글로벌아카데미

인공지능 활용 PDF

데이터 추출 미니 프로젝트

TEAM 1조

팀장: 신희섬 팀원 : 김근우 , 이인규 , 장태준

[멘토] 김지량 , 김석원

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam vulputate, purus ut tempus fermentum, velit dui efficitur libero, et porta elit. consequat pretium. Nam vulputate nulla ac lacus ornare vulputate. Morbi ornare faucibus eros sagittis maximus. Curabitur cursus molestie. Quisque nec ipsum purus. Duis et mi eros.

고용노동부

KOREATECH 직업능력심사평가원

목차보기

미니프로젝트 프레젠테이션의 목차입니다

01 프로젝트 개요

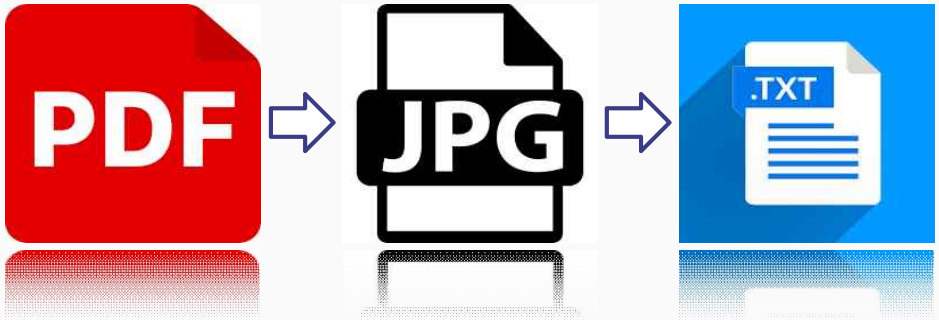
02 프로젝트 팀 구성 및 역할

03 프로젝트 수행 절차 및 방법

04 프로젝트 수행 결과

05 자체 의견

프로젝트 개요



프로젝트 목표

PDF 파일을 업로드하면 txt파일과 json 파일이 생성되고 그 파일을 다운로드 할 수 있도록 한다.

프로젝트 개요

선정 배경 및 기획의도

선정 배경 :

- 업무 및 학습 과정에서 PDF 문서의 텍스트 추출의 필요성을 느낌
- 영어 문서에 비해 한국어 문서에 대한 OCR로 변환하는 프로그램이 부족하다 생각함
- 대량의 PDF 파일을 일괄 처리할 수 있는 효율적인 도구가 필요하다 생각함

기획 의도 :

- PaddleOCR의 우수한 한국어 인식 성능을 활용한 정확한 텍스트 추출
- 드래그 앤 드롭 기반의 직관적인 UI로 사용자 편의성 극대화

프로젝트 개요

핵심 기능

PDF 텍스트 변환 :

- PaddleOCR 한국어 모델 기반 고정밀 텍스트 인식
- 개별 파일 변환 및 폴더 일괄 변환 지원
- 실시간 진행률 표시 (Converting X/Y)

파일 관리 :

- 드래그 앤 드롭 업로드 인터페이스
- 변환 중, 완료, 에러, 취소 상태 관리
- 개별 파일 및 폴더 단위 삭제 기능

프로젝트 개요

핵심 기능

다운로드 기능 :

- TXT 포맷: 순수 텍스트 파일
- JSON 포맷: 메타데이터 포함 (파일명, 텍스트, 크기, 변환 시간)
- ZIP 다운로드: 폴더 내 모든 파일 일괄 다운로드
- 저장 위치 선택: File System Access API 활용

사용자 경험 :

