SK네트웍스 Family AI 과정 10기

데이터 수집 및 저장 데이터베이스 설계문서

산출물 단계	데이터 수집 및 저장
평가 산출물	데이터베이스 설계문서
제출 일자	25.05.30
깃허브 경로	https://github.com/SKNETWORKS-FAMILY-AICAMP/SKN10-FINAL-3Team
작성 팀원	문승기, 신민주

1. 채택한 모델링 방법론: IE/Crow's Foot 모델

1.1 이유

- IE/Crow's Foot 모델은 엔터티 간의 관계를 시각적으로 명확하게 표현할 수 있는 ERD 표기법으로, 1:1, 1:N, M:N과 같은 관계를 직관적으로 나타낼 수 있어 모델링 구조를 이해하기 쉽습니다. 또한, 본 프로젝트에서는 사용자, 사건, 문서 등 다양한 엔터티가 서로 관계를 맺고 있으며, 특히 사용자-전문분야 등 다대다 관계를 포함한 복합적인 구조를 갖기 때문에 관계 표현이 용이하고 해석이 쉬운 IE/Crow's Foot 모델이 적합하다고 판단하였습니다.

2. 논리 데이터 모델

2.1 엔터티 목록

엔터티명	설명	식별자 (PK)	속성명	데이터 타입	Not Null	설명
사용자	사용자 정보 테이블	사용자 번호	이름	VARCHAR(20)	✅	사용자 이름
			이메일	VARCHAR(100)	✅	사용자 이메일 주소

			비밀번호	VARCHAR(128)	✅	비밀번호
			자기소개	TEXT		자기소개
			소속경력	TEXT		경력 - 소속(회사)
			수행경력	TEXT		경력 - 수행(업무, 자문)
			고등 학력	VARCHAR(100)		고등 학력
			대학 학력	VARCHAR(100)		대학 학력
			대학원 학력	VARCHAR(100)		대학원 학력
			소속 부서	VARCHAR(30)	✅	소속 부서(CODE 테이블 참고)
			역할	VARCHAR(30)	✅	역할(CODE 테이블 참고)
			전문 분야	VARCHAR(30)	✅	전문 분야(CODE 테이블 참고)
			생성일	DATETIME	✅	가입일
			수정일	DATETIME	✅	수정일

엔터티명	설명	식별자 (PK)	속성명	데이터 타입	Not Null	설명
사건	사건 테이블	사건 번호	사건명	VARCHAR(100)	✅	사건명
			본문	TEXT	✅	사건 설명
			클라이언트	VARCHAR(20)	✅	의뢰인명
			사건 유형(대)	VARCHAR(20)	✅	민사, 형사 등(CODE 테이블 참고)
			사건	VARCHAR(50)	✅	손해배상, 폭행

			유형(중)			등
			사건 유형(소)	VARCHAR(50)	✅	교통사고, 의료사고 등
			메모	TEXT		메모/특이사항
			담당 부서	VARCHAR(30)		사건을 담당할 부서 - AI(CODE 테이블 참고)
			상태값	VARCHAR(30)		소송 1심,2심,3심 (CODE 테이블 참고)
			상태값	VARCHAR(30)	✅	사건 의뢰 및 진행 상태 (CODE 테이블 참고)
			상태값	VARCHAR(30)		사건 종결 세분화(CODE 테이블 참고)
			소송 재기일	DATETIME	✅	소장을 제출해서 시작된 날짜
			생성일	DATETIME	✅	사건 진행 상태
			수정일	DATETIME	✅	사건 등록일

엔터티명	설명	식별자 (PK)	속성명	데이터 타입	Not Null	설명
업무	업무 테이블	업무 번호	사용자 번호	INT	✅	사용자 ID
			사건 번호	INT	✅	사건 ID
			업무명	VARCHAR(100)	✅	업무명
			본문	TEXT	✅	업무 설명

			업무 지시자	INT	✅	지시자
			업무 대상자	INT	✅	대상자
			상태값	VARCHAR(30)	✅	업무 상태(CODE 테이블 참고)
			생성일	DATETIME	✅	생성일
			수정일	DATETIME	✅	수정일

엔터티명	설명	식별자 (PK)	속성명	데이터 타입	Not Null	설명
판례	판례 테이블	사건 번호	법원명	VARCHAR(60)	✅	법원이름
			판례 키워드	VARCHAR(100)	✅	판례 키워드
			판례 결과	VARCHAR(15)	✅	판례 결과
			선고일자	DATETIME	✅	선고일자

