ChatGPT

OK금융그룹 HRIS 메뉴 및 인터페이스 설계 문서

1. 핵심업무 모듈 메뉴 구조

OK금융그룹의 새로운 HRIS(Human Resources Information System)는 주요 HR 업무를 다음과 같은 **7개 핵심 모듈**로 구성합니다. 각 모듈은 HR 프로세스의 한 영역을 담당하며, 시스템 내 메뉴로 제공됩니다:

- **대시보드:** 전체 HR 현황과 주요 지표를 한 눈에 보여주는 메인 화면입니다. 사용자 개인화 정보, 공지사항, 할 일 등을 포함하며 다른 모듈의 핵심 정보를 요약 제공합니다.
- **직원관리:** 직원 개개인의 정보와 근속 현황을 관리하는 모듈입니다. 신규 입사자 등록, 퇴사자 처리, 인사 기록 열람, 휴가 및 근태 관리 등 **직원의 라이프사이클 전반**을 다룹니다.
- **평가관리:** 성과 목표 설정부터 **직원 평가(인사고과)**까지를 지원하는 모듈입니다. 목표 수립, 중간 피드백, 연말 성과평가 등의 프로세스를 관리하며 평가 결과와 피드백 기록을 볼 수 있습니다.
- **보상관리:** 급여와 보너스 등 **금전적 보상** 관련 업무 모듈입니다. 직원 급여 명세 관리, 성과급/인센티브 지급, 복리후생 내역 조회 및 승진/연봉 인상 등의 기능을 포함합니다.
- **조직관리:** 회사 조직 구조와 직책/직급 정보를 관리하는 모듈입니다. **조직도 열람**, 부서 정보 수정, 직위/직급 체계 관리, 조직 개편 이력 등의 기능을 제공하여 조직 체계를 체계적으로 관리합니다.
- **채용관리:** 신규 인재 채용 과정을 지원하는 모듈입니다. **채용 공고 생성/공유**, 지원자 목록 및 이력서 열람, 전형 단계별 현황 관리, **면접 일정 관리** 및 채용 결과 기록 등을 포함합니다.
- **AIRISS:** AI 기반 HR 지원 기능 모듈입니다. OK금융그룹의 새로운 HR 제도에 맞춘 **인사 데이터 분석 및 인사이트 제공**을 담당합니다. 예를 들어 HR 데이터 대시보드, AI 예측 분석 리포트, HR 챗봇 상담 등이 해당되며, 최신 AI 기술을 활용해 의사결정을 지원합니다.

이상 7개의 모듈이 사이드바 등의 **메인 메뉴**에 노출되어 있으며, 사용자는 필요한 HR 업무 영역을 선택해 들어갈 수 있습니다.

2. 모듈별 서브메뉴 JSON 구조 정의

각 핵심 모듈은 세부 업무 화면(서브메뉴)을 가지고 있습니다. React와 Vue 양쪽에서 동일한 구조의 데이터를 활용할 수 있도록, 메뉴와 하위 메뉴를 **JSON 구조**로 정의합니다. 아래는 각 모듈 및 서브메뉴의 예시 JSON 데이터 구조입니다:

```
[
 {
  "title": "대시보드",
  "path": "/dashboard",
  "icon": "HomeIcon",
  "children": []
 },
 {
  "title": "직원관리",
  "path": "/employees",
  "icon": "UserGroupIcon",
  "children": [
   { "title": "직원 목록", "path": "/employees/list" },
   { "title": "입사자 관리", "path": "/employees/onboarding" },
```

```
    { "title": "퇴사자 관리", "path": "/employees/offboarding" },
    { "title": "근태 관리", "path": "/employees/attendance" }
   ]
  },
  {
   "title": "평가관리",
   "path": "/performance",
   "icon": "ClipboardListIcon",
   "children": [
    { "title": "목표 설정", "path": "/performance/goals" },
    { "title": "성과 평가", "path": "/performance/reviews" },
    { "title": "피드백 관리", "path": "/performance/feedback" }
   ]
  },
  {
   "title": "보상관리",
   "path": "/compensation",
   "icon": "CurrencyDollarIcon",
   "children": [
    { "title": "급여 관리", "path": "/compensation/salary" },
    { "title": "보너스/인센티브", "path": "/compensation/bonus" },
    { "title": "복리후생", "path": "/compensation/benefits" }
   ]
  },
  {
   "title": "조직관리",
   "path": "/organization",
   "icon": "BuildingOfficeIcon",
   "children": [
    { "title": "조직도", "path": "/organization/orgchart" },
    { "title": "부서 관리", "path": "/organization/departments" },
    { "title": "직위 관리", "path": "/organization/positions" }
   ]
  },
  {
   "title": "채용관리",
   "path": "/recruitment",
   "icon": "BriefcaseIcon",
   "children": [
    { "title": "채용 공고", "path": "/recruitment/postings" },
    { "title": "지원자 관리", "path": "/recruitment/applicants" },
    { "title": "면접 일정", "path": "/recruitment/interviews" }
   ]
  },
  {
   "title": "AIRISS",
   "path": "/airiss",
   "icon": "CogIcon",
   "children": [
    { "title": "HR 분석 대시보드", "path": "/airiss/analytics" },
    { "title": "AI 예측 분석", "path": "/airiss/predictions" }
```

```
  ]
 }
]
```