- 폴더 계층 구조 표시 및 확장/축소
- 결과 텍스트 미리보기

프로젝트 개요

UI 설계도

Drag & Drop 형식으로 PDF파일을 추가할 수 있다

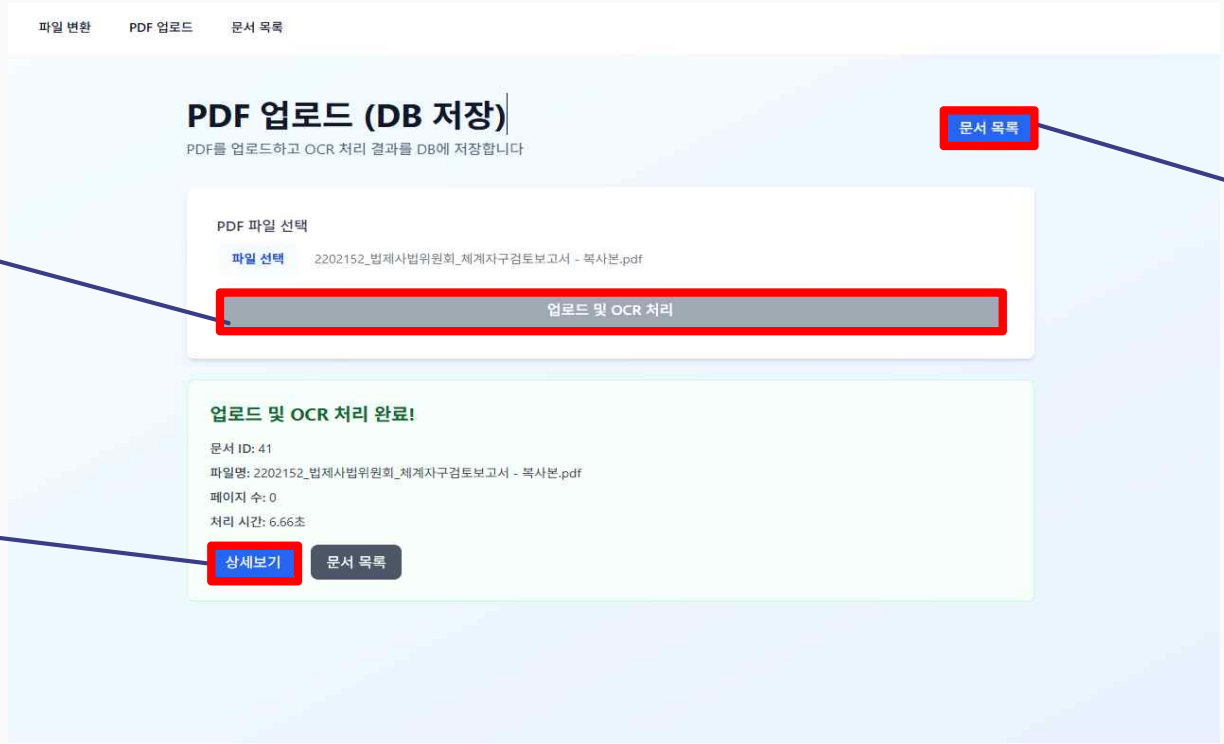
개별 파일이나 폴더를 선택해 PDF파일을 추가할 수 있다

텍스트로 변환된 파일을 보여준다

버튼을 눌러 PDF -> txt로 변환시킬 수 있다

프로젝트 개요

UI 설계도



PaddleOCR 을 통한 텍스트 추출 및 DB 에 저장한다

OCR 처리 결과 및 카테고리 추출, 요약생성 가능한 페이지로 이동한다

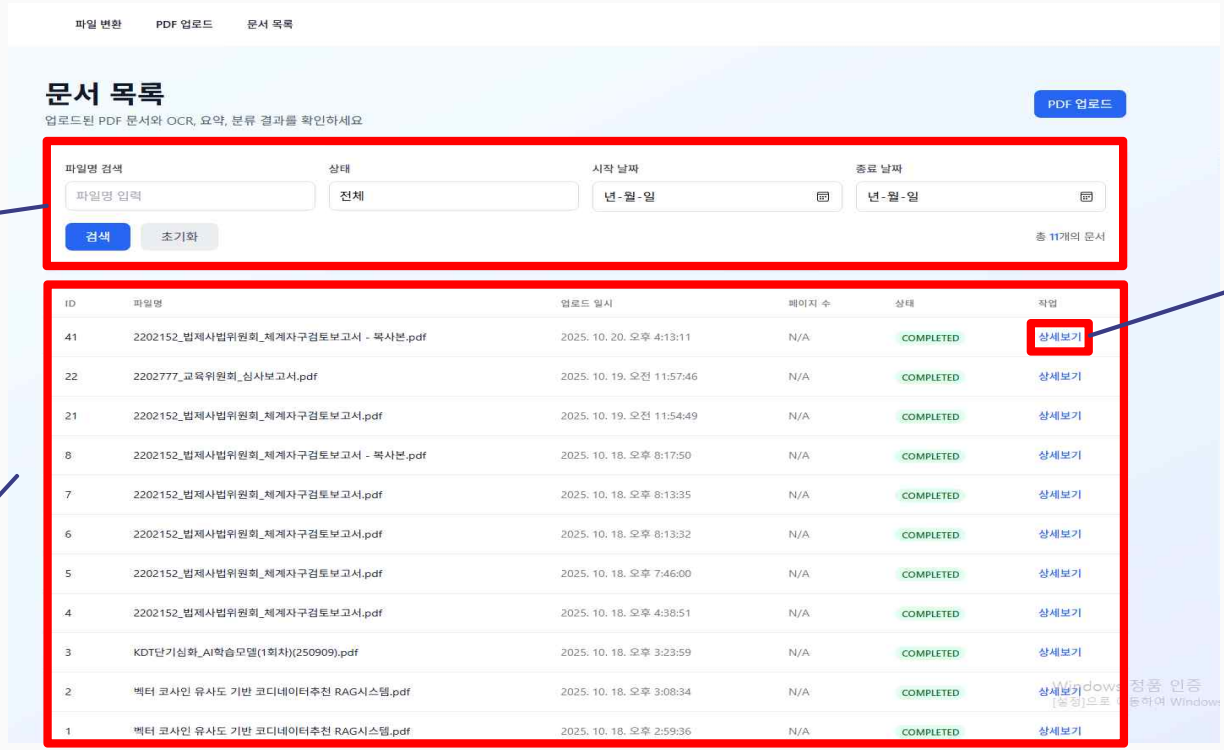
문서 목록으로 이동한다

프로젝트 개요

UI 설계도

조건에 맞는 PDF를 검색한다

DB에 저장되어 있는 ocr돌아간 내용들을 확인 가능하다.



