현대적 보건관리시스템을 위한 아키텍처 청사진: UI/UX 재설계 및 프론트엔드 리팩토링 가이드

제 1부: 기초 전략 - 보건관리시스템의 해체 및 재구성

이 파트에서는 프로젝트의 전략적 기반을 구축합니다. 시스템이 누구를 위해 무엇을 해야 하는지 정의하고, 데이터베이스 테이블이 아닌 사용자 목표를 중심으로 정보 구조를 설계합니다. 이 기초 작업은 시각적으로는 현대적이지만 기능적으로는 결함이 있는 애플리케이션을 만드는 것을 방지하는 데 매우 중요합니다.

제 1장: 핵심 기능 도메인 및 사용자 페르소나 정의

본 장에서는 현대 산업안전보건(Occupational Health & Safety, OHS) 시스템의 복잡한 요구사항을 분석하고 해체합니다. 단순한 기능 목록을 넘어, 상호 연결된 기능적 도메인과 이를 사용하는 사용자 페르소나를 설정하여 시스템의 근본적인 목적과 사용자를 명확히 합니다.

1.1. 현대 OHS 플랫폼의 기능적 지형 종합

현대적 시스템에 요구되는 포괄적인 기능들을 시장 조사를 통해 분석하고, 이를 논리적 도메인으로 분류합니다.

- 핵심 도메인 식별:
    - 건강 감시 및 모니터링: 채용 전 건강 검진, 정기 건강 검진, 예방 접종 추적, 생체 정보 모니터링.
    - 사고 및 사례 관리: 부상/아차사고 보고 및 조사, 근본 원인 분석, 업무 복귀 계획.
    - 위험 및 유해 요인 관리: 선제적 위험성 평가(HIRA), 유해 요인 식별, 완화 계획 및 안전 관찰.
    - 규정 준수 및 규제 관리: OSHA/ISO 45001 기록 관리, 감사 관리, 허가 추적, 자동화된 보고.
    - 교육 및 참여: 교육 관리, 자격증 추적, 안전 회의 문서화, 직원 웰니스 프로그램.

1.2. 주요 사용자 페르소나 및 핵심 목표 정의

기능적 도메인을 기반으로 주요 사용자 역할을 정의합니다. 명확한 사용자 정의가 없는 시스템은 누구의 요구도 효과적으로 충족시키지 못하는 경우가 많습니다.

- 페르소나 1: 보건안전(H&S) 관리자: 파워 유저. 동향 모니터링, 진행 중인 사례 관리, 규정 준수 확인, 시정 조치 할당 등을 위한 분석적/운영적 관점이 필요합니다.
- 페르소나 2: 임원/관리자: 전략적 사용자. 재해율(LTI), 규정 준수 점수, 위험 노출과 같은 핵심 성과 지표(KPI)에 대한 고차원적인 대시보드 뷰를 통해 정보에 입각한 비즈니스 결정을 내릴 필요가 있습니다.
- 페르소나 3: 일반 직원: 일선 사용자. 유해/사고 보고, 건강 검진 일정 확인, 안전 교육 이수와 같은 작업을 위한 간단하고 접근하기 쉬운 포털이 필요합니다.
- 페르소나 4: 산업 보건 전문가(의료인): 임상 데이터, 건강 검진(예: DOT 신체검사), 직무

적합성 평가 등을 관리하는 전문 사용자입니다.

1.3. 기능적 사일로에서 통합 워크플로우로의 전환

이러한 기능적 도메인들은 독립적인 사일로(silo)가 아니라, PDCA(Plan-Do-Check-Act) 사이클과 같은 더 큰 프로세스 프레임워크의 일부로서 깊이 상호 연결되어 있습니다. 예를 들어, 하나의 '사고'(Do)는 '조사'와 '위험성 평가'(Check)를 촉발하고, 이는 새로운 '예방 조치'와 '교육'(Act/Plan)으로 이어집니다. 시스템이 이러한 실제 워크플로우를 반영하지 않고 단순히 데이터 유형(예: '사고' 메뉴, '위험' 메뉴)에 따라 분리되어 설계된다면, 사용자는 각 단계를 수동으로 연결해야 합니다. 이는 인지 부하를 증가시키고 오류 가능성을 높입니다. 따라서 시스템 아키텍처는 데이터베이스 엔터티 중심이 아닌, '사고 사례 관리'와 같은 실제 사용자 과업 중심으로 설계되어야 합니다. 이러한 접근 방식은 설계 패러다임을 데이터 중심에서 사용자 중심으로 근본적으로 전환시키며, 다음 장에서 제안할 정보 구조의 핵심 동인이 됩니다.

표 **1:** 핵심 기능 모듈 및 관련 사용자 페르소나

기능 도메인	하위 기능	주요 사용자 페르소나	핵심 목표
건강 감시 및 모니터링	예방 접종 추적, 정기 검진	보건안전 관리자, 의료인	작업 인력이 예방 접종 요건을 준수하도록 보장
사고 및 사례 관리	사고 보고, 근본 원인 분석	보건안전 관리자, 일반 직원	사고를 신속하게 보고하고 재발 방지 대책 수립
위험 및 유해 요인 관리	위험성 평가, 안전 관찰	보건안전 관리자, 일반 직원	잠재적 위험을 사전에 식별하고 선제적으로 제거
규정 준수 및 규제 관리	감사 관리, 규제 보고서 생성	보건안전 관리자, 임원/관리자	법적 요구사항을 준수하고 규제 기관 감사를 대비
교육 및 참여	교육 이력 관리, 안전 회의	보건안전 관리자, 일반 직원	모든 직원이 필수 안전 교육을 이수하도록 보장

제 **2**장**:** 과업 중심 엔터프라이즈 애플리케이션을 위한 정보 구조

본 장에서는 애플리케이션 구조의 청사진을 제시합니다. 시스템의 복잡성을 길들이기 위해 확립된 정보 구조(Information Architecture, IA) 원칙을 적용하며, 사용자를 위한 직관적이고 과업 중심적인 경로를 만드는 데 중점을 둡니다.