엔터티명	설명	식별자 (PK)	속성명	데이터 타입	Not Null	설명
코드	코드 테이블	코드	코드 이름	VARCHAR(50)	✅	
			코드 설명	VARCHAR(50)		
			상위 코드	VARCHAR(20)		
			설명	TEXT		

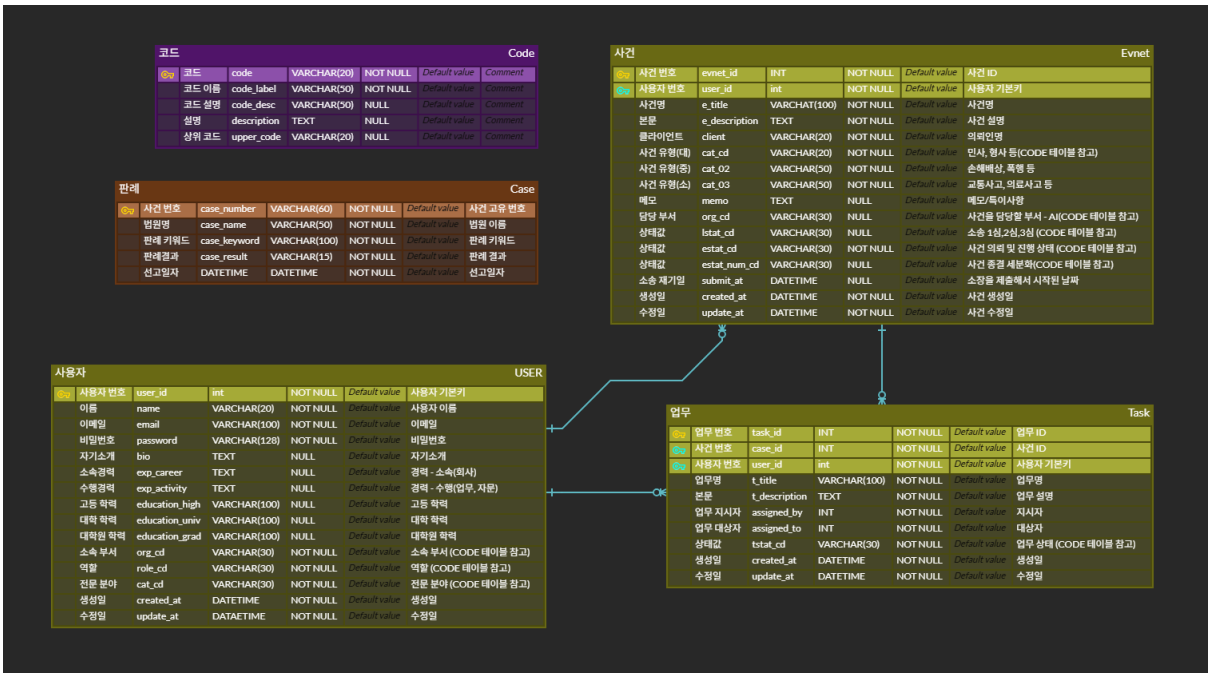
2.2 엔터티 간 관계

관계명	주 엔터티	종 엔터티	관계 종류	설명

사용자 - 사건	사용자	사건	1:N	한 명의 사용자는 여러 사건을 생성할 수 있으나, 하나의 사건은 1명의 사용자에게만 소속된다.
사용자 - 업무	사용자	업무	1:N	한 명의 사용자는 여러 업무를 지시하거나 또는 수행할 수 있음. assigned_by, assigned_to 양쪽 모두 사용자 ID를 참조한다.
사건 - 업무	사건	업무	1:N	하나의 사건은 여러 업무(Task)로 구성되며, 업무는 반드시 하나의 사건에 소속된다.

2.3 ERD (이미지로 삽입)

- 캡처 화면으로 잘 보이지 않을 것 같아서 추가적인 첨부파일로 제출했습니다. (DB_ERD.png)



3. 물리 데이터 모델

3.1 테이블 정의서

- 논리 데이터 모델은 주로 비즈니스 관점에서 핵심 정보를 정의하는 데 중점을 두고, 물리 데이터 모델은 이를 실제 구현 가능한 데이터베이스 구조로 구체화하는 단계이기 때문에, 동일한 엔터티 구조를 기반으로 하되 물리 모델에서는 제약 조건, 데이터 타입, 인덱스, 기본값 등 시스템 수준의 세부 사항을 추가하여 설계의 완성도를 높이는 방식으로 구성하였습니다.