해당 PDF의 상세보기 페이지로 이동한다

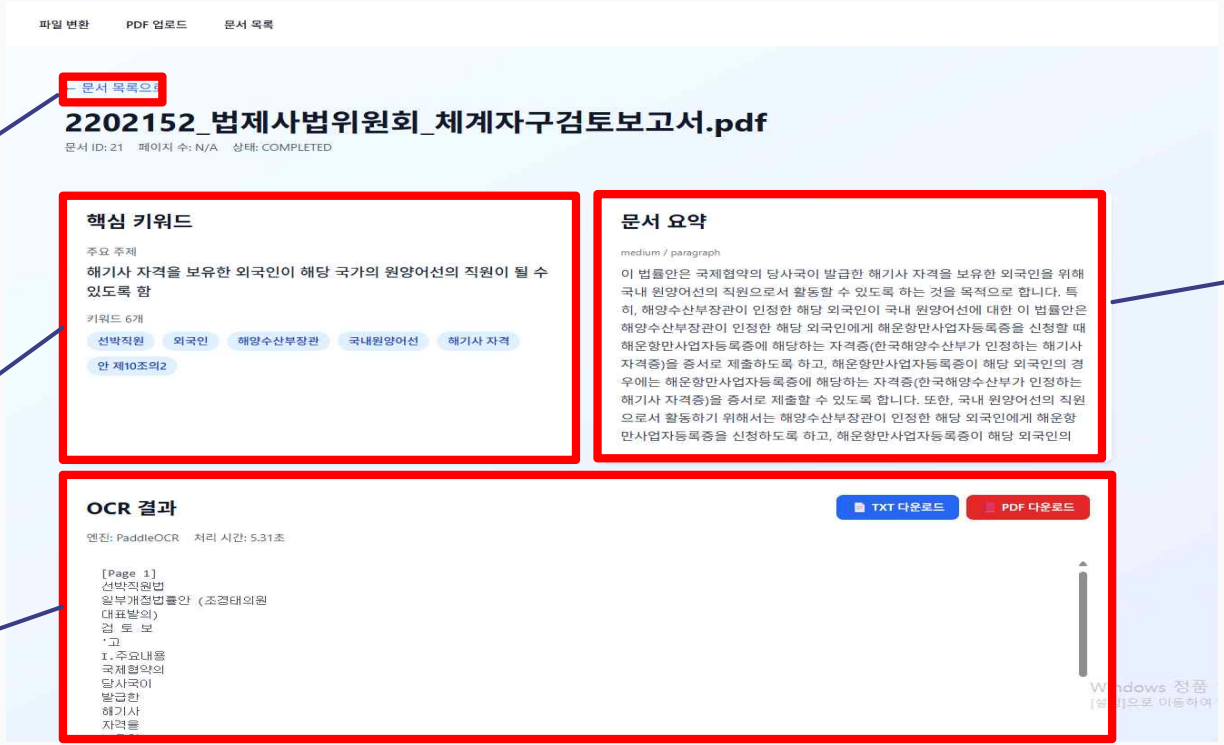
프로젝트 개요

UI 설계도

문서목록으로 돌아간다

gemma를 통해 핵심 키워드들을 추출 및 DB에 저장한다.

OCR이 돌아간 텍스트 랜더링 및 txt, PDF로 다운로드가 가능하다



gemma를 통해 문서요약 및 DB에 저장한다.

프로젝트 개요

활용 방안

교육분야 :
- 스캔된 교재, 강의자료의 텍스트 추출 및 디지털화
- 장애 학생을 위한 접근성 향상 (TTS 연계 가능)

업무 분야 :
- 계약서, 보고서 등 문서 관리 및 검색 시스템 구축
- 대량 문서 아카이빙 자동화

연구 분야 :
- 논문, 학술자료의 텍스트 데이터 추출
- 텍스트 마이닝 및 자연어 처리 전처리

개인 사용 :
- 전자책, 스캔 문서의 텍스트 추출
- 클립보드 복사를 통한 즉시 활용

프로젝트 개요

기대효과

업무 효율성 향상 :
- 수작업 타이핑 대비 시간 절약
- 폴더 일괄 처리로 대량 문서 처리 시간 단축

비용 절감 :
- 무료 오픈소스 기반으로 라이선스 비용 절감
- GPU 가속을 통한 빠른 처리 속도

접근성 개선 :
- 직관적인 웹 인터페이스로 기술 진입 장벽 낮춤
- 별도 설치 없이 브라우저만으로 사용 가능

확장 가능성 :
- API 기반 설계로 타 시스템 연동 용이
- AI 기반 문서 분류, 요약 기능 추가 가능

프로젝트 팀 구성 및 역할

신희성	Paddle OCR 및 EasyOCR의 테스팅 , 프론트를 담당하였습니다.	김근우	kr-ocr 테스팅 및 TrOCR 의 파인튜닝을 담당하였습니다.
이인규	keras-ocr 테스팅, HyperCLOVAX-SEED 테스팅을 담당했습니다.	장태준	Tesseract 테스팅 및 Teseract의 파인튜닝을 담당하였습니다.

프로젝트 수행 절차 및 방법

프로젝트 계획 및 준비

프로젝트 계획 및
라이브러리 및 모델 서칭

09.24 - 09.25

콘텐츠 개발 및 디자인

각자 찾은
라이브러리 및 모델
적용 혹은 파인튜닝

09.26 - 09.29

실행 및 배포작업

최종 AI모델
선택,GITHUB공유 및
발표 자료 준비

09.30 - 10.01

최종평가 및 피드백

최종 수정 및 발표
피드백 반영

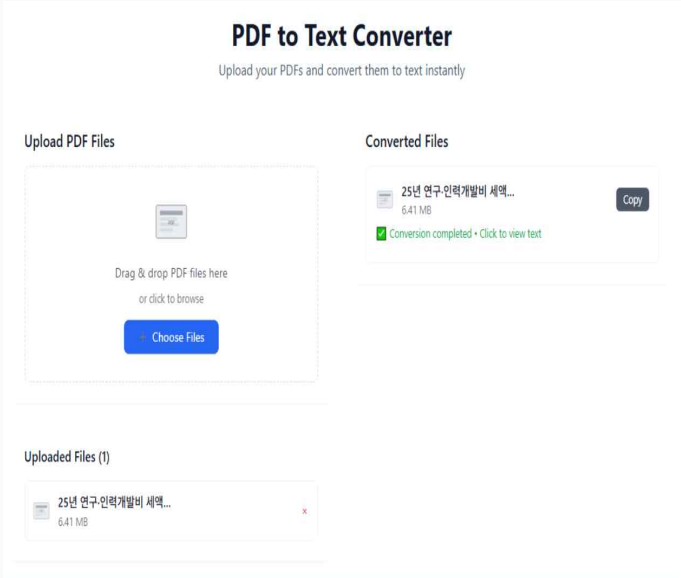
~10.17

프로젝트 수행 절차 및 방법

개발 환경

라이브러리
프레임워크

AI모델



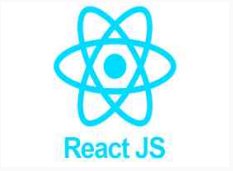
화면구성

프로젝트 수행 절차 및 방법

소프트웨어 구성 요소

프로그래밍 언어
파이썬

라이브러리
리액트,pdf2image,pillow,uvicorn



프레임워크
FastAPI,Paddle,

프로젝트 수행 절차 및 방법

소프트웨어 구성 요소

이름	S/W	버전	프로그램 링크 및 명령어
파이썬	프로그래밍언어	3.11	https://www.python.org/downloads/
paddlepaddle	프레임워크	2.7.0	pip install paddlepaddle-gpu
FastAPI	프레임워크	0.104.1	pip install fastapi

