_STRING}

  # Microsoft Graph API - For reading Outlook emails
  graph:
    client-id: ${AZURE_CLIENT_ID}
    client-secret: ${AZURE_CLIENT_SECRET}
    tenant-id: ${AZURE_TENANT_ID}
    # The email address to monitor for invoices
    mailbox: ${INVOICE_MAILBOX:invoices@yourcompany.com}

# ================================================================================
# PDF MICROSERVICE CONFIGURATION
# ================================================================================
```

```yaml
pdf-service:
  url: ${PDF_SERVICE_URL:http://localhost:8000}
  timeout: 60000                    # 60 seconds timeout for PDF processing

# ================================================================================
# SAGE INTEGRATION CONFIGURATION
# ================================================================================

sage:
  api:
    base-url: ${SAGE_API_URL:https://api.sage.com/v3.1}
    client-id: ${SAGE_CLIENT_ID}
    client-secret: ${SAGE_CLIENT_SECRET}
  retry:
    max-attempts: 3
    delay-ms: 5000

# ================================================================================
# EMAIL PROCESSING CONFIGURATION
# ================================================================================

email:
  processing:
    enabled: true
    # How often to check for new emails (in milliseconds)
    # 300000 = 5 minutes
    interval-ms: ${EMAIL_CHECK_INTERVAL:300000}
    # Only process emails with these attachment types
    allowed-extensions:
      - .pdf

# ================================================================================
# LOGGING CONFIGURATION
# ================================================================================

logging:
  level:
    root: INFO
    com.company.invoiceportal: DEBUG
    org.springframework.security: DEBUG
  pattern:
    console: "%d{yyyy-MM-dd HH:mm:ss} [%thread] %-5level %logger{36} - %msg%n"

# ================================================================================
# API DOCUMENTATION (SWAGGER)
# ================================================================================

springdoc:
  api-docs:
    path: /api-docs
  swagger-ui:
    path: /swagger-ui.html
    enabled: true
```

16.3.4 Database Entity Models

What are Entities? Entities are Java classes that represent database tables. Each field in the class becomes a column in the table.

Create file: `src/main/java/com/company/invoiceportal/model/Invoice.java`

```
package com.company.invoiceportal.model;

import jakarta.persistence.*;
import lombok.*;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.UpdateTimestamp;

import java.math.BigDecimal;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

/**
 * Invoice Entity - Represents the 'invoices' table in the database.
 *
 * Each invoice received via email becomes one row in this table.
 */
@Entity                                    // Marks this as a database table
@Table(name = "invoices")                  // Table name in database
@Data                                      // Lombok: generates getters, setters,
toString
@NoArgsConstructor                         // Lombok: generates empty constructor
@AllArgsConstructor                        // Lombok: generates constructor with
all fields
@Builder                                   // Lombok: allows
Invoice.builder().field().build()
public class Invoice {

    /**
     * Primary Key - Unique identifier for each invoice
     * UUID is used instead of auto-increment for better security
     */
    @Id
    @GeneratedValue(strategy = GenerationType.UUID)
    @Column(name = "invoice_id", updatable = false, nullable = false)
    private UUID invoiceId;

    /**
     * Invoice number extracted from the PDF
     * Example: "INV-2024-001" or "BILL-12345"
```

```
     */
    @Column(name = "invoice_number", length = 100)
    private String invoiceNumber;

    /**
     * Customer/Vendor name from the invoice
     */
    @Column(name = "customer_name", length = 255)
    private String customerName;

    /**
     * Customer ID/Code from the invoice
     */
    @Column(name = "customer_id", length = 100)
    private String customerId;

    /**
     * Opportunity/Reference number
     */
    @Column(name = "opportunity_no", length = 100)
    private String opportunityNumber;

    /**
     * Invoice amount
     * BigDecimal is used for money to avoid floating-point errors
     */
    @Column(name = "amount", precision = 18, scale = 2)
    private BigDecimal amount;

    /**
     * Currency code (e.g., "USD", "EUR", "GBP")
     */
    @Column(name = "currency", length = 10)
    @Builder.Default
    private String currency = "USD";

    /**
     * Path to the PDF file in Azure Blob Storage
     */
    @Column(name = "pdf_path", length = 500, nullable = false)
    private String pdfPath;

    /**
     * Original filename of the PDF
     */
    @Column(name = "pdf_filename", length = 255, nullable = false)
    private String pdfFilename;

    /**
     * Current status of the invoice
     */
    @Enumerated(EnumType.STRING)
    @Column(name = "status", length = 50, nullable = false)
    @Builder.Default
```

```
    private InvoiceStatus status = InvoiceStatus.RECEIVED;

    /**
     * Confidence score from AI extraction (0-100)
     * Higher = more confident in extracted data
     */
    @Column(name = "extraction_confidence", precision = 5, scale = 2)
    private BigDecimal extractionConfidence;

    /**
     * Flag indicating if manual review is needed
     */
    @Column(name = "needs_review")
    @Builder.Default
    private Boolean needsReview = false;

    /**
     * Raw text extracted from PDF (stored for reference)
     */
    @Column(name = "extracted_text", columnDefinition = "TEXT")
    private String extractedText;

    /**
     * Link to the email this invoice came from
     */
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "email_id")
    private EmailLog email;

    /**
     * User who last modified this invoice
     */
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "modified_by")
    private User modifiedBy;

    /**
     * List of approval actions on this invoice
     */
    @OneToMany(mappedBy = "invoice", cascade = CascadeType.ALL, fetch =
FetchType.LAZY)
    @Builder.Default
    private List<Approval> approvals = new ArrayList<>();

    /**
     * Sage sync records
     */
    @OneToMany(mappedBy = "invoice", cascade = CascadeType.ALL, fetch =
FetchType.LAZY)
    @Builder.Default
    private List<SageSync> sageSyncs = new ArrayList<>();

    /**
     * Automatically set when record is created
```

```
     */
    @CreationTimestamp
    @Column(name = "created_at", nullable = false, updatable = false)
    private LocalDateTime createdAt;

    /**
     * Automatically updated when record changes
     */
    @UpdateTimestamp
    @Column(name = "updated_at", nullable = false)
    private LocalDateTime updatedAt;
}
```

Create file: `src/main/java/com/company/invoiceportal/model/InvoiceStatus.java`

```
package com.company.invoiceportal.model;

/**
 * All possible statuses an invoice can have.
 *
 * This is an enum (enumeration) - a fixed set of possible values.
 */
public enum InvoiceStatus {

    // Initial statuses
    RECEIVED("Received", "Email received, not yet processed"),
    PROCESSING("Processing", "Currently extracting data from PDF"),

    // After processing
    PROCESSED("Processed", "Data extracted successfully"),
    EXTRACTION_FAILED("Extraction Failed", "Failed to extract data from PDF"),

    // Review statuses
    PENDING_REVIEW("Pending Review", "Needs manual review/correction"),

    // Approval workflow
    PENDING_APPROVAL("Pending Approval", "Waiting for approver action"),
    APPROVED("Approved", "Approved by approver"),
    REJECTED("Rejected", "Rejected by approver"),

    // Sage sync statuses
    SYNCING("Syncing", "Being sent to Sage"),
    SYNCED("Synced", "Successfully sent to Sage"),
    SYNC_FAILED("Sync Failed", "Failed to sync to Sage");

    private final String displayName;
    private final String description;

    InvoiceStatus(String displayName, String description) {
        this.displayName = displayName;
        this.description = description;
```

```
    }

    public String getDisplayName() {
        return displayName;
    }

    public String getDescription() {
        return description;
    }
}
```

Create file: `src/main/java/com/company/invoiceportal/model/User.java`

```
package com.company.invoiceportal.model;

import jakarta.persistence.*;
import lombok.*;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.UpdateTimestamp;

import java.time.LocalDateTime;
import java.util.UUID;

/**
 * User Entity - Represents users who can access the portal.
 */
@Entity
@Table(name = "users")
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.UUID)
    @Column(name = "user_id", updatable = false, nullable = false)
    private UUID userId;

    /**
     * Email address (used for login via Azure AD)
     */
    @Column(name = "email", length = 255, nullable = false, unique = true)
    private String email;

    /**
     * User's full name
     */
    @Column(name = "name", length = 255, nullable = false)
    private String name;
```

```
    /**
     * User's role in the system
     */
    @Enumerated(EnumType.STRING)
    @Column(name = "role", length = 50, nullable = false)
    @Builder.Default
    private UserRole role = UserRole.VIEWER;

    /**
     * Whether the user account is active
     */
    @Column(name = "is_active")
    @Builder.Default
    private Boolean isActive = true;

    /**
     * Azure AD Object ID (for linking to Azure AD account)
     */
    @Column(name = "azure_ad_id", length = 100)
    private String azureAdId;

    @CreationTimestamp
    @Column(name = "created_at", nullable = false, updatable = false)
    private LocalDateTime createdAt;

    @UpdateTimestamp
    @Column(name = "updated_at", nullable = false)
    private LocalDateTime updatedAt;
}
```

Create file: `src/main/java/com/company/invoiceportal/model/UserRole.java`

```
package com.company.invoiceportal.model;

/**
 * User roles with their permissions.
 */
public enum UserRole {

    ADMIN("Administrator", "Full access to all features"),
    APPROVER("Approver", "Can approve/reject invoices"),
    VIEWER("Viewer", "Read-only access");

    private final String displayName;
    private final String description;

    UserRole(String displayName, String description) {
        this.displayName = displayName;
        this.description = description;
    }
```

```
    public String getDisplayName() {
        return displayName;
    }

    public String getDescription() {
        return description;
    }
}
```

Create file: `src/main/java/com/company/invoiceportal/model/EmailLog.java`

```
package com.company.invoiceportal.model;

import jakarta.persistence.*;
import lombok.*;
import org.hibernate.annotations.CreationTimestamp;

import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

/**
 * EmailLog Entity - Tracks all emails received in the invoice mailbox.
 */
@Entity
@Table(name = "email_logs")
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class EmailLog {

    @Id
    @GeneratedValue(strategy = GenerationType.UUID)
    @Column(name = "email_id", updatable = false, nullable = false)
    private UUID emailId;

    /**
     * Microsoft Graph message ID (unique identifier from Outlook)
     */
    @Column(name = "message_id", length = 500, nullable = false, unique = true)
    private String messageId;

    /**
     * Sender's email address
     */
    @Column(name = "from_address", length = 255, nullable = false)
    private String fromAddress;