위 JSON 배열에서 **각 객체는 하나의 모듈을 나타내며**, `title`은 메뉴 표시 이름, `path`는 해당 화면의 라우팅 경로, `icon`은 메뉴에 표시할 아이콘(식별자)입니다. `children` 배열이 존재하는 경우 그 모듈 하위의 세부 메뉴들을 나타냅니다 (`대시보드`처럼 하위 메뉴가 없는 경우 빈 배열). 이 구조는 **프레임워크에 독립적**이므로 React/Vue 컴포넌트에서 가져와 메뉴 트리를 생성하는 용도로 재사용할 수 있습니다.

3. Tailwind CSS 기반 사이드바 컴포넌트 설계 예시

HRIS의 사이드바는 왼쪽에 고정되어 위에서 정의한 메뉴들을 표시하는 **네비게이션 바**입니다. Tailwind CSS 유틸리티와 shadcn/ui 라이브러리의 Sidebar 컴포넌트를 활용하여 구현할 수 있습니다. 아래는 React를 예시로 한 Sidebar 컴포넌트 코드 구조입니다 (shadcn/ui 컴포넌트 기반):

```
import { Home, Users, Award, DollarSign, Building, Briefcase, Brain } from "lucide-react";
import {
 Sidebar, SidebarContent, SidebarMenu, SidebarMenuItem, SidebarMenuButton
} from "@/components/ui/sidebar";
// 아이콘과 Sidebar 구성 요소 불러오기 (shadcn/ui에서 제공)

const menuItems = [
 { title: "대시보드", url: "/dashboard", icon: Home },
 { title: "직원관리", url: "/employees", icon: Users },
 { title: "평가관리", url: "/performance", icon: Award },
 { title: "보상관리", url: "/compensation", icon: DollarSign },
 { title: "조직관리", url: "/organization", icon: Building },
 { title: "채용관리", url: "/recruitment", icon: Briefcase },
 { title: "AIRISS", url: "/airiss", icon: Brain }
];

export function AppSidebar() {
 return (
  <Sidebar side="left" collapsible="icon"> {/* 좌측에 고정, 아이콘 축소형으로 접기 가능 */}
   <SidebarContent className="h-screen bg-white border-r">
    {/* 전체 높이 사이드바, 배경/테두리 Tailwind 클래스로 지정 */}
    <SidebarMenu>
     {menuItems.map(item => (
      <SidebarMenuItem key={item.title}>
       <SidebarMenuButton asChild>
        <a href={item.url} className="flex items-center space-x-2 px-3 py-2 hover:bg-gray-100">
         <item.icon className="w-5 h-5" />
         <span>{item.title}</span>
        </a>
       </SidebarMenuButton>
      </SidebarMenuItem>
     ))}
    </SidebarMenu>
   </SidebarContent>
```

```
    </Sidebar>
  );
}
```

위 코드에서는 `menuItems` 배열에 **모든 모듈의 메뉴명, 경로, 아이콘**을 정의하고, 이를 `.map()`으로 순회하여 Sidebar 메뉴 리스트를 렌더링합니다. Tailwind CSS 클래스 (`flex`, `items-center`, `space-x-2`, `px-3`, `py-2`, `hover:bg-gray-100` 등)을 사용해 각 메뉴 항목의 레이아웃과 스타일을 지정했습니다. shadcn/ui의 Sidebar 컴포넌트는 내부적으로 Radix UI와 Tailwind로 구성되어 있어, `collapsible="icon"`과 같은 속성을 통해 **사이드바를 아이콘만 보이는 축소 형태**로 토글할 수 있습니다.