- 코드 테이블 정의서

    - 코드 테이블 정의서는 시스템 전반에서 반복적으로 사용되는 분류 값(예: 부서, 역할, 사건 상태, 업무 상태 등)을 중앙에서 일관되게 관리하고 유지하기 위한 목적으로 작성하였습니다.

    - 이 테이블은 각종 엔터티의 코드 속성값들을 정규화하여 중복을 방지하고, 향후 새로운 코드 값 추가나 수정 시에도 데이터베이스 구조를 변경하지 않고도 유연하게 대응할 수 있도록 설계되었습니다. 또한 upper_code를 활용한 계층형 구조 정의를 통해, 상위-하위 코드 간 의미 있는 관계를 표현함으로써 시스템의 확장성과 유지보수성을 향상시키는 기반을 다졌습니다.

테이블명	컬럼명	데이터 타입	PK	FK	Not Null	제약조건 설명
Code	code	VARCHAR(20)	✅		✅	
	code_label	VARCHAR(50)			✅	
	code_desc	VARCHAR(50)				
	description	TEXT				
	upper_co	VARCHAR(50)		✅	✅	

	de					

- 논리 데이터 모델에서 정의했던 테이블 정의서

테이블명	컬럼명	데이터 타입	PK	FK	Not Null	제약조건 설명
User	user_id	INT	✅		✅	Auto Increment
	name	VARCHAR(20)			✅	
	email	VARCHAR(100)			✅	Unique
	password	VARCHAR(128)			✅	
	bio	TEXT				
	exp_career	TEXT				
	exp_activity	TEXT				
	education_high	VARCHAR(100)				
	education_univ	VARCHAR(100)				
	education_grad	VARCHAR(100)				
	created_at	DATETIME			✅	Default(CURRENT_TIMESTAMP)
	update_at	DATETIME			✅	Default(CURRENT_TIMESTAMP)

테이블명	컬럼명	데이터 타입	PK	FK	Not Null	제약조건 설명
Event	event_id	INT	✅		✅	Auto Increment
	user_id	INT		✅	✅	FK to User.user_id
	e_title	VARCHAR(100)			✅	
	e_description	TEXT			✅	
	client	VARCHAR(20)			✅	
	cat_cd	VARCHAR(20)			✅	
	cat_02	VARCHAR(50)			✅	
	cat_03	VARCHAR(50)			✅	
	memo	TEXT				
	org_cd	VARCHAR(30)				
	lstat_cd	VARCHAR(30)				
	estat_cd	VARCHAR(30)			✅	
	estat_num_cd	VARCHAR(30)				
	submit_cd	DATETIME			✅	Default(CURRENT_TIMESTAMP)
	created_at	DATETIME			✅	Default(CURRENT_TIMESTAMP)
	update_at	DATETIME			✅	Default(CURRENT_TIMESTAMP)

테이블명	컬럼명	데이터 타입	PK	FK	Not Null	제약조건 설명
Task	task_id	INT	✅		✅	Auto Increment
	event_id	INT		✅	✅	FK to Event.event_id
	user_id	INT		✅	✅	FK to User.user_id
	t_title	VARCHAR(100)				
	t_description	TEXT				
	assigned_by	INT		✅	✅	FK to User.user_id
	assigned_to	INT		✅	✅	FK to User.user_id
	tstat_cd	VARCHAR(30)			✅	
	created_at	DATETIME			✅	Default(CURRENT_TIMESTAMP)
	update_at	DATETIME			✅	Default(CURRENT_TIMESTAMP)

테이블명	컬럼명	데이터 타입	PK	FK	Not Null	제약조건 설명
Case	case_number	VARCHAR(60)	✅		✅	
	case_name	VARCHAR(50)			✅	
	case_keyword	VARCHAR(100)			✅	

	case_result	VARCHAR(15)			✅	
	case_at	DATETIME			✅	Default(CURRENT_TIMESTAMP)

3.2 제약 조건 명세

- Primary Key: 각 테이블에 1개 이상 존재하며, NULL 및 중복을 허용하지 않음
- Foreign Key: 참조 테이블 명시 및 on delete / on update 정책 설정
- Unique: 중복을 허용하지 않아야 하는 컬럼에 설정
- Default: 컬럼에 값이 없을 경우 기본값 자동 설정
- Auto Increment: 숫자형 컬럼에 대해 자동 증가 설정 (주로 기본키에 사용)