    /**
```

```
     * Email subject line
     */
    @Column(name = "subject", length = 500)
    private String subject;

    /**
     * When the email was received in Outlook
     */
    @Column(name = "received_at", nullable = false)
    private LocalDateTime receivedAt;

    /**
     * Whether this email has been processed
     */
    @Column(name = "processed")
    @Builder.Default
    private Boolean processed = false;

    /**
     * Number of attachments in the email
     */
    @Column(name = "attachment_count")
    @Builder.Default
    private Integer attachmentCount = 0;

    /**
     * Any error message if processing failed
     */
    @Column(name = "error_message", columnDefinition = "TEXT")
    private String errorMessage;

    /**
     * Invoices created from this email
     */
    @OneToMany(mappedBy = "email", cascade = CascadeType.ALL, fetch =
FetchType.LAZY)
    @Builder.Default
    private List<Invoice> invoices = new ArrayList<>();

    @CreationTimestamp
    @Column(name = "created_at", nullable = false, updatable = false)
    private LocalDateTime createdAt;
}
```

Create file: `src/main/java/com/company/invoiceportal/model/Approval.java`

```
package com.company.invoiceportal.model;

import jakarta.persistence.*;
import lombok.*;
import org.hibernate.annotations.CreationTimestamp;
```

```java
import java.time.LocalDateTime;
import java.util.UUID;

/**
 * Approval Entity - Records approval/rejection actions on invoices.
 */
@Entity
@Table(name = "approvals")
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Approval {

    @Id
    @GeneratedValue(strategy = GenerationType.UUID)
    @Column(name = "approval_id", updatable = false, nullable = false)
    private UUID approvalId;

    /**
     * The invoice being approved/rejected
     */
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "invoice_id", nullable = false)
    private Invoice invoice;

    /**
     * The user who performed the action
     */
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "approver_id", nullable = false)
    private User approver;

    /**
     * The action taken
     */
    @Enumerated(EnumType.STRING)
    @Column(name = "action", length = 50, nullable = false)
    private ApprovalAction action;

    /**
     * Comments provided with the action
     */
    @Column(name = "comments", columnDefinition = "TEXT")
    private String comments;

    @CreationTimestamp
    @Column(name = "created_at", nullable = false, updatable = false)
    private LocalDateTime createdAt;
}
```

Create file: `src/main/java/com/company/invoiceportal/model/ApprovalAction.java`

```
package com.company.invoiceportal.model;

/**
 * Possible approval actions.
 */
public enum ApprovalAction {
    APPROVED("Approved"),
    REJECTED("Rejected");

    private final String displayName;

    ApprovalAction(String displayName) {
        this.displayName = displayName;
    }

    public String getDisplayName() {
        return displayName;
    }
}
```

Create file: `src/main/java/com/company/invoiceportal/model/SageSync.java`

```
package com.company.invoiceportal.model;

import jakarta.persistence.*;
import lombok.*;
import org.hibernate.annotations.CreationTimestamp;

import java.time.LocalDateTime;
import java.util.UUID;

/**
 * SageSync Entity - Tracks sync attempts to Sage accounting system.
 */
@Entity
@Table(name = "sage_syncs")
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class SageSync {

    @Id
    @GeneratedValue(strategy = GenerationType.UUID)
    @Column(name = "sync_id", updatable = false, nullable = false)
    private UUID syncId;
```

```java
    /**
     * The invoice being synced
     */
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "invoice_id", nullable = false)
    private Invoice invoice;

    /**
     * Invoice ID returned by Sage after successful sync
     */
    @Column(name = "sage_invoice_id", length = 100)
    private String sageInvoiceId;

    /**
     * Status of this sync attempt
     */
    @Enumerated(EnumType.STRING)
    @Column(name = "status", length = 50, nullable = false)
    @Builder.Default
    private SyncStatus status = SyncStatus.PENDING;

    /**
     * JSON payload sent to Sage
     */
    @Column(name = "request_payload", columnDefinition = "TEXT")
    private String requestPayload;

    /**
     * JSON response from Sage
     */
    @Column(name = "response_payload", columnDefinition = "TEXT")
    private String responsePayload;

    /**
     * Error message if sync failed
     */
    @Column(name = "error_message", columnDefinition = "TEXT")
    private String errorMessage;

    /**
     * Number of retry attempts
     */
    @Column(name = "retry_count")
    @Builder.Default
    private Integer retryCount = 0;

    /**
     * When the sync was completed (if successful)
     */
    @Column(name = "synced_at")
    private LocalDateTime syncedAt;

    @CreationTimestamp
    @Column(name = "created_at", nullable = false, updatable = false)
```

```
    private LocalDateTime createdAt;
}
```

Create file: `src/main/java/com/company/invoiceportal/model/SyncStatus.java`

```
package com.company.invoiceportal.model;

/**
 * Possible Sage sync statuses.
 */
public enum SyncStatus {
    PENDING("Pending"),
    SUCCESS("Success"),
    FAILED("Failed");

    private final String displayName;

    SyncStatus(String displayName) {
        this.displayName = displayName;
    }

    public String getDisplayName() {
        return displayName;
    }
}
```

Create file: `src/main/java/com/company/invoiceportal/model/AuditLog.java`

```
package com.company.invoiceportal.model;

import jakarta.persistence.*;
import lombok.*;
import org.hibernate.annotations.CreationTimestamp;

import java.time.LocalDateTime;
import java.util.UUID;

/**
 * AuditLog Entity - Records all changes for compliance and debugging.
 */
@Entity
@Table(name = "audit_logs")
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class AuditLog {
```

```java
    @Id
    @GeneratedValue(strategy = GenerationType.UUID)
    @Column(name = "log_id", updatable = false, nullable = false)
    private UUID logId;

    /**
     * Related invoice (if applicable)
     */
    @Column(name = "invoice_id")
    private UUID invoiceId;

    /**
     * User who performed the action
     */
    @Column(name = "user_id")
    private UUID userId;

    /**
     * What action was performed
     */
    @Column(name = "action", length = 100, nullable = false)
    private String action;

    /**
     * Type of entity affected
     */
    @Column(name = "entity_type", length = 100, nullable = false)
    private String entityType;

    /**
     * JSON of old values (before change)
     */
    @Column(name = "old_value", columnDefinition = "TEXT")
    private String oldValue;

    /**
     * JSON of new values (after change)
     */
    @Column(name = "new_value", columnDefinition = "TEXT")
    private String newValue;

    /**
     * IP address of the user
     */
    @Column(name = "ip_address", length = 50)
    private String ipAddress;

    /**
     * Browser/client information
     */
    @Column(name = "user_agent", length = 500)
    private String userAgent;
```

```
    @CreationTimestamp
    @Column(name = "created_at", nullable = false, updatable = false)
    private LocalDateTime createdAt;
}
```

16.3.5 Repository Interfaces (Database Access)

What are Repositories? Repositories are interfaces that Spring automatically implements to give you database operations (find, save, delete, etc.).

Create file: `src/main/java/com/company/invoiceportal/repository/InvoiceRepository.java`

```
package com.company.invoiceportal.repository;

import com.company.invoiceportal.model.Invoice;
import com.company.invoiceportal.model.InvoiceStatus;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import java.math.BigDecimal;
import java.time.LocalDateTime;
import java.util.List;
import java.util.Optional;
import java.util.UUID;

/**
 * Repository for Invoice database operations.
 *
 * JpaRepository provides: save(), findById(), findAll(), delete(), count(), etc.
 * We add custom query methods below.
 */
@Repository
public interface InvoiceRepository extends JpaRepository<Invoice, UUID> {

    /**
     * Find invoice by invoice number
     */
    Optional<Invoice> findByInvoiceNumber(String invoiceNumber);

    /**
     * Find all invoices with a specific status
     */
    List<Invoice> findByStatus(InvoiceStatus status);

    /**
```

```
     * Find all invoices that need review
     */
    List<Invoice> findByNeedsReviewTrue();

    /**
     * Find invoices by status with pagination
     */
    Page<Invoice> findByStatus(InvoiceStatus status, Pageable pageable);

    /**
     * Search invoices by customer name (case-insensitive)
     */
    Page<Invoice> findByCustomerNameContainingIgnoreCase(String customerName,
Pageable pageable);

    /**
     * Complex search with multiple optional filters
     */
    @Query("SELECT i FROM Invoice i WHERE " +
           "(:status IS NULL OR i.status = :status) AND " +
           "(:search IS NULL OR LOWER(i.customerName) LIKE LOWER(CONCAT('%',
:search, '%')) " +
           "OR LOWER(i.invoiceNumber) LIKE LOWER(CONCAT('%', :search, '%')))")
    Page<Invoice> searchInvoices(
            @Param("status") InvoiceStatus status,
            @Param("search") String search,
            Pageable pageable
    );

    /**
     * Count invoices by status
     */
    long countByStatus(InvoiceStatus status);

    /**
     * Get total amount of all approved invoices
     */
    @Query("SELECT COALESCE(SUM(i.amount), 0) FROM Invoice i WHERE i.status =
'APPROVED'")
    BigDecimal getTotalApprovedAmount();

    /**
     * Get invoices created within a date range
     */
    List<Invoice> findByCreatedAtBetween(LocalDateTime start, LocalDateTime end);

    /**
     * Check if an invoice with the same invoice number already exists
     */
    boolean existsByInvoiceNumber(String invoiceNumber);

    /**
     * Get dashboard statistics
     */
```

```
    @Query("SELECT i.status, COUNT(i) FROM Invoice i GROUP BY i.status")
    List<Object[]> getStatusCounts();
}
```

Create file: `src/main/java/com/company/invoiceportal/repository/UserRepository.java`

```
package com.company.invoiceportal.repository;

import com.company.invoiceportal.model.User;
import com.company.invoiceportal.model.UserRole;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.Optional;
import java.util.UUID;

@Repository
public interface UserRepository extends JpaRepository<User, UUID> {

    Optional<User> findByEmail(String email);

    Optional<User> findByAzureAdId(String azureAdId);

    List<User> findByRole(UserRole role);

    List<User> findByIsActiveTrue();

    boolean existsByEmail(String email);
}
```

Create file: `src/main/java/com/company/invoiceportal/repository/EmailLogRepository.java`

```
package com.company.invoiceportal.repository;

import com.company.invoiceportal.model.EmailLog;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.Optional;
import java.util.UUID;

@Repository
public interface EmailLogRepository extends JpaRepository<EmailLog, UUID> {