또한 Tailwind CSS를 활용해 Sidebar의 스타일을 커스터마이징할 수 있습니다. 예를 들어 `w-64` 클래스를 사용하거나 `style={{ "--sidebar-width": "16rem" }}`와 같은 방식으로 사이드바 너비를 조절할 수 있으며, 반응형 디자인을 위해 Tailwind의 숨김/표시 클래스 (`hidden`, `md:block` 등)나 미디어 쿼리를 적용해 모바일 화면에서 사이드바를 자동으로 숨기고 햄버거 메뉴로 대체하는 패턴을 구현할 수 있습니다.

> **노트:** 상기 예시는 단일 레벨 메뉴 구조를 보여줍니다. 만약 `children` 하위 메뉴가 있는 모듈(예: 직원관리)의 경우, 사이드바에서 해당 항목을 클릭 시 하위 메뉴를 펼치는 기능을 추가할 수 있습니다. 이때 shadcn/ui의 `Collapsible` 컴포넌트나 `SidebarGroup` 등을 사용하여 **중첩 메뉴**를 표시하면 됩니다. 구현 방식은 프로젝트 필요에 따라 결정합니다.

4. Figma 구성 기준에 준하는 레이아웃 구조 설명

프론트엔드 구현에 앞서, Figma로 **UI 레이아웃을 설계**하여 일관된 디자인 가이드라인을 수립합니다. 이때 고려해야 할 사항들은 다음과 같습니다:

- **계층 구조(Hierarchy):** Figma 디자인에서는 사이드바, 헤더, 메인 콘텐츠 영역을 명확히 나누어 **프레임 및 그룹화**합니다. 예를 들어, 전체 화면을 담는 상위 Frame를 만들고 왼쪽에는 Sidebar 컴포넌트(Frame)로, 오른쪽에는 Content 영역 Frame으로 배치합니다. Sidebar는 별도의 컴포넌트로 만들어 재사용하며, 레이어 순서상 항상 콘텐츠보다 위/뒤에 고정되도록 설정합니다.
- **그리드 시스템(Grid System):** 일관된 정렬과 반응형 대응을 위해 **12컬럼 레이아웃 그리드**를 사용합니다. Desktop 기준 아트보드(예: 1440px 폭)에 12등분 컬럼 그리드를 적용하고 적절한 좌우 마진과 거터(gutter)를 설정합니다. 또한 세로 여백 및 컴포넌트 간 간격은 4px 또는 8px **기준 단위**로 잡아, 모든 패딩/마진이 8px의 배수(또는 4px 단위 보조)로 떨어지게 디자인합니다. 이렇게 하면 개발 시 Tailwind CSS의 spacing 클래스 (`p-2`, `p-4`, `m-2` 등이 4px/8px 기반)와 쉽게 매핑할 수 있어 픽셀 오차를 줄일 수 있습니다.
- **자동 레이아웃 및 반응형:** Figma의 Auto Layout 기능을 활용하여 **반응형 동작을 시뮬레이션**합니다. Sidebar와 메인 콘텐츠를 포함하는 컨테이너 Frame에 Auto Layout을 적용하고, Sidebar 컴포넌트 Frame은 고정 너비(예: 240px)로 설정합니다. 콘텐츠 영역은 남은 공간을 채우도록 `Fill container`로 지정해 창 크기에 따라 유동적으로 확장됩니다. 또한 각 구성 요소에 **Constraints(제약)**를 설정하여, 화면 크기가 줄어들 때 Sidebar Frame이 왼쪽에 고정되고 Content Frame이 너비에 따라 줄어드는 레이아웃을 미리 확인합니다.
- **반응형 디자인 기준:** 주요 디바이스 화면 크기에 맞춰 별도의 Figma **프레임**을 만들어 디자인을 검토합니다. 예를 들어 **모바일 (최대 640px, Tailwind `sm` 이하)**, **태블릿 (768~1024px, `md` ~ `lg`)**, **데스크톱 (1280px 이상, `xl` 이상)** 세 가지 대표 해상도로 화면을 설계합니다. 작은 화면에서는 사이드바를 숨기고 상단에 메뉴 토글 버튼(햄버거 버튼)을 배치하는 변화를 Figma에서 표현해 둡니다. 이러한 반응형 가이드에 따라 개발자는 Tailwind CSS의 반응형 유틸리티 (`sm:`, `md:`, `lg:` 접두어 클래스 등)를 사용하여 실제 구현 시 조건부 스타일을 적용하게 됩니다.