2.1. 복잡한 시스템에서 전통적 **IA**의 실패

단순한 계층적 또는 조직도 기반의 내비게이션이 데이터 집약적인 애플리케이션에서 실패하는 이유를 논의합니다. 이러한 접근 방식은 깊고 혼란스러운 메뉴 구조와 높은 인지 부하를 유발합니다.

2.2. 현대적 **IA**를 위한 핵심 원칙

새로운 IA가 논리적이고 확장 가능하며 사용자 친화적이도록 보장하기 위해 다음과 같은 지침 원칙을 채택합니다.

- 선택의 원칙 **(Principle of Choices):** 의사결정 피로를 방지하기 위해 옵션을 제한합니다.

포괄적인 목록 대신 의미 있는 몇 가지 선택지를 제시합니다.

- 공개의 원칙 **(Principle of Disclosure):** 현재 맥락에 필요한 정보만 보여줍니다. 사용자가 더 깊이 탐색함에 따라 점진적으로 세부 정보를 공개("드릴다운")합니다. 이는 대시보드와 복잡한 양식에 핵심적입니다.
- 다중 분류의 원칙 **(Principle of Multiple Classification):** 사용자가 여러 논리적 경로를 통해 정보를 찾을 수 있도록 합니다 (예: 직원 파일, 프로젝트 안전 기록, 최근 사고 목록을 통해 동일한 사고 보고서에 접근)..
- 집중된 내비게이션의 원칙 **(Principle of Focused Navigation):** 내비게이션 메뉴를 사용자의 현재 과업에 집중시키고 관련 없는 링크를 피해 일관성을 유지합니다.

2.3. 제안 **IA** 모델**:** 과업 및 역할 기반 하이브리드 구조

제안하는 모델은 단일한 사이트맵이 아니라, 특정 페르소나에 맞춰진 사용자 여정 기반 흐름의 집합이 될 것입니다.

- 최상위 내비게이션**:** 주 내비게이션은 단순한 데이터 유형이 아닌, 상위 수준의 과업과 책임 영역별로 구성됩니다.
    - 대시보드 (개인화된 랜딩 페이지)
    - 작업자 관리 (직원 건강 기록, 검진, 교육 관리)
    - 운영 관리 (사업장/프로젝트의 사고, 감사, 위험성 평가 관리)
    - 규정 준수 (규제 보고서, 문서 관리, 표준)
    - 분석 및 보고서 (데이터 탐색 및 시각화)
    - 설정 (시스템 관리)
- 사용자 여정 매핑**:** "신규 부상 보고서 대응"과 같은 핵심 사용자 여정을 개괄적으로 설명하여, 사용자가 '모듈' 간을 의식적으로 전환할 필요 없이 IA의 여러 부분을 원활하게 이동하는 방식을 보여줍니다.

2.4. 인지 부하 감소의 핵심 동인으로서의 정보 구조

잘 설계된 정보 구조는 복잡한 애플리케이션에서 인지 부하를 최소화하는 가장 효과적인 단일 도구입니다. 시각 디자인은 복잡한 인터페이스를 더 깨끗하게 보이게 할 수 있지만, 오직 강력한 IA만이 근본적으로 사용하기 쉽게 만들 수 있습니다. 전통적인 IA는 종종 사용자에게 거대한 내비게이션 트리를 제시하여 필요한 항목을 찾기 위해 수십 개의 항목을 스캔하도록 강요함으로써 '공개'나 '선택'과 같은 원칙을 위반합니다. 이 스캔 및 필터링 행위 자체가 인지적 비용입니다. 제안된 과업 기반 IA는 특정 목표를 중심으로 행동과 정보를 그룹화하여 본질적으로 선택을 제한합니다. 이로 인해 효과적인 IA는 인지 부하 감소로 이어지고, 이는 더 빠른 과업 완료, 더 높은 사용자 만족도, 증가된 시스템 채택률, 그리고 궁극적으로는 더 안전한 작업 환경과 같은 더 나은 비즈니스 결과로 이어지는 인과 관계를 형성합니다. 이는 추상적인 정보 구조 개념이 조직의 핵심 비즈니스 목표에 직접적이고 측정 가능한 영향을 미친다는 것을 보여주며, IA 검증을 위한 사용자 리서치, 카드 소팅, 트리 테스팅에 대한 초기 투자를 정당화합니다.

제 **2**부**:** 디자인 시스템 **-** 시각적 및 상호작용적 통일성을 위한 청사진

이 파트에서는 전략적 기반을 구체적인 디자인 프레임워크로 전환합니다. 시스템의 외관, 느낌, 상호작용 패턴을 정의하여 전체 리팩토링 노력을 이끌 확장 가능하고 일관된 언어를

구축합니다.

제 **3**장**:** 현대적 시각 및 상호작용 언어 구축

여기서는 애플리케이션을 현대적이고 직관적으로 느끼게 할 미학적 및 사용자 경험 원칙을 정의하고, 개발을 가속화할 기초 프레임워크를 선택합니다.

3.1. "현대적**"** 미학의 정의**:** 명확성**,** 효율성**,** 접근성

목표는 일시적인 트렌드를 쫓는 것이 아니라, 엔터프라이즈 사용자에게 실질적인 가치를 제공하는 원칙을 채택하는 것입니다.