프로젝트 수행 절차 및 방법

소프트웨어 구성 요소

이름	S/W	버전	프로그램 링크 및 명령어
리액트	라이브러리	19.1.1	npm install
pdf2image	라이브러리	1.16.3	pip install pdf2image Pillow
pillow	라이브러리	10.1.0	pip install pillow
uvicorn	라이브러리	0.24.0	pip install uvicorn
python-mutipart	라이브러리	0.0.6	pip install python-multipart

프로젝트 수행 절차 및 방법

디렉터리 구조

```
src/
├── components/
│   ├── UploadArea.tsx          # PDF 파일 업로드 영역
│   ├── UploadedFilesList.tsx   # 업로드된 파일 목록
│   └── ConvertedFilesList.tsx  # 변환된 파일 목록
├── pages/
│   ├── Section.tsx             # 파일 변환 페이지 (File System Access API)
│   ├── UploadPage.tsx          # DB 업로드 페이지
│   ├── DocumentListPage.tsx    # 문서 목록 페이지
│   └── DocumentDetailPage.tsx  # 문서 상세 페이지
├── App.js                      # 메인 라우팅
└── index.js                    # 엔트리 포인트
```

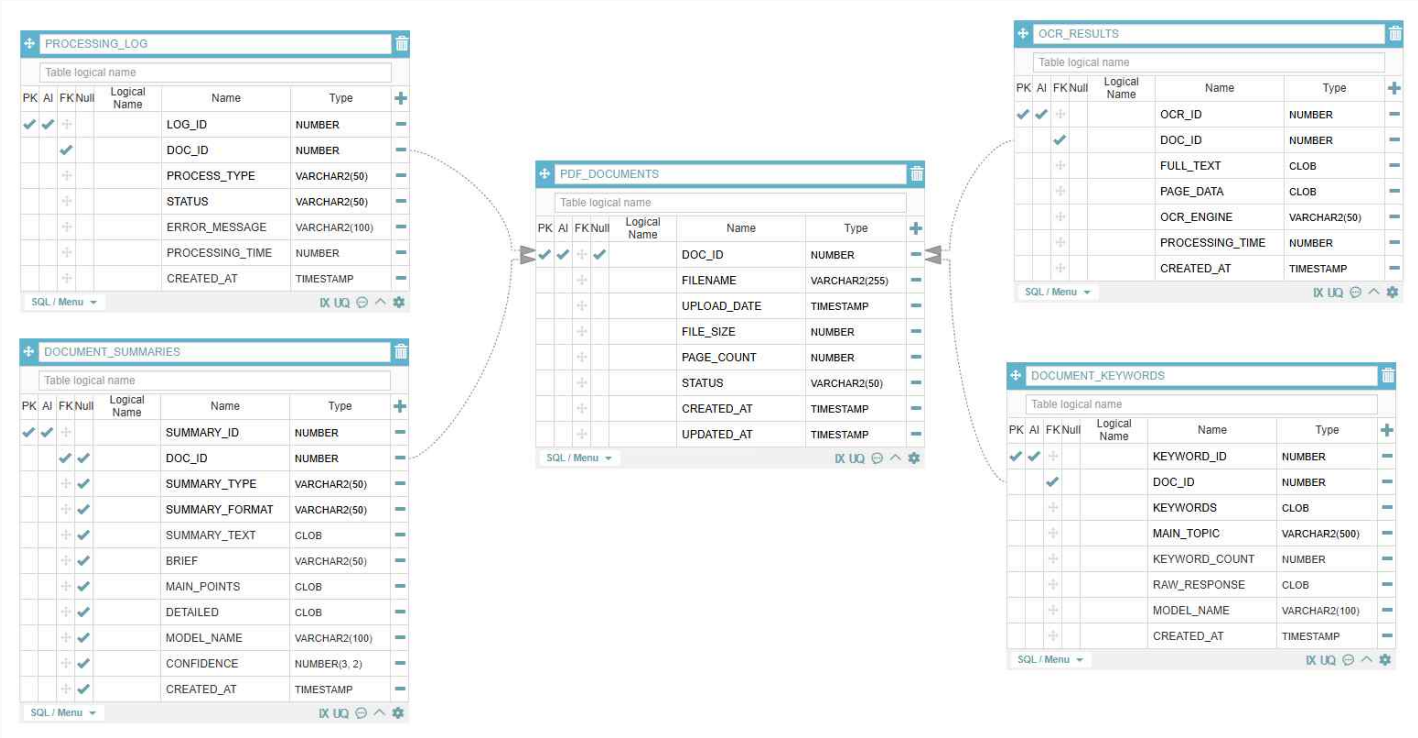
프론트엔드

```
backend/
├── app.py                      # FastAPI 메인 앱
├── config.py                   # 서버 설정 (포트, 경로 등)
├── database.py                 # Oracle DB 연결 관리
├── models.py                   # Pydantic 데이터 모델
├── repository.py               # DB CRUD 레이어
├── routes.py                   # 파일 변환 API
├── routes_db.py                # DB 연동 API
├── pdf_service.py              # PDF → 이미지 변환
├── ocr_service.py              # OCR 처리 (PaddleOCR)
├── summary_service.py          # 텍스트 요약 (Ollama)
├── keyword_extraction_service.py  # 키워드 추출 (Ollama)
├── text_to_pdf_service.py      # 텍스트 → PDF 변환 (reportlab)
└── create_tables.sql           # DB 테이블 생성 스크립트
```

백엔드

프로젝트 수행 절차 및 방법

ERD



프로젝트 수행 절차 및 방법

코드분석(프론트)

Section.tsx

```tsx
<div className="space-y-6">
  <UploadArea
    isDragOver={isDragOver}
    onDragOver={handleDragOver}
    onDragLeave={handleDragLeave}
    onDrop={handleDrop}
    onFileSelect={handleFileSelect}
    onFolderSelect={handleFolderSelect}
  />

  {hasUploadedFiles && (
    <UploadedFilesList
      folderGroups={folderGroups}
      uploadedFiles={uploadedFiles}
      onToggleFolderExpanded={toggleFolderExpanded}
      onConvertFolder={convertFolderToText}
      onRemoveFolder={removeFolder}
      onConvertFolderFile={convertFolderFileToText}
      onRemoveFolderFile={removeFolderFile}
      onConvertFile={convertToText}
      onRemoveFile={removeFile}
    />
  )}
</div>
```