위와 같은 Figma 설계 원칙을 준수하면 디자인 단계에서부터 개발 구현까지 **일관성 있는 레이아웃**을 유지할 수 있습니다. 디자이너와 개발자는 그리드와 컴포넌트 단위 등이 합의된 상태에서 작업하여, 픽셀 단위의 UI 구현과 다양한 화면에서의 사용자 경험을 효과적으로 관리할 수 있습니다.

5. 구현 단계별 워크플로우 (Super Claude 지시서)

마지막으로, 위 전체 기능과 디자인을 실제로 구현하기 위한 단계별 가이드(workflow)를 Super Claude 시스템 명령어 형태로 작성하면 다음과 같습니다:

```
/sc:workflow --magic -persona-frontend
1. 핵심 모듈 및 화면 목록 정의: 새로운 HRIS에 포함될 대시보드, 직원관리, 평가관리 등 핵심 모듈과 각 모듈의 세부 화면 목록을 확정한다.
2. Figma에서 UI 레이아웃 설계: 사이드바를 포함한 전체 화면을 12컬럼 그리드로 디자인하고, 각 모듈별 화면의 와이어프레임을 작성한다. (데스크톱, 모바일 등 반응형 레이아웃 고려)
3. 메뉴 JSON 데이터 생성: 확정된 메뉴 구조에 따라 모듈 및 서브메뉴 정보를 포함하는 JSON 파일(또는 객체)을 작성한다. (예: menu.json에 title, path, icon, children 구조로 정리)
4. React 사이드바 컴포넌트 구현: shadcn/ui의 Sidebar 컴포넌트를 프로젝트에 추가하고, 3단계의 JSON 데이터를 불러와 `.map()`을 통해 메뉴 항목들을 렌더링하는 Sidebar 컴포넌트를 만든다. Tailwind CSS 클래스를 적용해 사이드바 스타일(너비, 배경, hover 효과 등)을 지정한다.
5. Vue 메뉴 컴포넌트 구현: Vue에서도 동일한 JSON 데이터를 사용하여 `<template>`에서 `v-for` 디렉티브로 메뉴를 출력하는 컴포넌트를 구현한다. (React의 Sidebar와 유사한 구조로, 예를 들어 Element UI 또는 Vuetify 컴포넌트를 활용하거나 커스텀 구성)
6. Tailwind CSS 스타일링 및 기능 테스트: 사이드바가 의도한 대로 표시되는지 확인하고, 모바일 화면에서 사이드바 토글(축소/확장)이 잘 작동하도록 Tailwind 유틸리티 클래스(예: `hidden`, `block`, `md:hidden` 등)나 추가 JS 로직을 적용한다.
7. Figma 대비 검증: 구현된 UI를 Figma 디자인 시안과 대조하여 픽셀 단위로 확인하고, 색상·폰트·간격 등이 디자인 가이드와 일치하는지 검수한다. 필요한 경우 Tailwind 설정(theme 등을 조정)이나 CSS 수정으로 디자인 픽셀 퍼펙트를 달성한다.
8. 최종 점검 및 배포: 모든 모듈의 메뉴 네비게이션이 정상 동작하고, 다양한 브라우저 크기에서 레이아웃이 깨지지 않는 것을 확인한다. 코드 정리 및 주석을 추가한 후 변경 사항을 문서화하고 배포한다.
```