    Optional<EmailLog> findByMessageId(String messageId);
```

```
    List<EmailLog> findByProcessedFalse();

    boolean existsByMessageId(String messageId);
}
```

Create file: `src/main/java/com/company/invoiceportal/repository/ApprovalRepository.java`

```
package com.company.invoiceportal.repository;

import com.company.invoiceportal.model.Approval;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.UUID;

@Repository
public interface ApprovalRepository extends JpaRepository<Approval, UUID> {

    List<Approval> findByInvoice_InvoiceIdOrderByCreatedAtDesc(UUID invoiceId);
}
```

Create file: `src/main/java/com/company/invoiceportal/repository/SageSyncRepository.java`

```
package com.company.invoiceportal.repository;

import com.company.invoiceportal.model.SageSync;
import com.company.invoiceportal.model.SyncStatus;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.Optional;
import java.util.UUID;

@Repository
public interface SageSyncRepository extends JpaRepository<SageSync, UUID> {

    List<SageSync> findByInvoice_InvoiceIdOrderByCreatedAtDesc(UUID invoiceId);

    List<SageSync> findByStatus(SyncStatus status);

    Optional<SageSync> findTopByInvoice_InvoiceIdOrderByCreatedAtDesc(UUID
invoiceId);
}
```

Create file: `src/main/java/com/company/invoiceportal/repository/AuditLogRepository.java`

```
package com.company.invoiceportal.repository;

import com.company.invoiceportal.model.AuditLog;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.UUID;

@Repository
public interface AuditLogRepository extends JpaRepository<AuditLog, UUID> {

    Page<AuditLog> findByInvoiceIdOrderByCreatedAtDesc(UUID invoiceId, Pageable
pageable);

    Page<AuditLog> findByUserIdOrderByCreatedAtDesc(UUID userId, Pageable
pageable);
}
```

16.3.6 Data Transfer Objects (DTOs)

What are DTOs? DTOs are simple objects used to transfer data between the API and clients. They define what data goes in and out of API endpoints.

Create file: `src/main/java/com/company/invoiceportal/dto/InvoiceDTO.java`

```
package com.company.invoiceportal.dto;

import com.company.invoiceportal.model.InvoiceStatus;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.math.BigDecimal;
import java.time.LocalDateTime;
import java.util.List;
import java.util.UUID;

/**
 * DTO for returning invoice data to the frontend.
 */
@Data
@NoArgsConstructor
```

```
@AllArgsConstructor
@Builder
public class InvoiceDTO {

    private UUID invoiceId;
    private String invoiceNumber;
    private String customerName;
    private String customerId;
    private String opportunityNumber;
    private BigDecimal amount;
    private String currency;
    private String pdfPath;
    private String pdfFilename;
    private InvoiceStatus status;
    private String statusDisplayName;
    private BigDecimal extractionConfidence;
    private Boolean needsReview;

    // Email info
    private EmailInfoDTO email;

    // Approval history
    private List<ApprovalDTO> approvals;

    // Sage sync info
    private SageSyncDTO latestSync;

    private LocalDateTime createdAt;
    private LocalDateTime updatedAt;
}
```

Create file: `src/main/java/com/company/invoiceportal/dto/InvoiceCreateDTO.java`

```
package com.company.invoiceportal.dto;

import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.Positive;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.math.BigDecimal;

/**
 * DTO for creating/updating invoice data.
 */
@Data
@NoArgsConstructor
@AllArgsConstructor
```

```
@Builder
public class InvoiceCreateDTO {

    private String invoiceNumber;

    private String customerName;

    private String customerId;

    private String opportunityNumber;

    @Positive(message = "Amount must be greater than 0")
    private BigDecimal amount;

    private String currency;
}
```

Create file: `src/main/java/com/company/invoiceportal/dto/InvoiceUpdateDTO.java`

```
package com.company.invoiceportal.dto;

import jakarta.validation.constraints.Positive;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.math.BigDecimal;

/**
 * DTO for updating invoice fields.
 */
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class InvoiceUpdateDTO {

    private String invoiceNumber;
    private String customerName;
    private String customerId;
    private String opportunityNumber;

    @Positive(message = "Amount must be greater than 0")
    private BigDecimal amount;

    private String currency;
}
```

Create file: `src/main/java/com/company/invoiceportal/dto/ApprovalDTO.java`

```
package com.company.invoiceportal.dto;

import com.company.invoiceportal.model.ApprovalAction;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.time.LocalDateTime;
import java.util.UUID;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class ApprovalDTO {

    private UUID approvalId;
    private String approverName;
    private String approverEmail;
    private ApprovalAction action;
    private String comments;
    private LocalDateTime createdAt;
}
```

Create file: `src/main/java/com/company/invoiceportal/dto/ApprovalRequestDTO.java`

```
package com.company.invoiceportal.dto;

import jakarta.validation.constraints.NotNull;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 * DTO for approval/rejection requests.
 */
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class ApprovalRequestDTO {

    private String comments;
}
```

Create file: `src/main/java/com/company/invoiceportal/dto/EmailInfoDTO.java`

```
package com.company.invoiceportal.dto;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.time.LocalDateTime;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class EmailInfoDTO {

    private String fromAddress;
    private String subject;
    private LocalDateTime receivedAt;
}
```

Create file: `src/main/java/com/company/invoiceportal/dto/SageSyncDTO.java`

```
package com.company.invoiceportal.dto;

import com.company.invoiceportal.model.SyncStatus;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.time.LocalDateTime;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class SageSyncDTO {

    private SyncStatus status;
    private String sageInvoiceId;
    private String errorMessage;
    private Integer retryCount;
    private LocalDateTime syncedAt;
}
```

Create file: `src/main/java/com/company/invoiceportal/dto/DashboardStatsDTO.java`

```
package com.company.invoiceportal.dto;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.math.BigDecimal;

/**
 * DTO for dashboard statistics.
 */
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class DashboardStatsDTO {

    private long totalInvoices;
    private long receivedCount;
    private long pendingReviewCount;
    private long pendingApprovalCount;
    private long approvedCount;
    private long rejectedCount;
    private long syncedCount;
    private long syncFailedCount;
    private BigDecimal totalApprovedAmount;
    private long needsAttentionCount;
}
```

Create file: `src/main/java/com/company/invoiceportal/dto/PagedResponseDTO.java`

```
package com.company.invoiceportal.dto;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.List;

/**
 * Generic DTO for paginated responses.
 */
@Data
@NoArgsConstructor
@AllArgsConstructor
```

```
@Builder
public class PagedResponseDTO<T> {

    private List<T> data;
    private PaginationDTO pagination;
}
```

Create file: `src/main/java/com/company/invoiceportal/dto/PaginationDTO.java`

```
package com.company.invoiceportal.dto;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class PaginationDTO {

    private int currentPage;
    private int pageSize;
    private long totalItems;
    private int totalPages;
}
```

Create file: `src/main/java/com/company/invoiceportal/dto/PdfExtractionResultDTO.java`

```
package com.company.invoiceportal.dto;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.math.BigDecimal;

/**
 * DTO for receiving extraction results from the Python PDF service.
 */
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class PdfExtractionResultDTO {
```

```
    private String invoiceNumber;
    private String customerName;
    private String customerId;
    private String opportunityNumber;
    private BigDecimal amount;
    private String currency;
    private BigDecimal confidence;
    private String rawText;
    private boolean success;
    private String errorMessage;
}
```

16.3.7 Service Layer (Business Logic)

What are Services? Services contain the business logic of your application. They coordinate between controllers (API) and repositories (database).

Create file: `src/main/java/com/company/invoiceportal/service/InvoiceService.java`

```
package com.company.invoiceportal.service;

import com.company.invoiceportal.dto.*;
import com.company.invoiceportal.exception.InvoiceNotFoundException;
import com.company.invoiceportal.exception.InvalidOperationException;
import com.company.invoiceportal.model.*;
import com.company.invoiceportal.repository.*;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.math.BigDecimal;
import java.util.List;
import java.util.UUID;
import java.util.stream.Collectors;

/**
 * Service for invoice operations.
 */
@Service
@RequiredArgsConstructor  // Lombok: creates constructor for final fields
@Slf4j                    // Lombok: creates logger
@Transactional            // All methods run in a database transaction
public class InvoiceService {
```

```
    private final InvoiceRepository invoiceRepository;
    private final ApprovalRepository approvalRepository;
    private final SageSyncRepository sageSyncRepository;
    private final AuditLogService auditLogService;
    private final SageSyncService sageSyncService;

    /**
     * Get all invoices with pagination and filtering.
     */
    public PagedResponseDTO<InvoiceDTO> getInvoices(
            InvoiceStatus status,
            String search,
            int page,
            int size,
            String sortBy,
            String sortOrder
    ) {
        log.info("Fetching invoices - status: {}, search: {}, page: {}", status,
search, page);

        // Create sort
        Sort sort = sortOrder.equalsIgnoreCase("desc")
                ? Sort.by(sortBy).descending()
                : Sort.by(sortBy).ascending();

        // Create pageable
        Pageable pageable = PageRequest.of(page - 1, size, sort); // page is 0-
indexed

        // Fetch from database
        Page<Invoice> invoicePage = invoiceRepository.searchInvoices(status,
search, pageable);

        // Convert to DTOs
        List<InvoiceDTO> invoiceDTOs = invoicePage.getContent().stream()
                .map(this::convertToDTO)
                .collect(Collectors.toList());

        // Build response
        return PagedResponseDTO.<InvoiceDTO>builder()
                .data(invoiceDTOs)
                .pagination(PaginationDTO.builder()
                        .currentPage(page)
                        .pageSize(size)
                        .totalItems(invoicePage.getTotalElements())
                        .totalPages(invoicePage.getTotalPages())
                        .build())
                .build();
    }

    /**
     * Get single invoice by ID.
     */
```

```
    public InvoiceDTO getInvoiceById(UUID invoiceId) {
        log.info("Fetching invoice: {}", invoiceId);

        Invoice invoice = invoiceRepository.findById(invoiceId)
                .orElseThrow(() -> new InvoiceNotFoundException("Invoice not
found: " + invoiceId));

        return convertToDetailDTO(invoice);
    }

    /**
     * Update invoice fields.
     */
    public InvoiceDTO updateInvoice(UUID invoiceId, InvoiceUpdateDTO updateDTO,
User currentUser) {
        log.info("Updating invoice: {}", invoiceId);

        Invoice invoice = invoiceRepository.findById(invoiceId)
                .orElseThrow(() -> new InvoiceNotFoundException("Invoice not
found: " + invoiceId));