4. 데이터 정합성 및 무결성 관리 방안

- 중복 방지 전략
    - 데이터베이스의 정합성과 중복 방지를 위해 주요 테이블에 대해 UNIQUE 제약조건과 정규화된 코드 관리 체계를 도입하였습니다.
    - 우선, 사용자 정보 테이블의 `email` 컬럼에는 UNIQUE 제약을 설정하여 동일 이메일 주소의 중복 등록을 방지하고, 사용자 식별의 정확성과 인증 과정의 신뢰성을 높였습니다.
    - 조직 정보, 역할, 사건 상태, 업무 상태 등과 같이 반복적으로 사용되는 분류 값은 모두 코드 테이블을 통해 일관성 있게 관리됩니다. 코드 테이블은 `code`, `code_label`, `upper_code` 등을 포함한 계층 구조로 설계되었으며, 예를 들어 `ORG_01`, `ORG_01_01` 등과 같이 상하위 부서 구분이 가능하고, `ROLE_01`, `ROLE_02` 등 역할 구분도 명확히 할 수 있도록 구성되어 있습니다.

- 이러한 코드 값들은 사용자(User), 사건(Event), 업무(Task) 등의 테이블에서 외래키 형태로 참조되며, 시스템 전반에서 동일한 코드가 재사용되도록 설계함으로써 불필요한 중복 입력을 방지합니다. 예컨대 사용자 테이블의 소속 부서, 역할, 전문 분야 컬럼은 모두 코드 테이블의 값을 참조하게 되어 있으며, 코드 테이블 내에서도 code_label의 중복을 방지하기 위해 UNIQUE 제약 조건이 적용됩니다.

- 또한 사건 상태 코드, 업무 상태 코드, 소송 단계 코드 등은 명확한 정의와 계층 구조를 갖춘 상태 값으로 관리되며, 이로 인해 데이터 입력 시 잘못된 상태값의 사용을 방지하고, 유지보수 시에도 일관된 기준 하에 관리될 수 있도록 지원합니다.

- 결론적으로, 이메일 주소의 고유성 확보뿐만 아니라 코드 테이블 기반의 정규화된 구조 설계를 통해 전체 시스템 내 데이터의 중복을 최소화하고 무결성을 보장할 수 있도록 구성하였습니다.

- 정규화 수준: 1NF / 2NF / 3NF 여부 명시
  - **제1정규형(1NF)**:
    본 데이터베이스 설계는 모든 테이블에서 컬럼이 원자값만을 갖도록 구성되어 있으며, 반복 그룹이나 다중 값 컬럼 없이 데이터를 저장합니다. 예를 들어, 사용자 학력 정보는 education_high, education_univ, education_grad와 같이 각각의 학력 단계를 개별 컬럼으로 분리하여 구성함으로써 하나의 필드에 여러 값을 넣지 않도록 설계하였습니다. 또한 경력 정보(소속경력, 수행경력)나 사용자 자기소개 등의 항목도 TEXT 형태로 하나의 텍스트 필드에 명시적으로 기록되도록 하여 1NF의 조건을 만족합니다.

  - **제2정규형(2NF)**:
    모든 테이블은 단일 기본키를 중심으로 구성되어 있으며, 복합 기본키를 사용하는 관계 테이블은 존재하지 않습니다. 따라서 각 테이블의 비기본키

속성들은 기본키 전체에 완전 종속되어 있어, 부분 종속성이 존재하지 않습니다. 예를 들어, 사건(Event) 테이블에서는 `event_id`가 기본키이고, `client`, `cat_cd`, `cat_02`, `cat_03` 등 모든 컬럼은 해당 사건 번호에 완전히 종속됩니다. 업무(Task) 테이블에서도 `task_id`를 기본키로 하여 `assigned_by`, `assigned_to`, `tstat_cd` 등의 속성이 독립적으로 종속됩니다. 따라서 전체적으로 2NF를 충족합니다.

- **제3정규형(3NF)**:
  각 테이블의 비기본키 속성들은 오직 기본키에만 직접 종속되도록 설계되어 있으며, 이행적 종속은 존재하지 않도록 구성하였습니다. 예를 들어 사용자 테이블에서 소속 부서, 역할, 전문 분야와 같은 속성들은 Code 테이블의 코드값을 참조함으로써 별도 관리되며, 해당 속성값의 설명이나 이름은 코드 테이블을 통해 참조됩니다. 이로 인해 데이터 중복이나 업데이트 이상을 방지하고, 변경에 유연하게 대응할 수 있습니다. 이처럼 다수의 반복 사용되는 상태값또한 모두 코드 테이블로 정규화되어 있어, 제3정규형의 요구 조건을 충족합니다.