- 명확성 강조**:** 충분한 여백, 명확한 시각적 계층 구조, 가독성 높은 타이포그래피를 사용하여 인지 부하를 줄입니다.
- 카드 기반 **UI (**벤토 그리드**):** 대시보드 및 복잡한 디스플레이에 카드 기반 레이아웃을 채택합니다. 이 접근 방식은 모듈식이며, 훑어보기 쉽고, 본질적으로 반응형입니다.
- 의미 있는 마이크로인터랙션**:** 미묘한 애니메이션과 피드백을 사용하여 행동을 확인하고, 주의를 유도하며, 방해되지 않으면서 인터페이스가 반응적으로 느껴지게 합니다.
- 핵심 원칙으로서의 접근성**:** 처음부터 높은 명암비(특히 다크 모드), 명확한 포커스 상태, 스크린 리더 호환성을 고려하여 설계합니다.

3.2. 권장 사항**:** 성숙한 디자인 시스템 채택

일관성을 보장하고 개발을 가속화하기 위해, 기존의 엔터프라이즈급 디자인 시스템을 기반으로 구축할 것을 강력히 권장합니다.

- 후보 분석**:**
    - **Google Material Design (M2/M3):** 범용 애플리케이션에 탁월하며, 강력한 모바일 지원과 포괄적인 가이드라인을 제공합니다.
    - **Ant Design:** 데이터 집약적인 엔터프라이즈 애플리케이션에 특화되어 있습니다. 데이터 입력, 데이터 표시(테이블, 차트), 복잡한 양식을 위한 매우 풍부한 컴포넌트 세트를 제공하여 OHS 시스템의 요구사항과 완벽하게 일치합니다.
- 최종 권장 사항**: Ant Design.** 시스템이 복잡한 대시보드, 데이터 테이블, 양식에 중점을 둔다는 점을 고려할 때, Ant Design은 가장 큰 즉시 사용 가능한 이점을 제공하여 개발 시간과 위험을 줄여줍니다.

3.3. 전략적 비즈니스 자산으로서의 디자인 시스템

디자인 시스템은 단순히 디자이너를 위한 "UI 키트"나 개발자를 위한 "컴포넌트 라이브러리"가 아닙니다. 그것은 전체 제품 조직을 정렬시키는 공유된 언어입니다. Ant Design의 가치("자연스러움", "확실함", "의미 있음")가 일관된 사용자 경험을 만드는 방법과, 공유 프레임워크가 디자이너, 개발자, 콘텐츠 제작자 간의 협업을 강화하여 효율성과 품질을 높이는 방법에서 이를 확인할 수 있습니다. 디자이너가 디자인 시스템에서 미리 정의된 "모달" 컴포넌트를 사용하면, 개발자는 어떤 동작을 구현해야 할지 정확히 알고 제품 관리자는 무엇을 기대해야 할지 알게 됩니다. 성숙한 디자인 시스템을 채택하는 것은 프로젝트의 위험을 줄이고 장기적인 ROI를 향상시킵니다. 시스템을 학습하고 구성하는 초기 노력은 공통 UI 패턴을 설계, 구축 및 테스트하는 데 소요되는 시간을 극적으로 단축시킴으로써 상쇄됩니다. 이는 더 빠른 기능 제공, 더 적은 버그 수, 신규 개발자의 더 쉬운 온보딩, 그리고 단일 제품뿐만 아니라 전체 애플리케이션 포트폴리오에 걸쳐 더 응집력 있는 사용자 경험으로 이어집니다. 이는 UI 개발을

일회성 프로젝트의 연속에서 확장 가능하고 예측 가능한 프로세스로 전환시킵니다.

제 **4**장**:** 아토믹 디자인 프레임워크**:** 확장 가능한 컴포넌트 라이브러리 구축

본 장에서는 리팩토링 과정에서 프론트엔드 코드베이스를 구조화하는 구체적인 방법론을 제공하여, 새로운 UI가 모듈식이고 재사용 가능하며 유지보수가 용이하도록 보장합니다.

4.1. 아토믹 디자인 방법론 소개

아토믹 디자인 개념은 UI를 컴포넌트의 계층 구조로 구축하는 사고 모델입니다.

- 원자 **(Atoms):** 기본 구성 요소 (버튼, 입력 필드, 레이블, 아이콘).
- 분자 **(Molecules):** 함께 기능하는 원자의 간단한 그룹 (예: 검색 버튼이 있는 검색 입력 필드).
- 유기체 **(Organisms):** 분자와 원자로 구성된 복잡한 UI 컴포넌트 (예: 필터링 및 정렬 컨트롤이 있는 데이터 테이블 헤더, 완전한 양식).
- 템플릿 **(Templates):** 유기체를 레이아웃에 배치하는 페이지 수준의 구조.
- 페이지 **(Pages):** 실제 데이터가 포함된 템플릿의 특정 인스턴스.

4.2. Ant Design 컴포넌트를 아토믹 프레임워크에 매핑하기

Ant Design 라이브러리가 이 구조에 어떻게 부합하는지에 대한 실용적인 매핑을 제공합니다.

- 원자 **(Atoms):** Button, Icon, Typography.Text.
- 분자 **(Molecules):** Input.Search, DatePicker, Select, 레이블이 있는 Checkbox.
- 유기체 **(Organisms):** Table, Form, Card, PageHeader.

4.3. 리팩토링을 위한 아토믹 접근 방식의 이점

- 재사용성 및 일관성**:** 한 번 컴포넌트를 만들면 어디서든 사용할 수 있습니다. 애플리케이션의 모든 버튼이나 양식 필드가 동일하게 보이고 동작하도록 보장합니다.
- 더 빠른 개발**:** 기존 유기체 라이브러리에서 새로운 페이지를 신속하게 조립할 수 있습니다.
- 단순화된 테스트**:** 원자와 분자를 격리하여 테스트함으로써 큰 페이지 테스트의 복잡성을 줄입니다.
- 명확한 커뮤니케이션**:** 디자이너와 개발자에게 공유된 어휘를 제공합니다 ("이 템플릿에는 새로운 '사고 세부 정보' 유기체가 필요합니다").