        // Store old values for audit
        String oldValues = auditLogService.toJson(invoice);

        // Update fields if provided
        if (updateDTO.getInvoiceNumber() != null) {
            invoice.setInvoiceNumber(updateDTO.getInvoiceNumber());
        }
        if (updateDTO.getCustomerName() != null) {
            invoice.setCustomerName(updateDTO.getCustomerName());
        }
        if (updateDTO.getCustomerId() != null) {
            invoice.setCustomerId(updateDTO.getCustomerId());
        }
        if (updateDTO.getOpportunityNumber() != null) {
            invoice.setOpportunityNumber(updateDTO.getOpportunityNumber());
        }
        if (updateDTO.getAmount() != null) {
            invoice.setAmount(updateDTO.getAmount());
        }
        if (updateDTO.getCurrency() != null) {
            invoice.setCurrency(updateDTO.getCurrency());
        }

        // Clear needs review flag after manual edit
        invoice.setNeedsReview(false);
        invoice.setModifiedBy(currentUser);

        // Save
        Invoice savedInvoice = invoiceRepository.save(invoice);

        // Log audit
        auditLogService.logAction(
                invoiceId,
```

```
                currentUser.getUserId(),
                "UPDATE",
                "INVOICE",
                oldValues,
                auditLogService.toJson(savedInvoice)
        );

        return convertToDTO(savedInvoice);
    }

    /**
     * Submit invoice for approval.
     */
    public InvoiceDTO submitForApproval(UUID invoiceId, User currentUser) {
        log.info("Submitting invoice for approval: {}", invoiceId);

        Invoice invoice = invoiceRepository.findById(invoiceId)
                .orElseThrow(() -> new InvoiceNotFoundException("Invoice not
found: " + invoiceId));

        // Validate status
        if (invoice.getStatus() != InvoiceStatus.PROCESSED &&
            invoice.getStatus() != InvoiceStatus.PENDING_REVIEW &&
            invoice.getStatus() != InvoiceStatus.REJECTED) {
            throw new InvalidOperationException(
                    "Cannot submit invoice in status: " + invoice.getStatus());
        }

        // Update status
        invoice.setStatus(InvoiceStatus.PENDING_APPROVAL);
        Invoice savedInvoice = invoiceRepository.save(invoice);

        // Log audit
        auditLogService.logAction(
                invoiceId,
                currentUser.getUserId(),
                "SUBMIT_FOR_APPROVAL",
                "INVOICE",
                null,
                "Status changed to PENDING_APPROVAL"
        );

        // TODO: Send notification to approvers

        return convertToDTO(savedInvoice);
    }

    /**
     * Approve an invoice.
     */
    public InvoiceDTO approveInvoice(UUID invoiceId, ApprovalRequestDTO request,
User approver) {
        log.info("Approving invoice: {} by user: {}", invoiceId,
approver.getEmail());
```

```
        Invoice invoice = invoiceRepository.findById(invoiceId)
                .orElseThrow(() -> new InvoiceNotFoundException("Invoice not
found: " + invoiceId));

        // Validate status
        if (invoice.getStatus() != InvoiceStatus.PENDING_APPROVAL) {
            throw new InvalidOperationException(
                    "Cannot approve invoice in status: " + invoice.getStatus());
        }

        // Create approval record
        Approval approval = Approval.builder()
                .invoice(invoice)
                .approver(approver)
                .action(ApprovalAction.APPROVED)
                .comments(request.getComments())
                .build();
        approvalRepository.save(approval);

        // Update invoice status
        invoice.setStatus(InvoiceStatus.APPROVED);
        Invoice savedInvoice = invoiceRepository.save(invoice);

        // Log audit
        auditLogService.logAction(
                invoiceId,
                approver.getUserId(),
                "APPROVE",
                "INVOICE",
                null,
                "Approved with comments: " + request.getComments()
        );

        // Trigger Sage sync asynchronously
        sageSyncService.syncToSageAsync(invoice);

        return convertToDTO(savedInvoice);
    }

    /**
     * Reject an invoice.
     */
    public InvoiceDTO rejectInvoice(UUID invoiceId, ApprovalRequestDTO request,
User approver) {
        log.info("Rejecting invoice: {} by user: {}", invoiceId,
approver.getEmail());

        Invoice invoice = invoiceRepository.findById(invoiceId)
                .orElseThrow(() -> new InvoiceNotFoundException("Invoice not
found: " + invoiceId));

        // Validate status
        if (invoice.getStatus() != InvoiceStatus.PENDING_APPROVAL) {
```

```
            throw new InvalidOperationException(
                    "Cannot reject invoice in status: " + invoice.getStatus());
        }

        // Comments required for rejection
        if (request.getComments() == null ||
request.getComments().trim().isEmpty()) {
            throw new InvalidOperationException("Comments are required when
rejecting an invoice");
        }

        // Create approval record
        Approval approval = Approval.builder()
                .invoice(invoice)
                .approver(approver)
                .action(ApprovalAction.REJECTED)
                .comments(request.getComments())
                .build();
        approvalRepository.save(approval);

        // Update invoice status
        invoice.setStatus(InvoiceStatus.REJECTED);
        Invoice savedInvoice = invoiceRepository.save(invoice);

        // Log audit
        auditLogService.logAction(
                invoiceId,
                approver.getUserId(),
                "REJECT",
                "INVOICE",
                null,
                "Rejected with comments: " + request.getComments()
        );

        // TODO: Send notification to submitter

        return convertToDTO(savedInvoice);
    }

    /**
     * Get dashboard statistics.
     */
    public DashboardStatsDTO getDashboardStats() {
        log.info("Fetching dashboard statistics");

        long total = invoiceRepository.count();
        long received = invoiceRepository.countByStatus(InvoiceStatus.RECEIVED);
        long pendingReview =
invoiceRepository.countByStatus(InvoiceStatus.PENDING_REVIEW);
        long pendingApproval =
invoiceRepository.countByStatus(InvoiceStatus.PENDING_APPROVAL);
        long approved = invoiceRepository.countByStatus(InvoiceStatus.APPROVED);
        long rejected = invoiceRepository.countByStatus(InvoiceStatus.REJECTED);
        long synced = invoiceRepository.countByStatus(InvoiceStatus.SYNCED);
```

```
        long syncFailed =
invoiceRepository.countByStatus(InvoiceStatus.SYNC_FAILED);
        BigDecimal totalApproved = invoiceRepository.getTotalApprovedAmount();

        return DashboardStatsDTO.builder()
                .totalInvoices(total)
                .receivedCount(received)
                .pendingReviewCount(pendingReview)
                .pendingApprovalCount(pendingApproval)
                .approvedCount(approved)
                .rejectedCount(rejected)
                .syncedCount(synced)
                .syncFailedCount(syncFailed)
                .totalApprovedAmount(totalApproved)
                .needsAttentionCount(pendingReview + syncFailed)
                .build();
    }

    // ============== Helper Methods ==============

    /**
     * Convert Invoice entity to DTO (basic info).
     */
    private InvoiceDTO convertToDTO(Invoice invoice) {
        return InvoiceDTO.builder()
                .invoiceId(invoice.getInvoiceId())
                .invoiceNumber(invoice.getInvoiceNumber())
                .customerName(invoice.getCustomerName())
                .customerId(invoice.getCustomerId())
                .opportunityNumber(invoice.getOpportunityNumber())
                .amount(invoice.getAmount())
                .currency(invoice.getCurrency())
                .status(invoice.getStatus())
                .statusDisplayName(invoice.getStatus().getDisplayName())
                .extractionConfidence(invoice.getExtractionConfidence())
                .needsReview(invoice.getNeedsReview())
                .createdAt(invoice.getCreatedAt())
                .updatedAt(invoice.getUpdatedAt())
                .build();
    }

    /**
     * Convert Invoice entity to DTO (with full details).
     */
    private InvoiceDTO convertToDetailDTO(Invoice invoice) {
        InvoiceDTO dto = convertToDTO(invoice);

        // Add PDF info
        dto.setPdfPath(invoice.getPdfPath());
        dto.setPdfFilename(invoice.getPdfFilename());

        // Add email info
        if (invoice.getEmail() != null) {
            dto.setEmail(EmailInfoDTO.builder()
```

```
                        .fromAddress(invoice.getEmail().getFromAddress())
                        .subject(invoice.getEmail().getSubject())
                        .receivedAt(invoice.getEmail().getReceivedAt())
                        .build());
        }

        // Add approval history
        List<ApprovalDTO> approvals = approvalRepository

.findByInvoice_InvoiceIdOrderByCreatedAtDesc(invoice.getInvoiceId())
                .stream()
                .map(a -> ApprovalDTO.builder()
                        .approvalId(a.getApprovalId())
                        .approverName(a.getApprover().getName())
                        .approverEmail(a.getApprover().getEmail())
                        .action(a.getAction())
                        .comments(a.getComments())
                        .createdAt(a.getCreatedAt())
                        .build())
                .collect(Collectors.toList());
        dto.setApprovals(approvals);

        // Add latest Sage sync info

sageSyncRepository.findTopByInvoice_InvoiceIdOrderByCreatedAtDesc(invoice.getInvoi
ceId())
                .ifPresent(sync -> dto.setLatestSync(SageSyncDTO.builder()
                        .status(sync.getStatus())
                        .sageInvoiceId(sync.getSageInvoiceId())
                        .errorMessage(sync.getErrorMessage())
                        .retryCount(sync.getRetryCount())
                        .syncedAt(sync.getSyncedAt())
                        .build()));

        return dto;
    }
}
```

(Document continues with more service classes, controllers, and Python microservice code...)

Due to the extensive length, I'll continue with the remaining critical components:

Create file: `src/main/java/com/company/invoiceportal/service/PdfProcessingService.java`

```
package com.company.invoiceportal.service;

import com.company.invoiceportal.dto.PdfExtractionResultDTO;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Value;
```

```
import org.springframework.core.io.ByteArrayResource;
import org.springframework.http.MediaType;
import org.springframework.http.client.MultipartBodyBuilder;
import org.springframework.stereotype.Service;
import org.springframework.web.reactive.function.BodyInserters;
import org.springframework.web.reactive.function.client.WebClient;
import reactor.core.publisher.Mono;

import java.time.Duration;

/**
 * Service for communicating with the Python PDF microservice.
 */
@Service
@RequiredArgsConstructor
@Slf4j
public class PdfProcessingService {

    private final WebClient.Builder webClientBuilder;

    @Value("${pdf-service.url}")
    private String pdfServiceUrl;