- 어플리케이션/백엔드 레벨 검증 여부
  - 데이터의 무결성과 안정성을 확보하기 위해, 본 시스템에서는 다양한 검증 로직을 어플리케이션 및 백엔드 레벨에서 이중으로 적용할 예정입니다.
  - 우선, 사용자 등록 시 이메일 형식은 정규표현식을 통해 서버 단에서 검증하며, 비밀번호는 최소 길이, 문자 조합 요건(숫자, 특수문자 포함 등)을 만족하도록 검증할 예정입니다.
    비밀번호는 Django 프레임워크에서 기본 제공하는 PBKDF2 + SHA256 기반의 암호화 방식을 통해 해시 처리하여 저장할 계획이며, 사용자의 개인정보 보호와 보안성을 동시에 확보할 수 있도록 할 예정입니다.
  - 문서 업로드와 관련해서는 `.docx`, `.pdf` 등 허용된 확장자 이외의 파일은 백엔드에서 MIME 타입을 기준으로 차단할 예정이며, 프론트엔드에서도 사전 필터링을 적용하여 잘못된 파일 형식이 업로드되지 않도록 이중 검증

구조를 도입할 계획입니다.

- 또한, 사건 상태값, 소송 단계, 업무 처리 상태 등은 모두 Code 테이블 기반의 코드값으로 관리되며, ENUM 대신 유효한 코드값만 입력되도록 제한할 예정입니다. 프론트엔드에서는 선택 가능한 옵션만을 노출하고, 백엔드에서는 해당 코드가 Code 테이블에 존재하는지 확인하는 방식으로 입력값의 정합성을 검증할 예정입니다.

- 예외 데이터 처리 전략
  - `bio`, `memo`, `exp_activity` 등 선택 입력 항목은 NULL 허용을 통해 유연하게 처리할 예정이며, 반면 필수 항목 누락이나 외래키 무결성 위반, UNIQUE 제약 위반 등은 서버에서 즉시 감지하여 사용자에게 명확한 오류 메시지를 반환할 예정입니다.
  - 또한 업무 배정, 결재 등 사용자 권한에 따른 동작에 대해서도 예외 처리를 구체화하여, 사용자가 권한 밖의 행위를 시도할 경우 명확한 피드백을 제공하고, 시스템 안정성을 유지할 수 있도록 설계할 계획입니다.