4.4. 확장 가능한 리팩토링의 엔진으로서의 아토믹 디자인

아토믹 디자인은 사용자의 "리팩토링" 요청에 대한 명확하고 구조화된 경로를 제공합니다. 이는 단순히 코드를 다시 작성하는 것이 아니라, 더 논리적이고 유지보수 가능한 시스템으로 재설계하는 것입니다. 순진한 접근 방식은 각 페이지를 하나씩 다시 작성하는 것이지만, 이는 종종 중복된 코드와 불일치를 초래합니다. 아토믹 디자인 방법론은 체계적인 대안을 제공합니다. 팀은 선택한 디자인 시스템(Ant Design)을 기반으로 "원자" 라이브러리(버튼, 입력 필드)를 구축하는 것으로 시작할 수 있습니다. 그런 다음 "분자"(검색 바, 날짜 선택기)를 만듭니다. 이 기반 위에서, 기존 애플리케이션의 일부를 새로운 견고한 "유기체"(예: 새로운 데이터 테이블)로 교체하기 시작할 수 있습니다. 이러한 접근 방식은 점진적 리팩토링 프로세스를 가능하게 합니다. 팀은 "빅뱅" 릴리스를 위해 전체 시스템을 오프라인으로 전환할 필요가 없습니다. 새로운 컴포넌트 라이브러리를 도입하고 기존 페이지 내의 레거시 UI 요소를

점진적으로 새로운 아토믹 컴포넌트로 교체할 수 있습니다. 이는 위험을 줄이고, 가치의 지속적인 제공을 가능하게 하며, 전체 재설계라는 기념비적인 작업을 관리 가능하게 만듭니다. 이는 라이브 상태의 복잡한 엔터프라이즈 애플리케이션을 마이그레이션하는 실질적인 과제를 직접적으로 해결합니다.

제 **3**부**:** 핵심 화면 및 컴포넌트 심층 분석

이 파트에서는 1부와 2부에서 수립된 원칙과 프레임워크를 적용하여 보건관리시스템의 가장 중요한 컴포넌트에 대한 상세한 설계 청사진을 제공합니다.

제 **5**장**:** 중앙 허브**:** 역할 기반 메인 대시보드 설계

본 장에서는 애플리케이션의 가장 중요한 화면 설계를 상세히 다루며, 적시에 적절한 사용자에게 올바른 정보를 전달하는 데 중점을 둡니다.

5.1. 대시보드 설계 원칙**: "5**초 규칙**"**

사용자는 대시보드를 본 후 5초 이내에 가장 중요한 정보를 이해할 수 있어야 합니다.

- 핵심 원칙**:**
  - 시각적 계층 구조**:** 가장 중요한 정보가 시각적으로 가장 두드러져야 합니다 (왼쪽 상단, 더 큰 크기, 더 강한 색상).
  - 데이터 시각화**:** 올바른 데이터에 올바른 차트를 사용합니다 (예: 추세에는 라인 차트, 비교에는 막대 차트). 오해의 소지가 있는 시각화는 피합니다.
  - 단순성**:** 정보 과부하를 피하기 위해 위젯/카드 수를 5-9개로 제한합니다. 여백을 효과적으로 사용합니다.

5.2. 페르소나별 대시보드 와이어프레임 및 목업

각 주요 페르소나에 대한 시각적 디자인을 제시합니다.

- 보건안전 관리자 대시보드 **(**분석**/**운영**):**
  - 레이아웃**:** 카드 기반 그리드.
  - 핵심 유기체**:** "처리 대기 중인 사고 보고서" 목록, "기한 초과 시정 조치" 추적기, "규정 준수 현황" 차트, "예정된 감사" 캘린더, "실시간 안전 관찰" 피드.
- 임원 대시보드 **(**전략**):**
  - 레이아웃**:** 몇 가지 영향력 큰 KPI에 집중.
  - 핵심 유기체**:** "시간 손실 재해율(LTIFR)", "총 기록 재해율(TRIR)", "전체 위험 점수", "교육 이수율(%)"을 위한 대형 스코어카드 위젯.
- 직원 포털 **(**과업 중심**):**
  - 레이아웃**:** 단순하고 행동 지향적인 인터페이스.
  - 핵심 유기체**:** 대형 "사고/유해 요인 보고" 버튼, "나의 예정된 건강 검진" 카드, "나의 필수 교육" 목록.

5.3. 상호작용**:** 드릴다운 및 필터

대시보드는 정적인 보고서가 아닙니다. 모든 차트와 KPI는 상호작용이 가능해야 하며, 사용자가 클릭하여 기본 상세 데이터를 볼 수 있도록 해야 합니다. 사용자가 뷰를 사용자

정의할 수 있도록 전역 필터(예: 날짜 범위, 사업장, 부서별)가 제공되어야 합니다.

5.4. 내비게이션 허브로서의 대시보드

잘 설계된 대시보드는 단순한 정보 표시가 아니라 주요 내비게이션 도구입니다. 이는 사용자 워크플로우를 위한 동적이고 개인화된 시작점 역할을 합니다. IA의 "정문 원칙"에 따르면 사용자는 여러 다른 페이지를 통해 사이트에 들어옵니다. 대시보드는 그중 주요 정문입니다. 사용자에게 정적인 메뉴를 통해 탐색하도록 강요하는 대신(예: 운영 -> 사고 -> 미결건 보기), 대시보드는 "검토가 필요한 5개의 신규 사고"라는 카드를 제시할 수 있습니다. 이 카드를 클릭하면 사용자는 해당 5개 사고의 사전 필터링된 목록으로 직접 이동합니다. 이는 대시보드를 수동적인 보고 도구에서 능동적인 "작업 큐"로 변환시킵니다. 시스템이 사용자의 워크플로우를 안내함으로써, 사용자가 자신의 작업을 찾기 위해 어디로 가야 할지 기억하도록 강요하는 대신, 효율성과 사용자 참여를 극적으로 향상시킵니다. 이는 애플리케이션을 지능적이고 유용하게 만드는 데 중요한 과업 중심 접근 방식입니다.

