"""

Unit tests for PDF document loader.

This module contains comprehensive tests for the PDFDocumentLoader class,

including edge cases and error handling.

"""

import pytest

from pathlib import Path

from unittest.mock import Mock, patch, MagicMock

import tempfile

from src.data\_loader.pdf\_loader import PDFDocumentLoader

from src.utils.exceptions import PDFProcessingError, DataLoadingError

from langchain.schema import Document

class TestPDFDocumentLoader:

"""Test suite for PDFDocumentLoader."""

@pytest.fixture

def pdf\_loader(self):

"""Create a PDFDocumentLoader instance."""

return PDFDocumentLoader(extract\_images=False)

@pytest.fixture

def mock\_pdf\_file(self):

"""Create a mock PDF file path."""

with tempfile.NamedTemporaryFile(suffix='.pdf', delete=False) as tmp:

tmp.write(b'%PDF-1.4 mock content')

return Path(tmp.name)

def test\_initialization(self, pdf\_loader):

"""Test loader initialization."""

assert pdf\_loader.extract\_images is False

assert isinstance(pdf\_loader, PDFDocumentLoader)

def test\_load\_valid\_pdf(self, pdf\_loader, mock\_pdf\_file):

"""Test loading a valid PDF file."""

# Mock PyPDFLoader

with patch('src.data\_loader.pdf\_loader.PyPDFLoader') as mock\_loader\_class:

# Setup mock

mock\_loader = Mock()

mock\_loader.load.return\_value = [

Document(page\_content="Page 1 content", metadata={"page": 0}),

Document(page\_content="Page 2 content", metadata={"page": 1})

]

mock\_loader\_class.return\_value = mock\_loader

# Load PDF

documents = pdf\_loader.load(mock\_pdf\_file)

# Assertions

assert len(documents) == 2

assert all(isinstance(doc, Document) for doc in documents)

assert all(doc.metadata['source\_type'] == 'pdf' for doc in documents)

assert all(doc.metadata['file\_name'] == mock\_pdf\_file.name for doc in documents)

mock\_loader\_class.assert\_called\_once\_with(str(mock\_pdf\_file))

def test\_load\_nonexistent\_file(self, pdf\_loader):

"""Test loading a non-existent file."""

fake\_path = Path("nonexistent.pdf")

with pytest.raises(DataLoadingError, match="File not found"):

pdf\_loader.load(fake\_path)

def test\_load\_non\_pdf\_file(self, pdf\_loader):

"""Test loading a non-PDF file."""

with tempfile.NamedTemporaryFile(suffix='.txt', delete=False) as tmp:

tmp.write(b'Not a PDF')

txt\_path = Path(tmp.name)

with pytest.raises(DataLoadingError, match="Not a PDF file"):

pdf\_loader.load(txt\_path)

def test\_load\_corrupted\_pdf(self, pdf\_loader, mock\_pdf\_file):

"""Test loading a corrupted PDF file."""

with patch('src.data\_loader.pdf\_loader.PyPDFLoader') as mock\_loader\_class:

# Setup mock to raise exception

mock\_loader = Mock()

mock\_loader.load.side\_effect = Exception("PDF is corrupted")

mock\_loader\_class.return\_value = mock\_loader

with pytest.raises(PDFProcessingError, match="Failed to process PDF"):

pdf\_loader.load(mock\_pdf\_file)

def test\_load\_multiple\_success(self, pdf\_loader):

"""Test loading multiple PDF files successfully."""