UploadArea.tsx

UploadedFilesList.tsx

ConvertedFilesList.tsx

```tsx
<ConvertedFilesList
  folderGroups={folderGroups}
  uploadedFiles={uploadedFiles}
  expandedConvertedFolders={expandedConvertedFolders}
  expandedConvertedFiles={expandedConvertedFiles}
  selectedFileIndex={selectedFileIndex}
  onToggleConvertedFolder={toggleConvertedFolder}
  onToggleConvertedFile={toggleConvertedFile}
  onCardClick={handleCardClick}
```

usePDFConverter.ts

```ts
const [uploadedFiles, setUploadedFiles] = useState<FileWithText[]>([]);
const [folderGroups, setFolderGroups] = useState<FolderGroup[]>([]);
const [selectedFileIndex, setSelectedFileIndex] = useState<number | null>(null);
const [expandedConvertedFolders, setExpandedConvertedFolders] = useState<Set<number>>(new Set());
const [expandedConvertedFiles, setExpandedConvertedFiles] = useState<Set<string>>(new Set());
```

프로젝트 수행 절차 및 방법

코드분석(프론트엔드) - App.js 전체 라우팅 구조

```jsx
function App() {
  return (
    <Router>
      <div className="App">
        <nav className="bg-white shadow-sm">
          <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
            <div className="flex justify-between h-16">
              <div className="flex space-x-8">
                <Link
                  to="/"
                  className="inline-flex items-center px-1 pt-1 border-b-2 border-transparent hover:border-blue-500 text-sm font-medium text-gray-900"
                >
                  파일 변환
                </Link>
                <Link
                  to="/upload"
                  className="inline-flex items-center px-1 pt-1 border-b-2 border-transparent hover:border-blue-500 text-sm font-medium text-gray-900"
                >
                  DB 업로드
                </Link>
                <Link
                  to="/documents"
                  className="inline-flex items-center px-1 pt-1 border-b-2 border-transparent hover:border-blue-500 text-sm font-medium text-gray-900"
                >
                  문서 목록
                </Link>
              </div>
            </div>
          </div>
        </nav>

        <Routes>
          <Route path="/" element={<Section />} />
          <Route path="/upload" element={<UploadPage />} />
          <Route path="/documents" element={<DocumentListPage />} />
          <Route path="/document/:docId" element={<DocumentDetailPage />} />
        </Routes>
      </div>
    </Router>
```

프로젝트 수행 절차 및 방법

코드분석(프론트엔드) - UploadPage.tsx - DB업로드

```
const handleUpload = async () => {
  if (!selectedFile) return;

  setUploading(true);
  setError(null);
  setResult(null);

  try {
    const formData = new FormData();
    formData.append("pdfFile", selectedFile);

    const response = await fetch(`${API_BASE}/upload`, {
      method: "POST",
      body: formData,
    });

    const data = await response.json();

    if (data.success) {
      setResult(data);
      setSelectedFile(null);
    } else {
      setError(data.detail || "업로드에 실패했습니다.");
    }
  } catch (err) {
    console.error(err);
    setError("서버와 통신할 수 없습니다.");
  } finally {
    setUploading(false);
  }
};
```

PDF를 서버에 업로드하고
OCR 처리 결과를 DB에 저장

- async/await : 비동기 처리
- Content-Type 자동 설정: multipart/form-data
- React의 조건부 랜더링 : && 연산자

프로젝트 수행 절차 및 방법

코드분석(프론트엔드) - DocumentListPage.tsx 문서목록

```
const fetchDocuments = async () => {
  setLoading(true);
  setError(null);
  try {
    // 쿼리 파라미터 구성
    const params = new URLSearchParams({
      limit: "100",
      offset: "0"
    });

    if (searchQuery) params.append("search", searchQuery);
    if (statusFilter) params.append("status", statusFilter);
    if (fromDate) params.append("from_date", fromDate);
    if (toDate) params.append("to_date", toDate);

    const response = await fetch(`${API_BASE}/documents?${params.toString()}`);
    const data = await response.json();

    if (data.success) {
      setDocuments(data.documents);
      setTotalCount(data.total_count);
    } else {
      setError("문서 목록을 불러오는데 실패했습니다.");
    }
  } catch (err) {
    setError("서버와 통신할 수 없습니다.");
    console.error(err);
  } finally {
    setLoading(false);
  }
};
```

DB에 저장된 모든 문서를 페이징하여 조회

- 쿼리 파라미터 : ?limit=100&offset=0
- 테이블 랜더링

프로젝트 수행 절차 및 방법

코드분석(프론트엔드) - DocumentDetailPage.tsx 문서 상세

```
const generateSummary = async () => {
  setProcessing((prev) => ({ ...prev, summary: true }));
  try {
    const response = await fetch(
      `${API_BASE}/summarize/${docId}?summary_type=medium&summary_format=paragraph`,
      { method: "POST" }
    );
    const result = await response.json();

    if (result.success) {
      await fetchDocumentDetail();
    } else {
      alert("요약 생성에 실패했습니다.");
    }
  } catch (err) {
    console.error(err);
    alert("요약 생성 중 오류가 발생했습니다.");
  } finally {
    setProcessing((prev) => ({ ...prev, summary: false }));
  }
};

const extractKeywords = async () => {
  setProcessing((prev) => ({ ...prev, keywords: true }));
  try {
    const response = await fetch(`${API_BASE}/extract-keywords/${docId}?max_keywords=10`, {
      method: "POST",
    });
    const result = await response.json();

    if (result.success) {
      await fetchDocumentDetail();
    } else {
      alert("키워드 추출에 실패했습니다.");
    }
  } catch (err) {
    console.error(err);
    alert("키워드 추출 중 오류가 발생했습니다.");
  } finally {
    setProcessing((prev) => ({ ...prev, keywords: false }));
  }
};
```