제 **6**장**:** 데이터 마스터하기**:** 반응형 데이터 테이블 및 그리드 시스템

본 장에서는 모든 엔터프라이즈 애플리케이션의 핵심인 데이터 테이블 설계에 대한 포괄적인 가이드를 제공합니다. 모든 화면 크기에서 복잡한 데이터를 효과적으로 표시하는 과제를 해결하는 데 중점을 둡니다.

6.1. 고기능성 데이터 테이블의 해부학

MUI나 Ant Design과 같은 컴포넌트 라이브러리의 강력한 기능을 활용하여 필수 기능에 대한 상세한 분석을 제공합니다.

- 핵심 기능**:** 열 정렬, 고급 필터링(열별), 전역 검색, 페이지네이션, 행 선택(체크박스), 사용자 정의 가능한 열 가시성.
- 상호작용 요소**:** 인라인 편집, 행별 작업 버튼/메뉴(예: "편집", "삭제", "상세 보기"), 상태 칩, 툴팁.

6.2. 반응형 과제**:** 수평 스크롤을 넘어서

모바일에서 흔하지만 사용자에게 불편한 수평 스크롤 패턴에 대한 비판과 함께 대안을 제시합니다.

- 제안 해결책**:** 접이식 행 패턴**(Collapsible Row Pattern).**
    - 데스크톱에서는 전체 테이블을 표시합니다.
    - 모바일 및 태블릿 브레이크포인트에서는 테이블이 변형됩니다. 가장 중요한 3-4개의 열(예: 이름, 날짜, 상태)만 표시합니다.
    - 각 행에는 확장/축소 아이콘이 있습니다. 이 아이콘을 탭하면 해당 행의 나머지 데이터가 바로 아래에 잘 형식화된 "카드" 뷰로 표시됩니다. 이 패턴은 Material-UI의 접이식 테이블에서 잘 예시됩니다.

6.3. 시각 디자인 및 가독성

- 타이포그래피**:** 명확한 산세리프 글꼴을 사용합니다. 전체 대문자 사용은 피합니다.
- 간격**:** 훑어보기 용이하도록 넉넉한 행 높이와 패딩을 사용합니다.
- 레이아웃**:** 텍스트는 왼쪽 정렬, 숫자는 오른쪽 정렬하여 쉽게 비교할 수 있도록 합니다. 날짜와 값에 일관된 형식을 사용합니다.

- 지브라 스트라이핑**:** 미묘한 교차 행 색상을 사용하여 사용자가 수평으로 스캔하는 동안 위치를 유지하도록 돕습니다.

6.4. 데이터 테이블은 정적 디스플레이가 아닌 애플리케이션이다

현대의 데이터 테이블은 단순히 데이터를 표시하는 것이 아니라, 자체적인 상태, 상호작용, 워크플로우를 가진 미니 애플리케이션입니다. 정렬, 필터링, 페이지네이션, 선택과 같은 기능들은 각각 관리되어야 할 상태입니다. 사용자가 열을 정렬하면 UI는 다시 렌더링되어야 하고, 필터를 입력하면 데이터는 다시 가져오거나 클라이언트 측에서 필터링되어야 합니다. 이는 복잡한 애플리케이션 로직입니다. 이러한 미니 애플리케이션을 처음부터 구축하는 것은 오류가 발생하기 쉽고 시간이 많이 소요됩니다. 이는 성숙한 컴포넌트 라이브러리(Ant Design 또는 MUI)와 정교한 데이터 페칭 라이브러리(React Query)를 사용해야 하는 강력한 논거가 됩니다. React Query는 서버 측 상태(전체 데이터셋 캐싱, 페이지네이션 및 필터링 API 호출 처리)를 관리할 수 있으며, 컴포넌트 라이브러리는 견고하고 접근 가능하며 반응형인 UI 컴포넌트를 제공합니다. 이를 단순한 useState와 기본 HTML <table>로 구축하려는 시도는 버그가 많고 성능이 낮으며 유지보수가 불가능한 결과를 낳아, "리팩토링"이라는 목표와 정면으로 배치됩니다.

제 **7**장**:** 내비게이션 프레임워크**:** 반응형 및 모바일 우선 내비게이션 구현

본 장에서는 반응형 내비게이션 및 모바일 지원에 대한 사용자의 핵심 요청을 직접적으로 다루며, 구체적인 디자인 및 구현 전략을 제공합니다.

7.1. 모바일 우선 디자인 철학

모바일 우선 접근 방식은 가장 작은 화면을 먼저 디자인한 다음, 점진적으로 더 큰 화면을 위해 향상시키는 것입니다.

- 이점**:** 콘텐츠의 우선순위를 정하게 하고, 더 깨끗한 코드로 이어지며, 모바일 성능을 향상시키고, 자연스럽게 모든 장치에서 더 나은 사용자 경험을 제공합니다.