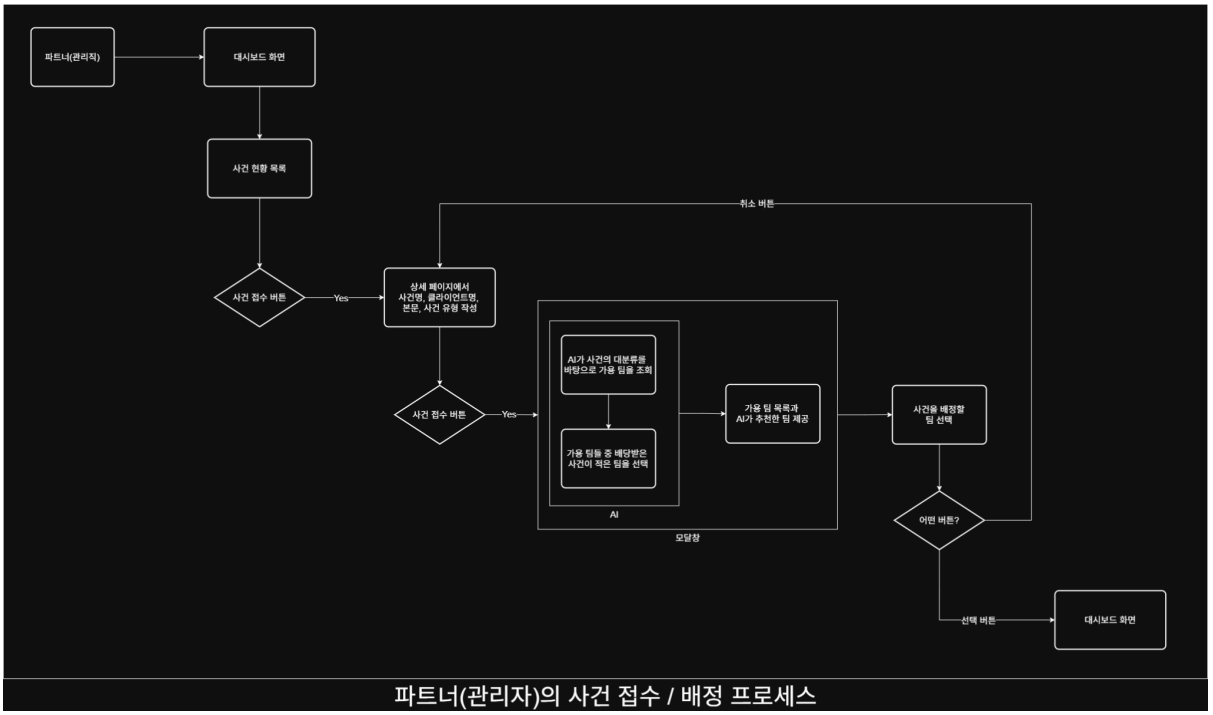
5. **변경 이력 및 적용 전략**

6. **추가 (선택)**

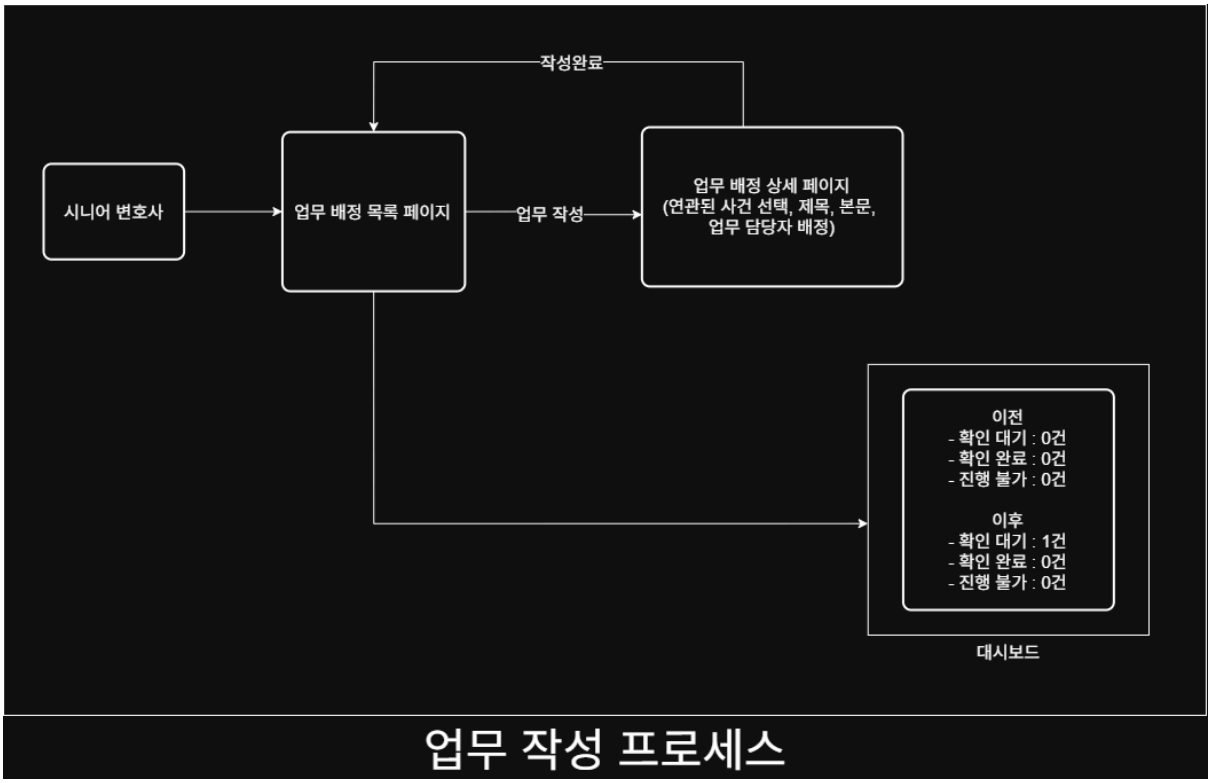
- 주요 테이블 샘플 데이터
  - 따로 첨부한 더미 데이터.xlsx를 통해 코드 테이블의 내용과 사용자 테이블의 내용을 확인하실 수 있습니다.

- 데이터 흐름도 (DFD) 또는 시퀀스 다이어그램 (캡처 화면으로 잘 보이지 않을 것 같아서 추가적인 첨부파일로 제출했습니다.)