문서 상세 정보 , OCR 결과 ,
AI 요약/키워드 처리

- 요약 생성
- 키워드 추출

프로젝트 수행 절차 및 방법

코드분석(백엔드) - app.py FastAPI 메인 앱

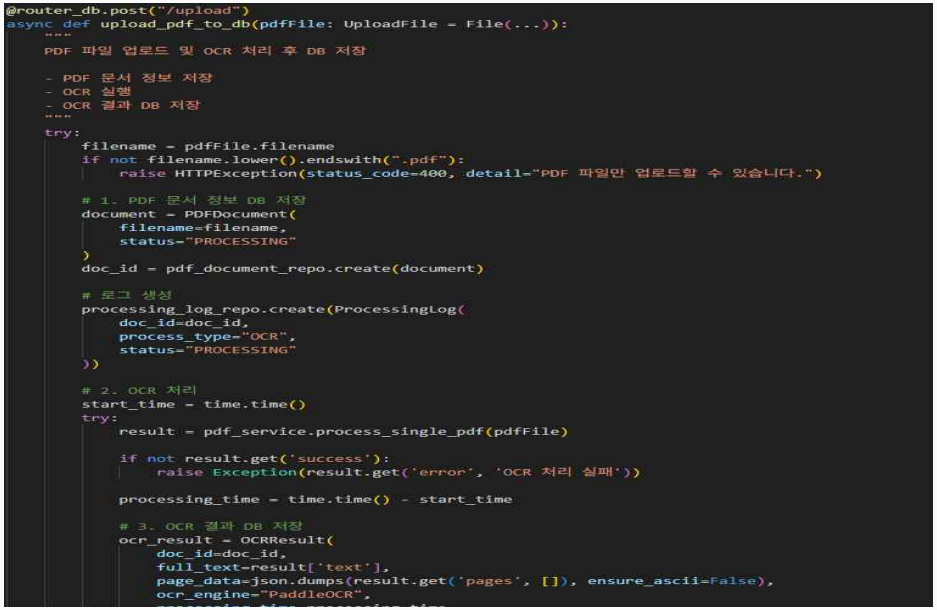
```python
app = FastAPI(title="PDF OCR API", version="1.0.0")

# CORS 설정
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],  # 필요한 경우 도메인 제한 가능
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# 라우터 등록
app.include_router(router)  # 기존 파일 기반 OCR 라우터
app.include_router(router_db)  # DB 연동 라우터

# 개발용 실행 명령
if __name__ == "__main__":
    uvicorn.run("app:app", host=HOST, port=PORT, reload=True)
```

FastAPI 애플리케이션 초기화 및 CORS 설정

- CORSMiddleware : 프론트에서 API 호출 가능하게
- include_router : 라우터를 분리해서 관리

프로젝트 수행 절차 및 방법

코드분석(백엔드) - routes_db.py DB 연동 API



```python
@router_db.post("/upload")
async def upload_pdf_to_db(pdfFile: UploadFile = File(...)):
    """
    PDF 파일 업로드 및 OCR 처리 후 DB 저장

    - PDF 문서 정보 저장
    - OCR 실행
    - OCR 결과 DB 저장
    """
    try:
        filename = pdfFile.filename
        if not filename.lower().endswith(".pdf"):
            raise HTTPException(status_code=400, detail="PDF 파일만 업로드할 수 있습니다.")

        # 1. PDF 문서 정보 DB 저장
        document = PDFDocument(
            filename=filename,
            status="PROCESSING"
        )
        doc_id = pdf_document_repo.create(document)

        # 로그 생성
        processing_log_repo.create(ProcessingLog(
            doc_id=doc_id,
            process_type="OCR",
            status="PROCESSING"
        ))

        # 2. OCR 처리
        start_time = time.time()
        try:
            result = pdf_service.process_single_pdf(pdfFile)

            if not result.get('success'):
                raise Exception(result.get('error', 'OCR 처리 실패'))

            processing_time = time.time() - start_time

            # 3. OCR 결과 DB 저장
            ocr_result = OCRResult(
                doc_id=doc_id,
                full_text=result['text'],
                page_data=json.dumps(result.get('pages', []), ensure_ascii=False),
                ocr_engine="PaddleOCR",
```

PDF 업로드 , OCR , 요약 , 키워드 추출 등 모든 DB 연동 API

- PDF 업로드 + OCR
- 문서 목록 조회
- 요약 및 키워드 생성

프로젝트 수행 절차 및 방법

코드분석(백엔드) - database.py Oracle DB 연결

```python
# 유틸리티 함수들
def execute_query(query: str, params: Optional[dict] = None) -> list:
    with db_manager.get_cursor() as cursor:
        if params:
            cursor.execute(query, params)
        else:
            cursor.execute(query)

        # Oracle은 컬럼명을 대문자로 반환하므로 소문자로 변환 (Pydantic 모델과 일치)
        columns = [col[0].lower() for col in cursor.description]
        results = []
        for row in cursor:
            # CLOB/LOB 객체를 문자열로 변환
            converted_row = []
            for value in row:
                if hasattr(value, 'read'):  # LOB 객체인 경우
                    converted_row.append(value.read())
                else:
                    converted_row.append(value)
            results.append(dict(zip(columns, converted_row)))

        return results

def execute_insert(query: str, params: dict, commit: bool = True) -> int:

    with db_manager.get_cursor(commit=commit) as cursor:
        cursor.execute(query, params)
        return cursor.rowcount


def execute_update(query: str, params: dict, commit: bool = True) -> int:

    with db_manager.get_cursor(commit=commit) as cursor:
        cursor.execute(query, params)
        return cursor.rowcount


def execute_delete(query: str, params: dict, commit: bool = True) -> int:

    with db_manager.get_cursor(commit=commit) as cursor:
        cursor.execute(query, params)
        return cursor.rowcount
```