7.2. 제안된 내비게이션 패턴

- 데스크톱 **(**대형 화면 **> 1200px):**
  - 왼쪽에 고정된 접이식 수직 사이드바. 이는 많은 내비게이션 항목이 있는 애플리케이션에 이상적이며, 충분한 공간과 명확한 계층 구조를 제공합니다. IA의 최상위 카테고리(작업자 관리, 운영 관리 등)가 주 항목이 됩니다.
- 태블릿 **(**중형 화면 **> 768px):**
  - 사이드바는 기본적으로 아이콘만 표시하도록 축소됩니다. "햄버거" 메뉴 아이콘을 사용하여 확장할 수 있습니다. 이는 내비게이션 접근성을 유지하면서 수평 공간을 절약합니다.
- 모바일 **(**소형 화면 **< 768px):**
  - 사이드바는 완전히 숨겨집니다. 상단 헤더의 햄버거 메뉴가 드로어/오프캔버스 패널에서 전체 내비게이션 메뉴를 표시합니다.
  - 가장 빈번하게 사용되는 핵심 작업(특히 직원 페르소나를 위해)에는 하단 탭 바를 사용합니다. 이는 3-5개의 주요 영역(예: 홈/대시보드, 보고, 내 교육, 프로필)에 대한 지속적이고 한 번의 탭으로 접근할 수 있는 접근성을 제공합니다. 이는 표준적이고 매우 인체공학적인 모바일 UX 패턴입니다.

7.3. 반응형 그리드 시스템

모든 레이아웃을 지배할 기본 그리드를 정의합니다. 유연한 12열 그리드 시스템을 사용할 것입니다. Bootstrap이나 Material Design과 같은 프레임워크의 표준에 맞춰 일반적인 장치 크기를 기반으로 특정 브레이크포인트를 정의합니다.

표 **2:** 반응형 브레이크포인트 및 그리드 구성

브레이크포인트	최소 너비	열(Columns)	여백(Gutters)	마진(Margins)
xs (Mobile)	0px	4	16px	16px
sm (Tablet)	600px	8	24px	24px
md (Laptop)	960px	12	24px	24px
lg (Desktop)	1280px	12	24px	32px
xl (Large Desktop)	1920px	12	24px	32px

이 표는 개발자를 위한 모든 모호성을 제거합니다. 애플리케이션의 반응형 동작에 대한 단일 진실 공급원(Single Source of Truth)을 제공하고, 모든 페이지와 컴포넌트에서 일관성을 보장하며, 광범위한 장치에서 작동하는 것으로 입증된 모범 사례를 기반으로 하므로 상당한 테스트 및 디버깅 시간을 절약할 수 있습니다.

제 **4**부**:** 기술 아키텍처 및 리팩토링 전략

이 마지막 파트에서는 새로운 프론트엔드의 기반이 될 기술 스택에 대한 실행 가능한 권장 사항과 프로젝트 실행을 위한 상위 수준의 계획을 제공합니다. 이는 사용자 쿼리의 "리팩토링" 요구를 직접적으로 다룹니다.

제 **8**장**:** 프론트엔드 아키텍처 및 기술 스택 권장 사항

본 장에서는 새로운 프론트엔드의 중추를 형성할 핵심 라이브러리에 대한 구체적이고 정당화된 권장 사항을 제시합니다.

8.1. 상태 관리 논쟁**:** 클라이언트 상태 **vs.** 서버 상태

현대 React 개발의 중요한 개념은 모든 상태가 동일하지 않다는 것입니다.

- 서버 상태**:** 서버에 존재하며 API를 통해 가져오는 데이터입니다. 캐싱, 재검증, 동기화와 같은 속성을 가집니다 (예: 사고 목록).
- 클라이언트 상태**:** 브라우저에만 존재하는 데이터입니다 (예: "내비게이션 메뉴가 열려 있는가?", 현재 테마).
- 아키텍처 원칙**:** 올바른 작업에 올바른 도구를 사용합니다. 서버 상태는 전용 데이터 페칭 라이브러리로 관리합니다.

8.2. 데이터 페칭 라이브러리**: React Query vs. SWR**

- 비교**:** 둘 다 훌륭하지만, React Query는 일반적으로 더 많은 기능을 갖추고 있으며, 뮤테이션, 페이지네이션 및 더 고급 캐싱 전략에 대한 내장 지원을 제공하여 복잡한 애플리케이션에 이상적입니다. SWR은 더 간단하고 가볍습니다.
- 권장 사항**: React Query (TanStack Query).** 수많은 CRUD(생성, 읽기, 업데이트, 삭제) 작업, 복잡한 데이터 테이블, 낙관적 업데이트와 같은 기능이 필요한 데이터 집약적인

OHS 시스템의 경우, React Query의 강력한 기능 세트는 상당한 이점을 제공하며 더 깨끗하고 유지보수하기 쉬운 코드로 이어질 것입니다.

8.3. 전역 클라이언트 상태 관리**: Redux Toolkit vs. Zustand**

- 비교**:** 서버 상태를 React Query로 옮긴 후, 전역 클라이언트 상태의 양은 훨씬 작아집니다. Zustand는 간단하고 보일러플레이트가 없는 API를 제공합니다. Redux Toolkit(RTK)은 더 구조화되어 있으며 강력한 디버깅 도구를 갖추고 있습니다.
- 권장 사항**: Redux Toolkit.**
- 엔터프라이즈 환경에서 구조는 버그가 아닌 기능이다**.** Zustand는 소규모 프로젝트나 프로토타입에서 시작하기 더 빠르지만, 강제된 구조의 부재는 팀이 개발하는 대규모의 장기적인 엔터프라이즈 애플리케이션에서 부채가 될 수 있습니다. OHS 시스템은 미션 크리티컬하며, 버그는 심각한 결과를 초래할 수 있습니다. 복잡한 상태 상호작용을 디버깅하는 능력은 무엇보다 중요합니다. 시간 여행 디버깅과 액션 로깅을 갖춘 Redux DevTools는 이를 위한 비할 데 없는 도구입니다. RTK의 "보일러플레이트"(슬라이스, 리듀서, 액션)는 예측 가능하고 단방향적인 데이터 흐름을 강제하여, 특히 팀에 새로 합류하는 개발자가 애플리케이션의 로직을 더 쉽게 이해할 수 있도록 만듭니다. 따라서 최적의 아키텍처는 모든 서버 상태에 **React Query**를, 그리고 남아있는 중요한 전역 클라이언트 상태**(**인증**,** 권한**,** 단일 컴포넌트에 묶이지 않은 복잡한 **UI** 상태 등**)**에 **Redux Toolkit**을 사용하는 조합입니다. 이 하이브리드 접근 방식은 데이터 페칭을 위한 React Query의 강력함과 단순함, 그리고 핵심 애플리케이션 상태를 위한 Redux의 구조, 예측 가능성, 디버깅 용이성이라는 두 세계의 장점을 모두 제공합니다.