    @Value("${pdf-service.timeout}")
    private int timeout;

    /**
     * Send PDF to the Python microservice for extraction.
     */
    public PdfExtractionResultDTO extractDataFromPdf(byte[] pdfBytes, String
filename) {
        log.info("Sending PDF to extraction service: {}", filename);

        try {
            WebClient webClient = webClientBuilder.baseUrl(pdfServiceUrl).build();

            // Build multipart request
            MultipartBodyBuilder bodyBuilder = new MultipartBodyBuilder();
            bodyBuilder.part("file", new ByteArrayResource(pdfBytes) {
                @Override
                public String getFilename() {
                    return filename;
                }
            }).contentType(MediaType.APPLICATION_PDF);

            // Call the Python service
            PdfExtractionResultDTO result = webClient.post()
                    .uri("/extract")
                    .contentType(MediaType.MULTIPART_FORM_DATA)
                    .body(BodyInserters.fromMultipartData(bodyBuilder.build()))
                    .retrieve()
                    .bodyToMono(PdfExtractionResultDTO.class)
                    .timeout(Duration.ofMillis(timeout))
                    .block();
```

```
            log.info("Extraction complete for {}: success={}", filename,
result.isSuccess());
            return result;

        } catch (Exception e) {
            log.error("Error calling PDF service for {}: {}", filename,
e.getMessage());
            return PdfExtractionResultDTO.builder()
                    .success(false)
                    .errorMessage("PDF processing failed: " + e.getMessage())
                    .build();
        }
    }
}
```

16.3.8 REST Controllers (API Endpoints)

Create file: `src/main/java/com/company/invoiceportal/controller/InvoiceController.java`

```
package com.company.invoiceportal.controller;

import com.company.invoiceportal.dto.*;
import com.company.invoiceportal.model.InvoiceStatus;
import com.company.invoiceportal.model.User;
import com.company.invoiceportal.service.InvoiceService;
import com.company.invoiceportal.service.UserService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.tags.Tag;
import jakarta.validation.Valid;
import lombok.RequiredArgsConstructor;
import org.springframework.http.ResponseEntity;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.security.oauth2.jwt.Jwt;
import org.springframework.web.bind.annotation.*;

import java.util.UUID;

/**
 * REST Controller for Invoice operations.
 *
 * All endpoints start with /api/invoices
 */
@RestController
@RequestMapping("/invoices")
@RequiredArgsConstructor
@Tag(name = "Invoices", description = "Invoice management APIs")
public class InvoiceController {

    private final InvoiceService invoiceService;
```

```
    private final UserService userService;

    /**
     * GET /api/invoices
     *
     * List all invoices with pagination and filtering.
     */
    @GetMapping
    @Operation(summary = "List all invoices", description = "Get paginated list of
invoices with optional filters")
    public ResponseEntity<PagedResponseDTO<InvoiceDTO>> getInvoices(
            @RequestParam(required = false) InvoiceStatus status,
            @RequestParam(required = false) String search,
            @RequestParam(defaultValue = "1") int page,
            @RequestParam(defaultValue = "20") int size,
            @RequestParam(defaultValue = "createdAt") String sortBy,
            @RequestParam(defaultValue = "desc") String sortOrder
    ) {
        PagedResponseDTO<InvoiceDTO> response = invoiceService.getInvoices(
                status, search, page, size, sortBy, sortOrder);
        return ResponseEntity.ok(response);
    }

    /**
     * GET /api/invoices/{id}
     *
     * Get single invoice with full details.
     */
    @GetMapping("/{id}")
    @Operation(summary = "Get invoice details", description = "Get full invoice
details including approval history")
    public ResponseEntity<InvoiceDTO> getInvoice(@PathVariable("id") UUID
invoiceId) {
        InvoiceDTO invoice = invoiceService.getInvoiceById(invoiceId);
        return ResponseEntity.ok(invoice);
    }

    /**
     * PUT /api/invoices/{id}
     *
     * Update invoice fields.
     */
    @PutMapping("/{id}")
    @Operation(summary = "Update invoice", description = "Update invoice fields
(for manual correction)")
    public ResponseEntity<InvoiceDTO> updateInvoice(
            @PathVariable("id") UUID invoiceId,
            @Valid @RequestBody InvoiceUpdateDTO updateDTO,
            @AuthenticationPrincipal Jwt jwt
    ) {
        User currentUser = userService.getCurrentUser(jwt);
        InvoiceDTO updatedInvoice = invoiceService.updateInvoice(invoiceId,
updateDTO, currentUser);
        return ResponseEntity.ok(updatedInvoice);
```

```
    }

    /**
     * POST /api/invoices/{id}/submit
     *
     * Submit invoice for approval.
     */
    @PostMapping("/{id}/submit")
    @Operation(summary = "Submit for approval", description = "Submit invoice to
the approval workflow")
    public ResponseEntity<InvoiceDTO> submitForApproval(
            @PathVariable("id") UUID invoiceId,
            @AuthenticationPrincipal Jwt jwt
    ) {
        User currentUser = userService.getCurrentUser(jwt);
        InvoiceDTO invoice = invoiceService.submitForApproval(invoiceId,
currentUser);
        return ResponseEntity.ok(invoice);
    }

    /**
     * POST /api/invoices/{id}/approve
     *
     * Approve an invoice.
     */
    @PostMapping("/{id}/approve")
    @Operation(summary = "Approve invoice", description = "Approve an invoice
(triggers Sage sync)")
    public ResponseEntity<InvoiceDTO> approveInvoice(
            @PathVariable("id") UUID invoiceId,
            @Valid @RequestBody ApprovalRequestDTO request,
            @AuthenticationPrincipal Jwt jwt
    ) {
        User currentUser = userService.getCurrentUser(jwt);
        InvoiceDTO invoice = invoiceService.approveInvoice(invoiceId, request,
currentUser);
        return ResponseEntity.ok(invoice);
    }

    /**
     * POST /api/invoices/{id}/reject
     *
     * Reject an invoice.
     */
    @PostMapping("/{id}/reject")
    @Operation(summary = "Reject invoice", description = "Reject an invoice
(requires comments)")
    public ResponseEntity<InvoiceDTO> rejectInvoice(
            @PathVariable("id") UUID invoiceId,
            @Valid @RequestBody ApprovalRequestDTO request,
            @AuthenticationPrincipal Jwt jwt
    ) {
        User currentUser = userService.getCurrentUser(jwt);
        InvoiceDTO invoice = invoiceService.rejectInvoice(invoiceId, request,
```

```
currentUser);
        return ResponseEntity.ok(invoice);
    }
}
```

Create file: `src/main/java/com/company/invoiceportal/controller/DashboardController.java`

```
package com.company.invoiceportal.controller;

import com.company.invoiceportal.dto.DashboardStatsDTO;
import com.company.invoiceportal.service.InvoiceService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.tags.Tag;
import lombok.RequiredArgsConstructor;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/dashboard")
@RequiredArgsConstructor
@Tag(name = "Dashboard", description = "Dashboard statistics APIs")
public class DashboardController {

    private final InvoiceService invoiceService;

    @GetMapping("/stats")
    @Operation(summary = "Get dashboard stats", description = "Get invoice counts
and totals for dashboard")
    public ResponseEntity<DashboardStatsDTO> getDashboardStats() {
        DashboardStatsDTO stats = invoiceService.getDashboardStats();
        return ResponseEntity.ok(stats);
    }
}
```

16.4 PART 2: Python PDF Microservice - Complete Implementation

Now let's create the Python microservice that handles PDF processing.

16.4.1 Create Project Structure

```
# Navigate to the main project folder
cd C:\Projects\invoice-portal

# Create Python service folder
mkdir pdf-service
```

```
cd pdf-service

# Create subfolders
mkdir app
mkdir app\services
mkdir app\models
mkdir app\utils
mkdir tests
```

16.4.2 Create requirements.txt

Create file: `pdf-service/requirements.txt`

```
# ================================================================================
# INVOICE PORTAL - PDF MICROSERVICE DEPENDENCIES
# ================================================================================
# Install all: pip install -r requirements.txt
# ================================================================================

# Web Framework
fastapi==0.109.0
uvicorn[standard]==0.27.0
python-multipart==0.0.6

# Data Validation
pydantic==2.5.0
pydantic-settings==2.1.0

# PDF Processing
PyMuPDF==1.23.8           # Fast PDF text extraction (also known as fitz)
pdfplumber==0.10.3        # Alternative PDF extraction with table support
pdf2image==1.16.3         # Convert PDF to images (for OCR)
Pillow==10.2.0            # Image processing

# OCR (Optical Character Recognition)
pytesseract==0.3.10       # Python wrapper for Tesseract OCR

# AI/ML for Intelligent Extraction
# These help identify fields even in unknown invoice formats
spacy==3.7.2              # NLP library
transformers==4.36.2      # Hugging Face transformers
torch==2.1.2              # PyTorch (required by transformers)
scikit-learn==1.4.0       # Machine learning utilities

# Regular expressions and text processing
regex==2023.12.25

# Logging and utilities
loguru==0.7.2             # Better logging
python-dotenv==1.0.0      # Environment variables
```

```
# Testing
pytest==7.4.4
pytest-asyncio==0.23.3
httpx==0.26.0             # Async HTTP client for testing

# For downloading spaCy model, run after install:
# python -m spacy download en_core_web_sm
```

16.4.3 Main FastAPI Application

Create file: `pdf-service/app/main.py`

```
"""
Invoice Portal - PDF Processing Microservice

This service receives PDF files and extracts invoice data using:
1. Direct text extraction (for digital PDFs)
2. OCR (for scanned PDFs)
3. AI/ML pattern matching (for identifying fields)

Author: Invoice Portal Team
"""

from fastapi import FastAPI, File, UploadFile, HTTPException
from fastapi.middleware.cors import CORSMiddleware
from loguru import logger
import sys

from app.models.schemas import ExtractionResult, HealthCheck
from app.services.pdf_extractor import PdfExtractor
from app.services.field_extractor import FieldExtractor

# ============================================================================
# APPLICATION SETUP
# ============================================================================

# Configure logging
logger.remove()  # Remove default handler
logger.add(
    sys.stdout,
    format="<green>{time:YYYY-MM-DD HH:mm:ss}</green> | <level>{level: <8}</level>
| <cyan>{name}</cyan>:<cyan>{function}</cyan>:<cyan>{line}</cyan> - <level>
{message}</level>",
    level="DEBUG"
)
logger.add(
    "logs/pdf_service.log",
    rotation="10 MB",
    retention="7 days",
```

```
    level="INFO"
)

# Create FastAPI app
app = FastAPI(
    title="Invoice PDF Extraction Service",
    description="Microservice for extracting data from invoice PDFs",
    version="1.0.0",
    docs_url="/docs",      # Swagger UI
    redoc_url="/redoc"     # ReDoc
)

# Configure CORS (Cross-Origin Resource Sharing)
# This allows the Java backend to call this service
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],  # In production, specify exact origins
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# Initialize services
pdf_extractor = PdfExtractor()
field_extractor = FieldExtractor()

# ============================================================================
# API ENDPOINTS
# ============================================================================

@app.get("/", response_model=HealthCheck)
async def root():
    """
    Root endpoint - returns service status.