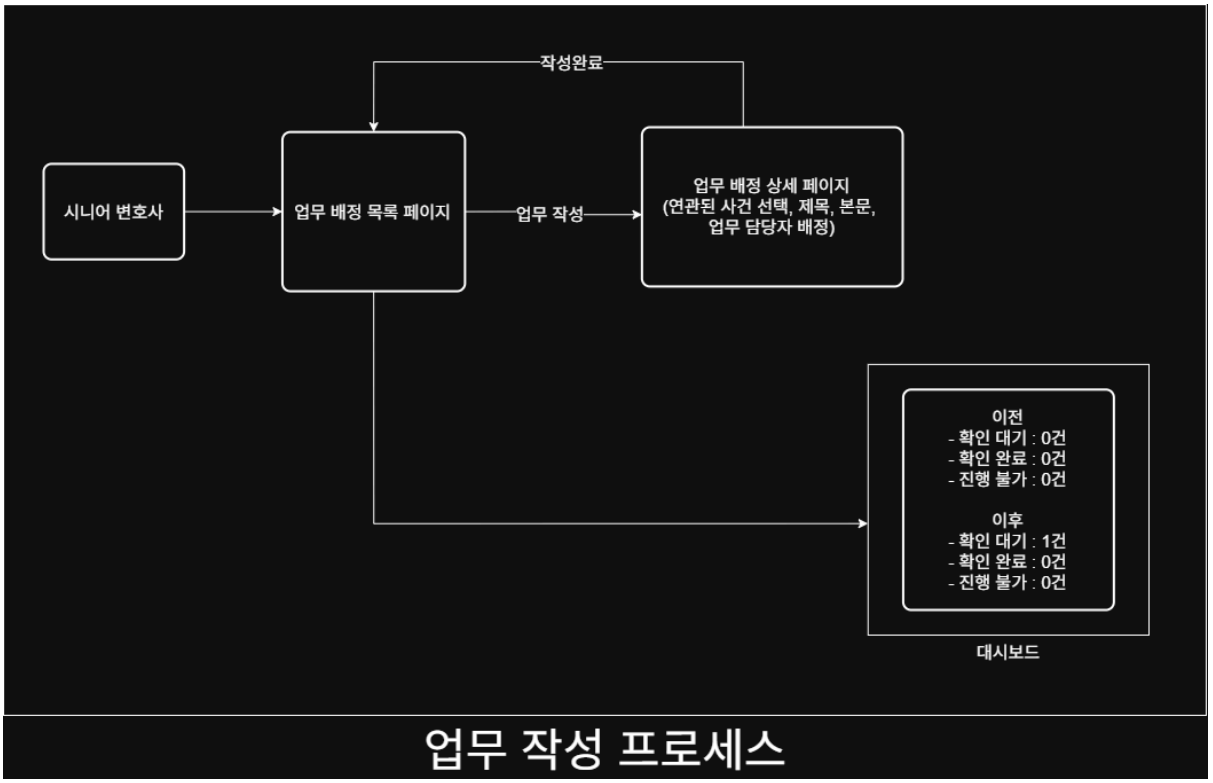
- DFD_파트너의 사건 접수/배정 프로세스.png



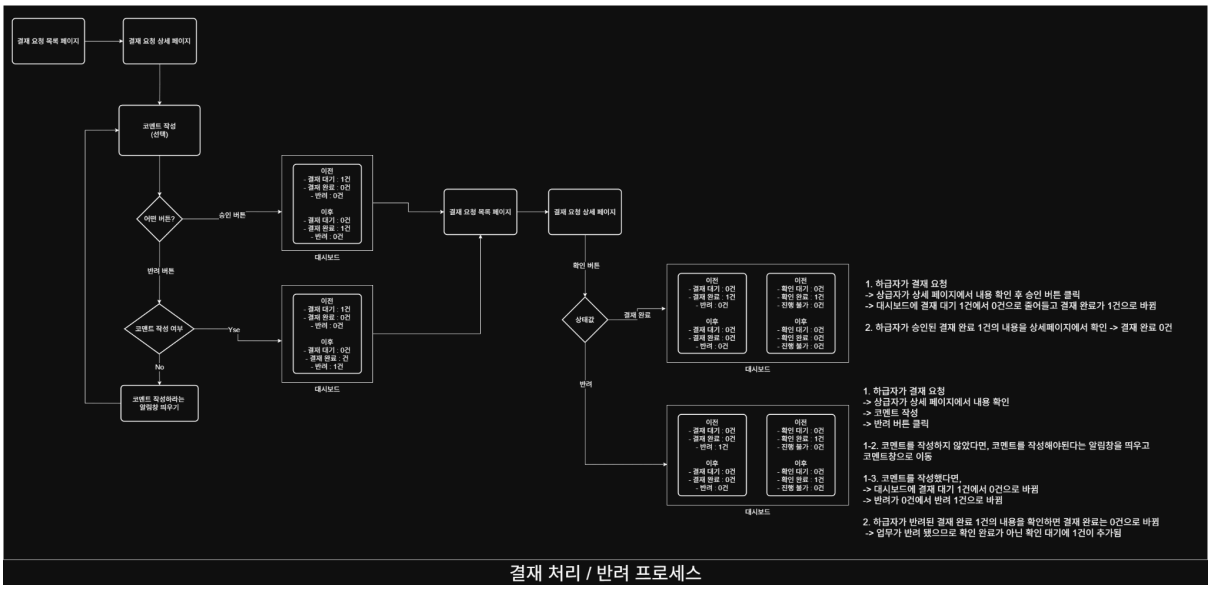
- DFD 업무 작성 프로세스.png



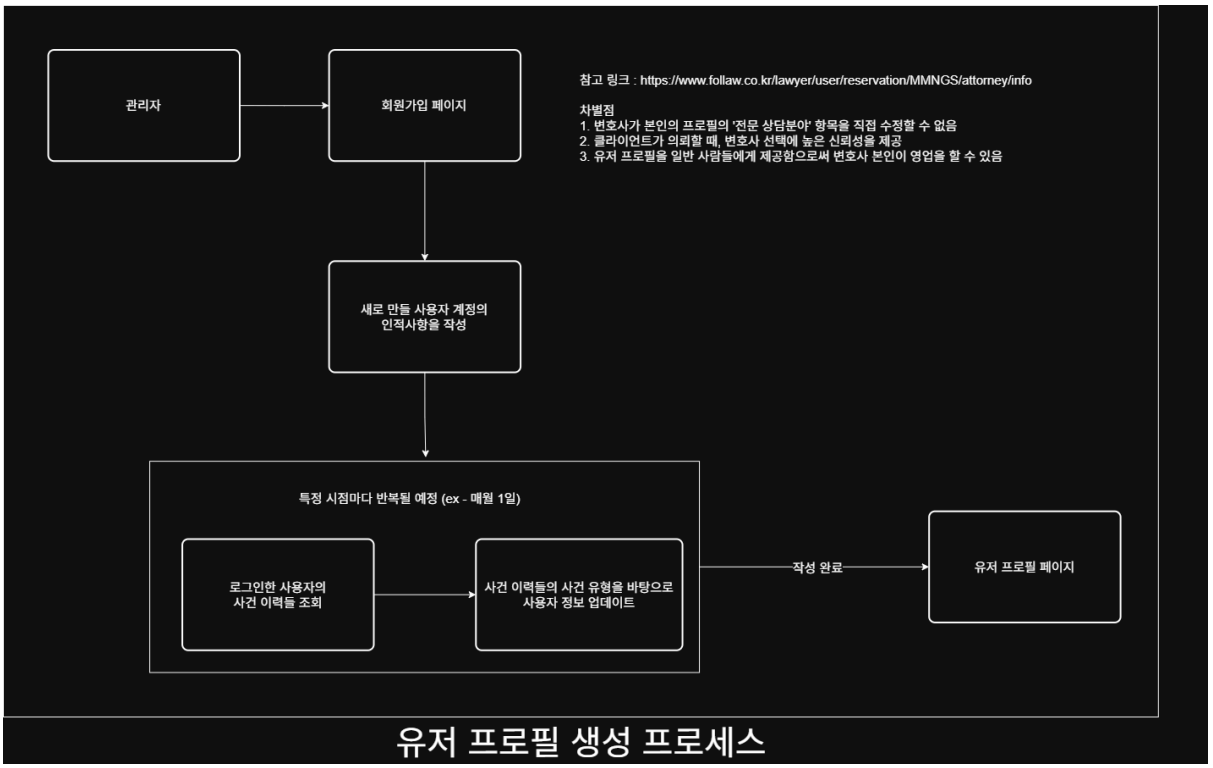
- DFD 업무 승인/거절 프로세스.png



- DFD 결재 처리/반려 프로세스.png



- DFD 유저 프로필 생성 프로세스.png



- 개인정보 보호 항목 및 암호화 필드 정의
  - Django 프레임워크를 활용하여 백엔드 개발을 진행할 예정이기 때문에 프레임워크에서 지원하는 비밀번호 암호화 (PBKDF2 + SHA256 알고리즘) 방법을 활용하여 사용자의 비밀번호를 암호화하여 DB에 저장할 것입니다.
  - 이 경우, Django 프레임워크에서 기본적으로 제공하는 User 모델이 아니라 커스텀한 모델을 사용하더라도 동일하게 암호화가 가능합니다.