# Create mock files

with tempfile.TemporaryDirectory() as tmpdir:

pdf\_files = []

for i in range(3):

pdf\_path = Path(tmpdir) / f"test{i}.pdf"

pdf\_path.write\_bytes(b'%PDF-1.4 mock')

pdf\_files.append(pdf\_path)

with patch('src.data\_loader.pdf\_loader.PyPDFLoader') as mock\_loader\_class:

# Setup mock

mock\_loader = Mock()

mock\_loader.load.return\_value = [

Document(page\_content=f"Content {i}", metadata={"page": 0})

]

mock\_loader\_class.return\_value = mock\_loader

# Load multiple PDFs

documents = pdf\_loader.load\_multiple(pdf\_files)

# Assertions

assert len(documents) == 3

assert mock\_loader\_class.call\_count == 3

def test\_load\_multiple\_partial\_failure(self, pdf\_loader):

"""Test loading multiple PDFs with some failures."""

with tempfile.TemporaryDirectory() as tmpdir:

# Create valid PDF

valid\_pdf = Path(tmpdir) / "valid.pdf"

valid\_pdf.write\_bytes(b'%PDF-1.4 valid')

# Create invalid file

invalid\_file = Path(tmpdir) / "invalid.txt"

invalid\_file.write\_text("Not a PDF")

with patch('src.data\_loader.pdf\_loader.PyPDFLoader') as mock\_loader\_class:

# Setup mock

mock\_loader = Mock()

mock\_loader.load.return\_value = [

Document(page\_content="Valid content", metadata={})

]

mock\_loader\_class.return\_value = mock\_loader

# Load with partial failure

documents = pdf\_loader.load\_multiple([valid\_pdf, invalid\_file])

# Should succeed with only valid PDF

assert len(documents) == 1

assert documents[0].page\_content == "Valid content"

def test\_load\_multiple\_all\_fail(self, pdf\_loader):

"""Test loading multiple PDFs where all fail."""

with tempfile.TemporaryDirectory() as tmpdir:

# Create only invalid files

invalid\_files = []

for i in range(2):

invalid\_file = Path(tmpdir) / f"invalid{i}.txt"

invalid\_file.write\_text("Not a PDF")

invalid\_files.append(invalid\_file)

with pytest.raises(DataLoadingError, match="Failed to load any documents"):

pdf\_loader.load\_multiple(invalid\_files)

def test\_validate\_document(self, pdf\_loader):

"""Test document validation."""

# Valid document

valid\_doc = Document(page\_content="This is valid content", metadata={})

assert pdf\_loader.validate\_document(valid\_doc) is True

# Empty content

empty\_doc = Document(page\_content="", metadata={})

assert pdf\_loader.validate\_document(empty\_doc) is False

# Whitespace only

whitespace\_doc = Document(page\_content=" \n\t ", metadata={})

assert pdf\_loader.validate\_document(whitespace\_doc) is False

# Too short content

short\_doc = Document(page\_content="Hi", metadata={})

assert pdf\_loader.validate\_document(short\_doc) is False

def test\_path\_conversion(self, pdf\_loader):

"""Test that string paths are converted to Path objects."""

with patch('src.data\_loader.pdf\_loader.PyPDFLoader') as mock\_loader\_class:

mock\_loader = Mock()

mock\_loader.load.return\_value = [Document(page\_content="Test", metadata={})]

mock\_loader\_class.return\_value = mock\_loader

# Pass string path

with tempfile.NamedTemporaryFile(suffix='.pdf', delete=False) as tmp:

tmp.write(b'%PDF-1.4')

str\_path = str(tmp.name)

documents = pdf\_loader.load(str\_path)

assert len(documents) == 1

@patch('src.data\_loader.pdf\_loader.logger')

def test\_logging(self, mock\_logger, pdf\_loader, mock\_pdf\_file):

"""Test that appropriate logs are generated."""

with patch('src.data\_loader.pdf\_loader.PyPDFLoader') as mock\_loader\_class:

mock\_loader = Mock()

mock\_loader.load.return\_value = [Document(page\_content="Test", metadata={})]

mock\_loader\_class.return\_value = mock\_loader

pdf\_loader.load(mock\_pdf\_file)

# Check that info logs were called

assert mock\_logger.info.called

assert any("Loading PDF document" in str(call) for call in mock\_logger.info.call\_args\_list)