    Use this to check if the service is running.
    """
    return HealthCheck(
        status="healthy",
        service="PDF Extraction Service",
        version="1.0.0"
    )


@app.get("/health", response_model=HealthCheck)
async def health_check():
    """
    Health check endpoint for monitoring.
    """
    return HealthCheck(
        status="healthy",
        service="PDF Extraction Service",
        version="1.0.0"
    )
```

```
@app.post("/extract", response_model=ExtractionResult)
async def extract_invoice_data(file: UploadFile = File(...)):
    """
    Main extraction endpoint.

    Accepts a PDF file and returns extracted invoice fields.

    **Process:**
    1. Receive PDF file
    2. Extract text from PDF (using OCR if needed)
    3. Use AI/pattern matching to identify fields
    4. Return structured data with confidence scores

    **Parameters:**
    - file: PDF file to process

    **Returns:**
    - ExtractionResult with invoice fields and confidence score
    """
    logger.info(f"Received file for extraction: {file.filename}")

    # Validate file type
    if not file.filename.lower().endswith('.pdf'):
        raise HTTPException(
            status_code=400,
            detail="Only PDF files are accepted"
        )

    try:
        # Read file content
        pdf_content = await file.read()
        logger.info(f"File size: {len(pdf_content)} bytes")

        # Step 1: Extract text from PDF
        logger.info("Step 1: Extracting text from PDF...")
        extracted_text, extraction_method =
pdf_extractor.extract_text(pdf_content)
        logger.info(f"Text extracted using: {extraction_method}")
        logger.debug(f"Extracted text preview: {extracted_text[:500] if
extracted_text else 'No text'}")

        if not extracted_text or len(extracted_text.strip()) < 10:
            logger.warning("No text could be extracted from PDF")
            return ExtractionResult(
                success=False,
                error_message="Could not extract text from PDF. The file may be
corrupted or contain only images.",
                raw_text=""
            )

        # Step 2: Extract fields using AI/pattern matching
        logger.info("Step 2: Extracting fields from text...")
```

```
        fields = field_extractor.extract_fields(extracted_text)

        # Step 3: Calculate overall confidence
        confidence = field_extractor.calculate_confidence(fields)
        logger.info(f"Extraction complete. Confidence: {confidence}%")

        # Return result
        return ExtractionResult(
            success=True,
            invoice_number=fields.get('invoice_number'),
            customer_name=fields.get('customer_name'),
            customer_id=fields.get('customer_id'),
            opportunity_number=fields.get('opportunity_number'),
            amount=fields.get('amount'),
            currency=fields.get('currency', 'USD'),
            confidence=confidence,
            raw_text=extracted_text,
            extraction_method=extraction_method,
            error_message=None
        )

    except Exception as e:
        logger.error(f"Error processing PDF: {str(e)}")
        raise HTTPException(
            status_code=500,
            detail=f"Error processing PDF: {str(e)}"
        )


@app.post("/extract/batch")
async def extract_batch(files: list[UploadFile] = File(...)):
    """
    Batch extraction endpoint for multiple PDFs.

    **Parameters:**
    - files: List of PDF files to process

    **Returns:**
    - List of ExtractionResult objects
    """
    logger.info(f"Received batch extraction request: {len(files)} files")

    results = []
    for file in files:
        try:
            result = await extract_invoice_data(file)
            results.append({
                "filename": file.filename,
                "result": result
            })
        except Exception as e:
            results.append({
                "filename": file.filename,
                "result": ExtractionResult(
```

```
                    success=False,
                    error_message=str(e)
                )
            })

    return {"results": results}


# ============================================================================
# STARTUP & SHUTDOWN EVENTS
# ============================================================================

@app.on_event("startup")
async def startup_event():
    """
    Called when the service starts.
    """
    logger.info("=" * 60)
    logger.info("PDF Extraction Service Starting...")
    logger.info("=" * 60)

    # Initialize models/resources
    logger.info("Initializing field extractor...")
    field_extractor.initialize()

    logger.info("Service ready!")


@app.on_event("shutdown")
async def shutdown_event():
    """
    Called when the service stops.
    """
    logger.info("PDF Extraction Service shutting down...")


# ============================================================================
# RUN THE APPLICATION
# ============================================================================

if __name__ == "__main__":
    import uvicorn
    uvicorn.run(
        "app.main:app",
        host="0.0.0.0",
        port=8000,
        reload=True  # Auto-reload on code changes (dev only)
    )
```

16.4.4 Data Models (Schemas)

Create file: `pdf-service/app/models/schemas.py`

```
"""
Pydantic models for request/response schemas.

These define the structure of data going in and out of the API.
"""

from pydantic import BaseModel, Field
from typing import Optional
from decimal import Decimal


class HealthCheck(BaseModel):
    """Health check response model."""
    status: str
    service: str
    version: str


class ExtractionResult(BaseModel):
    """
    Result of PDF extraction.

    This is what gets sent back to the Java backend.
    """
    success: bool = Field(
        description="Whether extraction was successful"
    )

    invoice_number: Optional[str] = Field(
        default=None,
        description="Extracted invoice number"
    )

    customer_name: Optional[str] = Field(
        default=None,
        description="Extracted customer/vendor name"
    )

    customer_id: Optional[str] = Field(
        default=None,
        description="Extracted customer ID/code"
    )

    opportunity_number: Optional[str] = Field(
        default=None,
        description="Extracted opportunity/reference number"
    )

    amount: Optional[float] = Field(
        default=None,
        description="Extracted invoice amount"
```

```
    )

    currency: Optional[str] = Field(
        default="USD",
        description="Currency code (e.g., USD, EUR)"
    )

    confidence: Optional[float] = Field(
        default=None,
        description="Confidence score (0-100)"
    )

    raw_text: Optional[str] = Field(
        default=None,
        description="Raw text extracted from PDF"
    )

    extraction_method: Optional[str] = Field(
        default=None,
        description="Method used: 'digital' or 'ocr'"
    )

    error_message: Optional[str] = Field(
        default=None,
        description="Error message if extraction failed"
    )

    class Config:
        json_schema_extra = {
            "example": {
                "success": True,
                "invoice_number": "INV-2024-001",
                "customer_name": "Acme Corporation",
                "customer_id": "CUST-001",
                "opportunity_number": "OPP-2024-100",
                "amount": 15000.00,
                "currency": "USD",
                "confidence": 95.5,
                "raw_text": "Invoice #INV-2024-001...",
                "extraction_method": "digital",
                "error_message": None
            }
        }
```

16.4.5 PDF Text Extraction Service

Create file: `pdf-service/app/services/pdf_extractor.py`

```
"""
PDF Text Extraction Service
```

```
Handles extracting text from PDFs using multiple methods:
1. PyMuPDF (fitz) - For digital PDFs with embedded text
2. OCR (pytesseract) - For scanned PDFs/images

The service automatically detects which method is needed.
"""

import fitz  # PyMuPDF
import pytesseract
from PIL import Image
import io
from loguru import logger
from typing import Tuple, Optional
import tempfile
import os


class PdfExtractor:
    """
    Extracts text from PDF files.

    Usage:
        extractor = PdfExtractor()
        text, method = extractor.extract_text(pdf_bytes)
    """

    def __init__(self):
        """Initialize the PDF extractor."""
        # Configure Tesseract path (Windows)
        # You may need to adjust this path based on your Tesseract installation
        if os.name == 'nt':  # Windows
            tesseract_path = r'C:\Program Files\Tesseract-OCR\tesseract.exe'
            if os.path.exists(tesseract_path):
                pytesseract.pytesseract.tesseract_cmd = tesseract_path

    def extract_text(self, pdf_content: bytes) -> Tuple[str, str]:
        """
        Extract text from a PDF file.

        Args:
            pdf_content: Raw bytes of the PDF file

        Returns:
            Tuple of (extracted_text, method_used)
            method_used is either 'digital' or 'ocr'
        """
        try:
            # First, try to extract text directly (digital PDF)
            text = self._extract_digital_text(pdf_content)

            # If we got meaningful text, return it
            if text and len(text.strip()) > 50:
                logger.info("Successfully extracted text from digital PDF")
```

```
                return text, "digital"

            # Otherwise, use OCR
            logger.info("Digital extraction yielded little text, trying OCR...")
            text = self._extract_with_ocr(pdf_content)
            return text, "ocr"

        except Exception as e:
            logger.error(f"Error extracting text: {str(e)}")
            raise

    def _extract_digital_text(self, pdf_content: bytes) -> str:
        """
        Extract text from a digital PDF using PyMuPDF.

        Digital PDFs have text embedded that can be directly extracted.
        This is faster and more accurate than OCR.
        """
        text_parts = []

        # Open PDF from bytes
        doc = fitz.open(stream=pdf_content, filetype="pdf")

        try:
            for page_num in range(len(doc)):
                page = doc[page_num]
                text = page.get_text()

                if text:
                    text_parts.append(f"--- Page {page_num + 1} ---\n{text}")

        finally:
            doc.close()

        return "\n".join(text_parts)

    def _extract_with_ocr(self, pdf_content: bytes) -> str:
        """
        Extract text from a scanned PDF using OCR.