Oracle Database 연결 풀 관리 및 쿼리 실행

- Thick Mode 초기화
- Connection Pool
- Context Manager

프로젝트 수행 절차 및 방법

코드분석(백엔드) - repository.py DB CRUD 레이어

```python
def create(self, document: PDFDocument) -> int:
    query = """
        INSERT INTO PDF_DOCUMENTS
        (FILENAME, FILE_SIZE, PAGE_COUNT, STATUS)
        VALUES (:filename, :file_size, :page_count, :status)
    """

    with db_manager.get_cursor(commit=True) as cursor:
        cursor.execute(query, {
            'filename': document.filename,
            'file_size': document.file_size,
            'page_count': document.page_count,
            'status': document.status
        })
        # 트리거가 자동으로 생성한 ID를 시퀀스에서 가져오기
        cursor.execute("SELECT SEQ_PDF_DOCUMENTS.CURRVAL FROM DUAL")
        result = cursor.fetchone()
        return result[0]

def get_by_id(self, doc_id: int) -> Optional[PDFDocument]:
    """ID로 문서 조회"""
    query = """
        SELECT DOC_ID, FILENAME, UPLOAD_DATE, FILE_SIZE, PAGE_COUNT,
               STATUS, CREATED_AT, UPDATED_AT
        FROM PDF_DOCUMENTS
        WHERE DOC_ID = :doc_id
    """
    results = execute_query(query, {'doc_id': doc_id})

    if results:
        return PDFDocument(**results[0])
    return None
```

각 테이블에 대한 CRUD 작업 담당

- 문서 생성 : Trigger를 이용한 SEQUENCE 사용
- 페이징 : ROWNUM 사용

프로젝트 수행 절차 및 방법

코드분석(백엔드) - pdf_service.py PDF처리

```python
try:
    # OCR 처리
    full_text, page_data = ocr_service.extract_text_from_pdf(temp_path)

    # 텍스트 파일 저장
    with open(txt_path, "w", encoding="utf-8") as f:
        f.write(full_text)

    # JSON 파일 저장
    json_output = {
        "filename": filename,
        "text": full_text,
        "pages": page_data
    }

    with open(json_path, "w", encoding="utf-8") as jf:
        json.dump(json_output, jf, ensure_ascii=False, indent=2)

    return {
        "filename": filename,
        "success": True,
        "filename_txt": txt_filename,
        "filename_json": json_filename,
        "text": full_text
    }
```

PDF 파일을 받아서 OCR 처리

- OCR 서비스 호출 : 실제 텍스트 추출
- 임시 파일 : FastAPI의 UploadFile 임시 저장
- 리소스 정리 : finally 블록으로 확실히 삭제

프로젝트 수행 절차 및 방법

코드분석(백엔드) - ocr_service.py OCR처리

```python
def extract_text_from_pdf(self, pdf_path: str) -> tuple[str, list[dict]]:

    full_text = ""
    page_data = []

    # PDF를 이미지로 변환
    images = convert_from_path(pdf_path, dpi=PDF_DPI)

    # 각 페이지별로 OCR 실행
    for page_num, image in enumerate(images, 1):
        ocr_text = self._extract_text_from_image(image)

        # 결과 저장
        full_text += f"[Page {page_num}]\n{ocr_text}\n\n"
        page_data.append({
            "page": page_num,
            "text": ocr_text
        })

    return full_text, page_data

def _extract_text_from_image(self, image) -> str:
    # 임시 이미지 파일 저장
    with NamedTemporaryFile(delete=False, suffix=".png") as tmp_img:
        image.save(tmp_img.name)
        image_path = tmp_img.name

    try:
        # PaddleOCR 실행
        result = self.ocr.predict(image_path)

        # 텍스트 추출
        ocr_text = ""
        if result:
            for res in result:
                # res.str['res']['rec_texts']에서 텍스트 리스트 추출
                if 'res' in res.str and 'rec_texts' in res.str['res']:
                    texts = res.str['res']['rec_texts']
                    ocr_text += "\n".join(texts) + "\n"

        return ocr_text
```

PaddleOCR을 사용한
텍스트 추출

- predict() : OCR 실행
- pdf2image : PDF -> PIL Image 변환
- DPI 300 : 고해상도 이미지

프로젝트 수행 결과

성능 지표 및 시스템 안정성

변환 정확도

90% 이상 (PaddleOCR 한국어 모델)
- 테스트 문서: 한국어 계약서 10페이지 (텍스트 약 5,000자)
- PaddleOCR 인식 결과: 4,620자 정확 인식 (92.4%)
- 수정 필요 문자: 380자 (주로 손글씨, 저화질 스캔,한자)

에러 핸들링

- 개별 격리: 한 파일의 실패가 다른 파일 처리에 영향 없음
- 상태 유지: 에러 발생해도 파일 목록에서 제거하지 않음 (재시도 가능)

CORS 설정

Frontend (http://localhost:3000)
↓ fetch 요청
Backend (http://127.0.0.1:5000)
↓ 브라우저가 차단
CORS Error: No 'Access-Control-Allow-Origin' header is present

프로젝트 수행 결과

GITHUB소스 - https://github.com/Dosum999/pdf_txt_project

자체 의견

신희섬 - 수행중 느낀점 및 경험한 성과

case1) PaddleOCR GPU 버전 호환성 문제

- 초기에는 OCR 구동을 위한 정보 및 코드들을 인터넷을 통해서 수집하였으나 계속 되는 버젼문제에 충돌하였습니다.
- OCR 모델에 따른 호환되는 cuDNN 8.9 버젼이 필요하다고 확인되었고 cuDNN 8.9에 맞는 cuda 11.8의 필요성 또한 공식 문서를 통해 확인하였고 거기에 맞는 환경을 재설정해서 해결하였습니다.

case2) TypeScript 파일 분할 및 타입/컴파일 오류

- 파일 간의 타입 정의 및 참조 문제가 빈번하게 발생하여 작업 지연되었습니다
- tsconfig.json 파일에서 module 옵션을 프로젝트 환경에 맞게 명확히 설정 후 타입 정의 파일 (.d.ts) 또는 인터페이스 파일을 별도로 관리하여 참조 관계를 단방향으로 단순화하고, 순환 참조를 제거하여 컴파일러가 타입을 명확히 인식하도록 구조 개선하였습니다.