표 **3:** 상태 관리 및 데이터 페칭 라이브러리 비교 분석

항목	React Query (TanStack Query)	SWR	Redux Toolkit	Zustand
주요 사용 사례	복잡한 서버 상태 관리 (캐싱, 동기화, 뮤테이션)	간단한 서버 상태 관리 (주로 데이터 읽기)	복잡한 전역 클라이언트 상태 관리	간단한 전역 클라이언트 상태 관리
핵심 기능	뮤테이션, 페이지네이션, 낙관적 업데이트 내장	Stale-While-Revalidate, 자동 재시도	시간 여행 디버깅, 미들웨어, 구조화된 패턴	보일러플레이트 최소화, 훅 기반 API
번들 크기	큼 (~50KB)	작음 (~15KB)	큼 (~15KB)	매우 작음 (~4KB)
학습 곡선	다소 높음	낮음	중간	매우 낮음
디버깅	전용 DevTools 제공	제한적	뛰어난 DevTools 통합	제한적
권장 상황	데이터 중심의 대규모 엔터프라이즈 애플리케이션	가볍고 빠른 데이터 페칭이 필요한 프로젝트	예측 가능성과 디버깅이 중요한 대규모 팀/프로젝트	빠른 프로토타이핑, 중소 규모 프로젝트

제 **9**장**:** 단계적 구현 및 출시 로드맵

이 마지막 장에서는 전체 재설계 및 리팩토링이라는 어려운 작업을 관리 가능하게 만들고 가치를 점진적으로 제공하기 위한 상위 수준의 전략적 계획을 제공합니다.

9.1. 1단계**:** 기반 및 파일럿 모듈 **(1-3**개월**)**

- 설정**:** 새 리포지토리를 구축하고, 빌드 도구를 구성하며, 선택한 기술 스택(React, Ant Design, React Query, Redux Toolkit)을 설정합니다.
- 디자인 시스템**:** 아토믹 디자인 시스템의 핵심 "원자"와 "분자"를 구축합니다.
- 파일럿 모듈**:** 초기 리팩토링을 위해 독립적이고 가치가 높은 모듈 하나(예: "안전 교육 관리")를 선택합니다. 이 모듈을 새로운 아키텍처와 디자인 시스템을 사용하여 완전히 재구축합니다.

9.2. 2단계**:** 핵심 워크플로우 현대화 **(4-9**개월**)**

- 1부에서 식별된 가장 복잡하고 중요한 워크플로우를 다룹니다.
- 집중 영역**:** 주요 역할 기반 대시보드, 사고 관리 워크플로우, 핵심 위험성 평가 도구를 재구축합니다.
- 이 단계에서는 반응형 데이터 테이블과 같은 가장 복잡한 "유기체"를 구축하게 됩니다.

9.3. 3단계**:** 전체 시스템 마이그레이션 및 레거시 폐기 **(10-12**개월 이상**)**

- 남아있는 덜 중요한 모듈(예: 관리 설정, 오래된 보고 기능)을 체계적으로 리팩토링합니다.
- 필요한 경우, 리버스 프록시를 사용하여 사용자를 적절한 버전으로 라우팅함으로써 신규 및 기존 시스템을 병렬로 실행하는 전략을 구현합니다.
- 최종 사용자 인수 테스트(UAT), 성능 튜닝을 수행하고, 전체 출시 및 레거시 프론트엔드 폐기를 준비합니다.

9.4. 점진적 출시는 위험을 완화하고 진행 상황을 보여준다

이러한 복잡성을 가진 시스템에 대한 "빅뱅" 릴리스는 매우 위험합니다. 단계적 접근 방식을 통해 팀은 학습하고 적응하며, 파일럿 모듈에 대한 사용자 피드백을 수집하고, 이해관계자에게 실질적인 진행 상황을 조기에 자주 보여줄 수 있습니다. 이는 프로젝트에 대한 추진력과 신뢰를 구축하여 지속적인 지원과 궁극적인 성공을 보장합니다.

참고 자료

1. Top 10 Occupational Health Management Software Solutions 2025 – Boost Workplace Safety - SPRY PT, https://www.sprypt.com/blog/occupational-health-management-software 2. 7 Best Occupational Health Software of 2025 - Safety Culture, https://safetyculture.com/app/occupational-health-software/ 3. What is EHS Software? Benefits, Features & Interactive Builder, https://www.optial.com/resources/what-is-ehs-software 4. What is EHS Software - EHSSoftware.io, https://blog.ehssoftware.io/safetyinsiderblog/understanding-ehs-software-a-comprehensive-guide 5. Key Features of an EHS Software - SafetyIQ, https://safetyiq.com/insight/key-features-of-an-ehs-software/ 6. 안전보건관리시스템은 무엇을 특징으로 하는가, https://www.anjunj.com/news/articleView.html?idxno=34837 7. Best Occupational Health Software | Meddbase 2025, https://www.meddbase.com/us/occupational-health-software/ 8. Effective Dashboard Design Principles for 2025 - UXPin, https://www.uxpin.com/studio/blog/dashboard-design-principles/ 9. 30 Proven Dashboard Design Principles for Better Data Display - Aufait UX, https://www.aufaitux.com/blog/dashboard-design-principles/ 10. What is Website Information Architecture - The Complete Guide - Userlytics,