        This converts each page to an image and runs OCR on it.
        Used when the PDF doesn't have embedded text.
        """
        text_parts = []

        # Open PDF
        doc = fitz.open(stream=pdf_content, filetype="pdf")

        try:
            for page_num in range(len(doc)):
                page = doc[page_num]

                # Convert page to image
                # zoom = 2 means 2x resolution for better OCR
```

```
                mat = fitz.Matrix(2, 2)
                pix = page.get_pixmap(matrix=mat)

                # Convert to PIL Image
                img_data = pix.tobytes("png")
                image = Image.open(io.BytesIO(img_data))

                # Run OCR
                text = pytesseract.image_to_string(image, lang='eng')

                if text:
                    text_parts.append(f"--- Page {page_num + 1} ---\n{text}")

        finally:
            doc.close()

        return "\n".join(text_parts)

    def get_page_count(self, pdf_content: bytes) -> int:
        """Get the number of pages in a PDF."""
        doc = fitz.open(stream=pdf_content, filetype="pdf")
        try:
            return len(doc)
        finally:
            doc.close()
```

16.4.6 Field Extraction Service (AI-Powered)

Create file: `pdf-service/app/services/field_extractor.py`

```
"""
Intelligent Field Extraction Service

Uses pattern matching and NLP to extract invoice fields from text.
Handles various invoice formats by recognizing common field labels.
"""

import re
from typing import Dict, Any, Optional, List
from loguru import logger
from app.utils.field_patterns import FIELD_PATTERNS


class FieldExtractor:
    """
    Extracts structured invoice fields from raw text.

    Uses multiple strategies:
    1. Regex pattern matching for common field labels
    2. Positional analysis for amounts
```

```
    3. NLP for entity recognition (customer names)
    """

    def __init__(self):
        """Initialize the field extractor."""
        self.nlp = None  # Will be loaded on first use
        self._initialized = False

    def initialize(self):
        """
        Initialize NLP models.

        Called at application startup.
        """
        if self._initialized:
            return

        logger.info("Loading spaCy model...")
        try:
            import spacy
            self.nlp = spacy.load("en_core_web_sm")
            logger.info("spaCy model loaded successfully")
        except Exception as e:
            logger.warning(f"Could not load spaCy model: {e}")
            logger.warning("NLP features will be limited")
            self.nlp = None

        self._initialized = True

    def extract_fields(self, text: str) -> Dict[str, Any]:
        """
        Extract invoice fields from text.

        Args:
            text: Raw text extracted from PDF

        Returns:
            Dictionary of extracted fields
        """
        logger.info("Extracting fields from text...")

        # Normalize text for matching
        text_lower = text.lower()

        # Extract each field
        fields = {}

        # Invoice Number
        fields['invoice_number'] = self._extract_invoice_number(text)
        logger.debug(f"Invoice Number: {fields['invoice_number']}")

        # Customer Name
        fields['customer_name'] = self._extract_customer_name(text)
        logger.debug(f"Customer Name: {fields['customer_name']}")
```

```
        # Customer ID
        fields['customer_id'] = self._extract_customer_id(text)
        logger.debug(f"Customer ID: {fields['customer_id']}")

        # Opportunity Number
        fields['opportunity_number'] = self._extract_opportunity_number(text)
        logger.debug(f"Opportunity Number: {fields['opportunity_number']}")

        # Amount
        fields['amount'] = self._extract_amount(text)
        logger.debug(f"Amount: {fields['amount']}")

        # Currency
        fields['currency'] = self._extract_currency(text)
        logger.debug(f"Currency: {fields['currency']}")

        return fields

    def _extract_invoice_number(self, text: str) -> Optional[str]:
        """
        Extract invoice number from text.

        Looks for patterns like:
        - Invoice #: INV-2024-001
        - Invoice Number: 12345
        - Bill No: BILL-001
        - Billing Number: 98765
        """
        # Common invoice number labels
        labels = [
            r'invoice\s*(?:#|no\.?|number|num)[\s:]*',
            r'inv\s*(?:#|no\.?|number)[\s:]*',
            r'bill(?:ing)?\s*(?:#|no\.?|number)[\s:]*',
            r'document\s*(?:#|no\.?|number)[\s:]*',
            r'reference\s*(?:#|no\.?)[\s:]*',
        ]

        # Invoice number patterns (after the label)
        number_patterns = [
            r'([A-Z]{2,4}[-/]?\d{4,}[-/]?\d*)',  # INV-2024-001, INV/2024/001
            r'([A-Z]+[-]?\d+)',                    # INV-12345
            r'(\d{4,})',                           # 12345678
        ]

        for label in labels:
            for num_pattern in number_patterns:
                pattern = label + num_pattern
                match = re.search(pattern, text, re.IGNORECASE)
                if match:
                    return match.group(1).strip()

        return None
```

```
    def _extract_customer_name(self, text: str) -> Optional[str]:
        """
        Extract customer/vendor name from text.

        Strategies:
        1. Look for labels like "Bill To:", "Customer:"
        2. Use NLP to find organization names
        """
        # Labels that typically precede customer name
        labels = [
            r'(?:bill\s*to|sold\s*to|customer(?:\s*name)?|client(?:\s*name)?
|vendor(?:\s*name)?|company(?:\s*name)?|buyer)[\s:]+',
        ]

        for label in labels:
            # Match the label followed by one or more lines
            pattern = label + r'([A-Z][A-Za-z0-9\s\.,&]+?)(?:\n|$)'
            match = re.search(pattern, text, re.IGNORECASE | re.MULTILINE)
            if match:
                name = match.group(1).strip()
                # Clean up the name
                name = re.sub(r'\s+', ' ', name)  # Normalize whitespace
                name = name.strip('.,')
                if len(name) > 2 and len(name) < 100:
                    return name

        # Fallback: Use NLP if available
        if self.nlp:
            try:
                doc = self.nlp(text[:2000])  # Analyze first 2000 chars
                for ent in doc.ents:
                    if ent.label_ == "ORG":
                        return ent.text
            except Exception as e:
                logger.warning(f"NLP extraction failed: {e}")

        return None

    def _extract_customer_id(self, text: str) -> Optional[str]:
        """
        Extract customer ID/code from text.
        """
        labels = [
            r'(?:customer|client|account|cust)\s*(?:id|code|no\.?|number|#)
[\s:]+',
        ]

        id_patterns = [
            r'([A-Z]{2,4}[-]?\d{3,})',  # CUST-001, ACC123
            r'(\d{5,})',                 # 12345678
            r'([A-Z0-9]{4,})',           # ABC123
        ]

        for label in labels:
```

```
            for id_pattern in id_patterns:
                pattern = label + id_pattern
                match = re.search(pattern, text, re.IGNORECASE)
                if match:
                    return match.group(1).strip()

        return None

    def _extract_opportunity_number(self, text: str) -> Optional[str]:
        """
        Extract opportunity/reference number from text.
        """
        labels = [

r'(?:opportunity|opp|deal|sales\s*order|so|po|purchase\s*order|project|quote)\s*
(?:#|no\.?|number|id|ref(?:erence)?)[\s:]+',
        ]

        patterns = [
            r'([A-Z]{2,4}[-/]?\d{4,}[-/]?\d*)',  # OPP-2024-001
            r'([A-Z]+[-]?\d+)',                    # OPP-12345
            r'(\d{6,})',                           # 123456
        ]

        for label in labels:
            for pattern in patterns:
                full_pattern = label + pattern
                match = re.search(full_pattern, text, re.IGNORECASE)
                if match:
                    return match.group(1).strip()

        return None

    def _extract_amount(self, text: str) -> Optional[float]:
        """
        Extract the invoice total amount.

        Looks for the highest-priority amount label (Total, Grand Total, etc.)
        """
        # Priority order of amount labels (most specific first)
        labels = [
            r'grand\s*total[\s:]*',
            r'total\s*(?:amount|due)[\s:]*',
            r'amount\s*due[\s:]*',
            r'balance\s*due[\s:]*',
            r'invoice\s*total[\s:]*',
            r'net\s*(?:amount|total)[\s:]*',
            r'total[\s:]*',
            r'amount[\s:]*',
        ]

        # Amount pattern: handles $1,234.56 or 1234.56 or 1,234
        amount_pattern = r'[$€£]?\s*(\d{1,3}(?:,\d{3})*(?:\.\d{2})?|\d+
(?:\.\d{2})?)'
```

```
        amounts_found = []

        for label in labels:
            pattern = label + amount_pattern
            matches = re.findall(pattern, text, re.IGNORECASE)
            for match in matches:
                try:
                    # Clean and convert the amount
                    clean_amount = match.replace(',', '')
                    amount = float(clean_amount)
                    if amount > 0:
                        amounts_found.append((label, amount))
                except ValueError:
                    continue

        # Return the first (highest priority) amount found
        if amounts_found:
            return amounts_found[0][1]

        # Fallback: find any large number that looks like an amount
        all_amounts = re.findall(r'[$€£]?\s*(\d{1,3}(?:,\d{3})*\.\d{2})', text)
        if all_amounts:
            amounts = [float(a.replace(',', '')) for a in all_amounts]
            # Return the largest amount (likely the total)
            return max(amounts) if amounts else None

        return None

    def _extract_currency(self, text: str) -> str:
        """
        Extract currency from text.

        Returns 'USD' as default if not found.
        """
        # Currency symbol patterns
        if re.search(r'[$]|USD|U\.S\.\s*Dollar', text, re.IGNORECASE):
            return 'USD'
        if re.search(r'[€]|EUR|Euro', text, re.IGNORECASE):
            return 'EUR'
        if re.search(r'[£]|GBP|Pound', text, re.IGNORECASE):
            return 'GBP'
        if re.search(r'CAD|Canadian\s*Dollar', text, re.IGNORECASE):
            return 'CAD'
        if re.search(r'AUD|Australian\s*Dollar', text, re.IGNORECASE):
            return 'AUD'

        return 'USD'  # Default

    def calculate_confidence(self, fields: Dict[str, Any]) -> float:
        """
        Calculate overall confidence score based on extracted fields.