자체 의견

김근우 - 수행중 느낀점 및 경험한 성과

case1) 맵핑하는 과정에서 하드웨어 문제

- 하드웨어 성능이 생각보다 좋지 않아 진행의 문제가 생겼습니다
  이를 해결하고자 주피터를 이용한 가상 환경 설정 및 코랩으로 진행을 시도했지만
  주기적인 연결끊김 및 일정치를 사용하면 유료로 전환되는 시스템으로 인해 프로젝트 진행 있어 문제가 생겼습니다
- 이를 해결하고자 본인 노트북을 사용함으로써 하드웨어의 문제는 일단락 해결되었습니다.

case2) TrOCR 파인튜닝의 전처리 문제

- 맵핑된 이미지 경로를 통해 이미지를 가져와 텐서로 바꿔 전처리하는 과정에서
  본인 학습 부족으로 인해 이미지를 텐서로 바꾸지 못하는 문제가 생겼습니다
  이를 해결하고자 코딩분석을 시도하며 해결하려 했지만 시간 부족으로 인해 프로젝트 진행에 문제가 생겼 추후 따로 공부해서 진행하기로 결정했습니다.

자체 의견

이인규 - 수행중 느낀점 및 경험한 성과

case1) keras-ocr 한글 인식 문제

- keras-ocr을 적용하여 pdf의 이미지 부분을 처리하는 과정에서 한글은 인식률이 심각하게 떨어지고 영어도 순서가 바뀌거나 인식을 못하는 등의 문제가 발생했습니다.
- 이러한 문제를 해결하기 위해 국산 오픈소스인 hyperClovaX-seed를 통한 ocr 적용을 고려했습니다.

case2) hyperClovaX-seed 출력 제한 문제

- hyperClovaX-seed를 통해 pdf의 이미지를 처리하면서 정상적인 출력에는 성공했지만 출력에 제한을 걸지 않았음에도 항상 400자 미만으로 출력되는 문제가 발생했습니다.
- 결국 출력에 제한이 없는 타 ocr 라이브러리의 사용을 고려하게 됐습니다.

자체 의견

장태준 - 수행중 느낀점 및 경험한 성과

case1) Tesseract의 이미지 출력 정확도 문제

- 처음 Tesseract를 사용할 때 이미지의 글자가 출력되지 않거나, 글자의 위치가 이미지와는 다른곳에 생성되는 문제가 생겨 dpi를 조절하거나 클러스터의 y값의 평균값을 기준으로 정렬하는 방법을 사용하였습니다.
- 파인 튜닝을 하지 않은 상태에서는 정확도를 높이는데 문제가 있다고 생각하여 파인 튜닝을 하기로 결정했습니다.

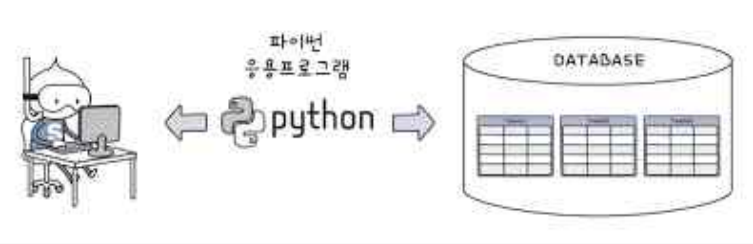
case2) Tesseract 파인 튜닝 문제

- Tesseract는 자체 학습 프레임워크를 사용하기 때문에 이미지와 box 파일이 있어야 하지만 Aihub에서 가져온 데이터에는 json 파일만 있어서 json 파일을 box 파일로 변환하는 작업이 필요했습니다.
- json 파일을 box파일로 변환은 시켰지만 학습시키기 위한 형식과는 맞지 않는지 학습이 실행되지 않았습니다. 현재는 시간 부족으로인해 더 해보지는 못 했지만 따로 배우는 시간을 가져 해결해보도록 하겠습니다

계획 및 발전 방향

한자 인식 확장

PaddleOCR 한국어 모델의 단점인 한자인식 문제를 해결하는 것이 중요합니다. 차후에 FineTunning을 통해 지속적으로 문제를 해결할 수 있습니다.



회원정보 추가

차후 프로젝트에 회원 정보를 넣어 전에 넣었던 이력을 확인할수있는 GUI 및 DB를 넣을 예정입니다.

자체 평가 의견

신희섬
9점

팀원들의 적극적인 참여에 너무 만족했던 프로젝트 였습니다. 협업속에서의 특히 PaddleOCR 한국어 인식률 92% 달성은 정말 만족스러우며 정말 뜻 깊었던 프로젝트 였습니다.

김근우
9점

팀원들의 참여도가 높은 프로젝트여서 생각보다 원활하게 진행이 되었음 뿐만이 아니라 기존모델에 추가 학습을 해보는 파인튜닝을 해보는 계기도 되어 생각보다 만족스러웠던 프로젝트였습니다.

이인규
9점

pdf to text 프로세스를 수행하면서 관련 라이브러리 사용법과 각각의 장단점 등 많은 것을 배웠습니다. 프로젝트 과정동안 모든 팀원들의 노력을 통해 가능했기 때문에 더욱 의미가 깊었습니다.

장태준
10점

팀원들 간의 꾸준한 소통으로 혼자라면 많은 시간을 소모했을 문제들을 빠른 시간내에 해결하여 다시금 협업의 중요성을 알게되었고 PDF파일을 txt 파일로 변환시키는 과정에서 모델이 사용되는 흐름을 더욱 명확하게 이해할 수 있었던 좋은 기회였습니다.

질문과 답변

프로젝트에 대해 궁금하신 점이 있으신가요?

자유롭게 질문해 주세요.

감사합니다.

팀장: 신희섬
팀원: 김근우 , 이인규 , 장태준