https://www.userlytics.com/resources/blog/website-information-architecture/ 11. 8 Useful Principles for a Better Information Architecture - Make:Iterate, https://makeiterate.com/8-useful-principles-for-a-better-information-architecture/ 12. What should information architecture NOT contain? : r/UXDesign - Reddit, https://www.reddit.com/r/UXDesign/comments/1fbr8fi/what_should_information_architecture_not_contain/ 13. What is Task-Oriented Design? | IxDF - The Interaction Design Foundation, https://www.interaction-design.org/literature/topics/task-oriented-design 14. Navigation UX: Pattern Types and Tips to Enhance User Experience - Userpilot, https://userpilot.com/blog/navigation-ux/ 15. Designing Effective Information Architectures for Large-Scale Web Sites - UXmatters, https://www.uxmatters.com/mt/archives/2024/09/designing-effective-information-architectures-for-large-scale-web-sites.php 16. 2025년 웹페이지 UI/UX 디자인 트렌드 알아보기 - 브런치, https://brunch.co.kr/@7217b71f43c34f7/93 17. 2025년 UI/UX 디자인 트렌드: 주목해야 할 5가, https://brunch.co.kr/@leoyh23/4 18. What is Mobile-First Design and Why Does It Matter? - MindInventory, https://www.mindinventory.com/blog/what-is-mobile-first-design/ 19. Mobile First Design: What it is & How to implement it | BrowserStack, https://www.browserstack.com/guide/how-to-implement-mobile-first-design 20. Guidelines - Material Design, https://m2.material.io/design/guidelines-overview 21. Material Design 3 - Google's latest open source design system, https://m3.material.io/ 22. Visualization Page - Ant Design, https://ant.design/docs/spec/visualization-page/ 23. Ant Design 101 – Introduction to a Design System for Enterprises | UXPin, https://www.uxpin.com/studio/blog/ant-design-introduction/ 24. About UX: An Overview of Ant Design Guidelines | by AxureBoutique, https://axureboutique.medium.com/about-ux-an-overview-of-ant-design-guidelines-560f4efefe4f 25. atomicdesign.bradfrost.com, https://atomicdesign.bradfrost.com/chapter-2/#:~:text=Atomic%20design%20is%20atoms%2C%20molecules,parts%20at%20the%20same%20time. 26. Challenges and Strategies for Designing Data-Intensive Applications - Eleken, https://www.eleken.co/blog-posts/how-to-design-data-intensive-applications-cases-and-practices 27. Dashboard Design: best practices and examples - Justinmind, https://www.justinmind.com/ui-design/dashboard-design-best-practices-ux 28. Data tables - Material Design, https://m2.material.io/components/data-tables 29. React Table component - Material UI - MUI, https://mui.com/material-ui/react-table/ 30. Data Table UI Design Guide & 30+ Examples - wpDataTables, https://wpdatatables.com/table-ui-design/ 31. Embracing Mobile First | Medium, https://medium.com/@philippebeck/css3-mobile-first-79ef48249a7c 32. Mobile-First Design Explained | Definition & Benefits - Sanity, https://www.sanity.io/glossary/mobile-first-design 33. Navigation design: Almost everything you need to know - Justinmind, https://www.justinmind.com/blog/navigation-design-almost-everything-you-need-to-know/ 34. Responsive web design best practices and examples [2025 guide] - Webflow, https://webflow.com/blog/responsive-web-design 35. Ultimate Guide to Grid Layouts for Web Apps - YouTube, https://www.youtube.com/watch?v=X6saBZFgj1M 36. Responsive layout grid - Material Design, https://m2.material.io/design/layout/responsive-layout-grid.html 37. The Beginner's Guide to Responsive Web Design (Code Samples & Layout Examples), https://kinsta.com/blog/responsive-web-design/ 38. React State Management — Part 1 : Comparison of Redux Toolkit, React Query, Zustand, and Context API | by David Zhao | Medium, https://medium.com/@david.zhao.blog/react-state-management-packages-13eb889a1c5f 39.

React Query vs SWR: Which One is Better? - NamasteDev Blogs, https://namastedev.com/blog/react-query-vs-swr-which-one-is-better-4/ 40. SWR vs React Query: Choosing the Right Data Fetching Library - Gerardo Perrucci, https://www.gperrucci.com/en/blog/react/swr-vs-react-query 41. Optimizing React Apps with SWR and React Query: A Technical Deep Dive - DEV Community, https://dev.to/hamzakhan/optimizing-react-apps-with-swr-and-react-query-a-technical-deep-dive-57ee 42. React Query or SWR: Which is best in 2025? - DEV Community, https://dev.to/rigalpatel001/react-query-or-swr-which-is-best-in-2025-2oa3 43. Zustand vs. Redux Toolkit vs. Jotai | Better Stack Community, https://betterstack.com/community/guides/scaling-nodejs/zustand-vs-redux-toolkit-vs-jotai/ 44. Zustand vs. Redux Toolkit: When I Pick One Over the Other and Why - Medium, https://medium.com/@raelsei/zustand-vs-redux-toolkit-when-i-pick-one-over-the-other-and-why-1cb442433300 45. React Redux vs Zustand – Which one should I go with? : r/reactjs - Reddit, https://www.reddit.com/r/reactjs/comments/1m4g1w7/react_redux_vs_zustand_which_one_should_i_go_with/ 46. State Management in 2025: When to Use Context, Redux, Zustand, or Jotai, https://dev.to/hijazi313/state-management-in-2025-when-to-use-context-redux-zustand-or-jotai-2d2k