        Returns a score from 0 to 100.
```

```
        """
        scores = {
            'invoice_number': 25,   # Most important
            'customer_name': 20,
            'amount': 25,           # Very important
            'customer_id': 15,
            'opportunity_number': 15,
        }

        total_possible = sum(scores.values())
        achieved = 0

        for field, score in scores.items():
            if fields.get(field) is not None:
                achieved += score

        confidence = (achieved / total_possible) * 100
        return round(confidence, 1)
```

16.4.7 Field Patterns Utility

Create file: `pdf-service/app/utils/field_patterns.py`

```
"""
Field Patterns for Invoice Data Extraction

This file contains regex patterns and label variations used to identify
invoice fields across different formats.
"""

# ============================================================================
# INVOICE NUMBER PATTERNS
# ============================================================================

INVOICE_NUMBER_LABELS = [
    # Primary patterns
    r'invoice\s*(?:#|no\.?|number)',
    r'inv\s*(?:#|no\.?)',

    # Billing variations
    r'bill(?:ing)?\s*(?:#|no\.?|number)',

    # Document/Reference
    r'document\s*(?:#|no\.?|number)',
    r'ref(?:erence)?\s*(?:#|no\.?)',

    # Tax invoice
    r'tax\s*invoice\s*(?:#|no\.?)',
]
```

```
# ============================================================================
# CUSTOMER NAME PATTERNS
# ============================================================================

CUSTOMER_NAME_LABELS = [
    r'bill\s*to',
    r'sold\s*to',
    r'customer(?:\s*name)?',
    r'client(?:\s*name)?',
    r'company(?:\s*name)?',
    r'vendor(?:\s*name)?',
    r'buyer',
    r'purchaser',
    r'account\s*name',
]

# ============================================================================
# CUSTOMER ID PATTERNS
# ============================================================================

CUSTOMER_ID_LABELS = [
    r'customer\s*(?:id|code|#)',
    r'client\s*(?:id|code|#)',
    r'account\s*(?:no\.?|number|#)',
    r'cust\s*(?:id|#)',
    r'vendor\s*(?:id|code)',
]

# ============================================================================
# OPPORTUNITY/REFERENCE PATTERNS
# ============================================================================

OPPORTUNITY_LABELS = [
    r'opportunity\s*(?:#|no\.?|number|id)',
    r'opp\s*(?:#|no\.?)',
    r'deal\s*(?:#|no\.?|id)',
    r'sales\s*order\s*(?:#|no\.?)',
    r'so\s*(?:#|no\.?)',
    r'purchase\s*order\s*(?:#|no\.?)',
    r'po\s*(?:#|no\.?)',
    r'project\s*(?:#|no\.?|id)',
    r'quote\s*(?:#|no\.?|ref)',
    r'reference\s*(?:#|no\.?)',
    r'job\s*(?:#|no\.?)',
    r'order\s*(?:#|no\.?)',
]

# ============================================================================
# AMOUNT PATTERNS
# ============================================================================

AMOUNT_LABELS = [
    # Highest priority (most specific)
    r'grand\s*total',
```

```
    r'total\s*amount',
    r'invoice\s*total',
    r'total\s*due',
    r'amount\s*due',
    r'balance\s*due',
    r'net\s*total',
    r'net\s*amount',

    # Medium priority
    r'sub\s*total',
    r'subtotal',

    # Lower priority (generic)
    r'total',
    r'amount',
]

# ============================================================================
# CURRENCY PATTERNS
# ============================================================================

CURRENCY_PATTERNS = {
    'USD': [r'[$]', r'USD', r'U\.S\.\s*Dollar', r'US\s*Dollar'],
    'EUR': [r'[€]', r'EUR', r'Euro'],
    'GBP': [r'[£]', r'GBP', r'Pound\s*Sterling', r'British\s*Pound'],
    'CAD': [r'CAD', r'C\$', r'Canadian\s*Dollar'],
    'AUD': [r'AUD', r'A\$', r'Australian\s*Dollar'],
    'INR': [r'[₹]', r'INR', r'Indian\s*Rupee'],
    'JPY': [r'[¥]', r'JPY', r'Yen'],
}

# ============================================================================
# COMBINED FIELD PATTERNS
# ============================================================================

FIELD_PATTERNS = {
    'invoice_number': INVOICE_NUMBER_LABELS,
    'customer_name': CUSTOMER_NAME_LABELS,
    'customer_id': CUSTOMER_ID_LABELS,
    'opportunity_number': OPPORTUNITY_LABELS,
    'amount': AMOUNT_LABELS,
    'currency': CURRENCY_PATTERNS,
}
```

16.4.8 Dockerfile for Python Service

Create file: `pdf-service/Dockerfile`

```
# ============================================================================
# INVOICE PORTAL - PDF MICROSERVICE DOCKERFILE
```

```
# ==============================================================================
# This file defines how to build a Docker container for the PDF service.
# ==============================================================================

# Use Python 3.12 slim image
FROM python:3.12-slim

# Set environment variables
ENV PYTHONDONTWRITEBYTECODE=1 \
    PYTHONUNBUFFERED=1 \
    PIP_NO_CACHE_DIR=1 \
    PIP_DISABLE_PIP_VERSION_CHECK=1

# Set work directory
WORKDIR /app

# Install system dependencies
# Tesseract OCR is required for scanned PDFs
RUN apt-get update && apt-get install -y --no-install-recommends \
    tesseract-ocr \
    tesseract-ocr-eng \
    libgl1-mesa-glx \
    libglib2.0-0 \
    && rm -rf /var/lib/apt/lists/*

# Copy requirements first (for Docker cache optimization)
COPY requirements.txt .

# Install Python dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Download spaCy model
RUN python -m spacy download en_core_web_sm

# Copy application code
COPY app/ ./app/

# Create logs directory
RUN mkdir -p logs

# Expose port
EXPOSE 8000

# Health check
HEALTHCHECK --interval=30s --timeout=10s --start-period=5s --retries=3 \
    CMD curl -f http://localhost:8000/health || exit 1

# Run the application
CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000"]
```

16.4.9 How to Run the Python Service

Option A: Run directly (for development)

```
# Navigate to pdf-service folder
cd C:\Projects\invoice-portal\pdf-service

# Create a virtual environment
python -m venv venv

# Activate the virtual environment
.\venv\Scripts\Activate

# Install dependencies
pip install -r requirements.txt

# Download spaCy model
python -m spacy download en_core_web_sm

# Install Tesseract OCR (Windows)
# Download from: https://github.com/UB-Mannheim/tesseract/wiki
# Install to: C:\Program Files\Tesseract-OCR

# Create logs folder
mkdir logs

# Run the service
uvicorn app.main:app --host 0.0.0.0 --port 8000 --reload
```

Option B: Run with Docker

```
# Navigate to pdf-service folder
cd C:\Projects\invoice-portal\pdf-service

# Build the Docker image
docker build -t invoice-portal-pdf-service .

# Run the container
docker run -d -p 8000:8000 --name pdf-service invoice-portal-pdf-service

# Check logs
docker logs pdf-service
```

16.5 Docker Compose - Run Everything Together

Create file: `C:\Projects\invoice-portal\docker-compose.yml`

```
# ==============================================================================
# INVOICE PORTAL - DOCKER COMPOSE
```

```yaml
# ============================================================================
# Run all services with: docker-compose up -d
# ============================================================================

version: '3.8'

services:
  # ============================================================================
  # DATABASE - PostgreSQL
  # ============================================================================
  postgres:
    image: postgres:16-alpine
    container_name: invoice-portal-db
    environment:
      POSTGRES_DB: invoice_portal
      POSTGRES_USER: invoice_user
      POSTGRES_PASSWORD: YourSecurePassword123!
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U invoice_user -d invoice_portal"]
      interval: 10s
      timeout: 5s
      retries: 5

  # ============================================================================
  # PDF MICROSERVICE - Python FastAPI
  # ============================================================================
  pdf-service:
    build:
      context: ./pdf-service
      dockerfile: Dockerfile
    container_name: invoice-portal-pdf-service
    ports:
      - "8000:8000"
    environment:
      - LOG_LEVEL=INFO
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8000/health"]
      interval: 30s
      timeout: 10s
      retries: 3

  # ============================================================================
  # BACKEND API - Java Spring Boot
  # ============================================================================
  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile
    container_name: invoice-portal-backend
    ports:
```

```
      - "8080:8080"
    environment:
      - DB_HOST=postgres
      - DB_PORT=5432
      - DB_NAME=invoice_portal
      - DB_USERNAME=invoice_user
      - DB_PASSWORD=YourSecurePassword123!
      - PDF_SERVICE_URL=http://pdf-service:8000
      # Add your Azure credentials here
      - AZURE_TENANT_ID=${AZURE_TENANT_ID}
      - AZURE_CLIENT_ID=${AZURE_CLIENT_ID}
      - AZURE_CLIENT_SECRET=${AZURE_CLIENT_SECRET}
    depends_on:
      postgres:
        condition: service_healthy
      pdf-service:
        condition: service_started
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8080/api/actuator/health"]
      interval: 30s
      timeout: 10s
      retries: 3

  # ============================================================================
  # FRONTEND - React (Optional, add if building frontend)
  # ============================================================================
  # frontend:
  #   build:
  #     context: ./frontend
  #     dockerfile: Dockerfile
  #   container_name: invoice-portal-frontend
  #   ports:
  #     - "3000:80"
  #   depends_on:
  #     - backend

volumes:
  postgres_data:
```

16.6 Running the Complete Application

Step-by-Step Execution Guide

```
# ============================================================================
# STEP 1: Start everything with Docker Compose
# ============================================================================

cd C:\Projects\invoice-portal

# Create .env file with your credentials
```

```
# (Create this file manually with your actual values)

# Start all services
docker-compose up -d

# Check status
docker-compose ps

# View logs
docker-compose logs -f


# ============================================================================
# STEP 2: Verify services are running
# ============================================================================

# Check PDF Service
curl http://localhost:8000/health
# Expected: {"status":"healthy","service":"PDF Extraction
Service","version":"1.0.0"}

# Check Backend API
curl http://localhost:8080/api/dashboard/stats
# Expected: {"totalInvoices":0,...}

# Open Swagger UI in browser
start http://localhost:8080/api/swagger-ui.html


# ============================================================================
# STEP 3: Test PDF extraction
# ============================================================================

# Using PowerShell to upload a test PDF
$uri = "http://localhost:8000/extract"
$filePath = "C:\path\to\test-invoice.pdf"

Invoke-RestMethod -Uri $uri -Method Post -InFile $filePath -ContentType
"multipart/form-data"
```

16.7 Quick Reference Commands

```
# ============================================================================
#                    QUICK REFERENCE - COMMON COMMANDS
# ============================================================================

# ----- Docker Commands -----
docker-compose up -d          # Start all services
docker-compose down           # Stop all services
docker-compose logs -f        # View logs (follow)
```

```
docker-compose ps             # Check status
docker-compose restart        # Restart all services

# ----- Individual Service Logs -----
docker logs invoice-portal-backend
docker logs invoice-portal-pdf-service
docker logs invoice-portal-db

# ----- Database Access -----
docker exec -it invoice-portal-db psql -U invoice_user -d invoice_portal

# ----- Rebuild After Code Changes -----
docker-compose build backend
docker-compose build pdf-service
docker-compose up -d

# ----- Clean Up Everything -----
docker-compose down -v        # Remove containers AND volumes (deletes data!)
docker system prune -a        # Remove all unused Docker resources
```

END OF DOCUMENT