ChatGPT

React 간편결제 파트너 콘솔 UI/UX 통합 명세

1. 디자인 토큰 사양 (Design Tokens)

왜: 디자인 시스템의 토큰화로 일관성과 다크모드 등 테마 지원을 확보합니다 [1] [2]. 글로벌 토큰(색상, 타이포 등)으로 스타일을 제어하면 테마 전환과 브랜드 확장이 용이합니다 [3]. Atlassian 등은 8px 단위를 기반으로 토큰 스케일을 정의해 시각적 일관성을 유지합니다 [4].

무엇: - **컬러 팔레트:** 브랜드 기본색(Primary), 중립색(Neutral), 상태색(정보/성공/경고/위험) 등을 **라이트/다크 모드** 별로 별도 값 매핑합니다 [5]. 예컨대 `color.bg.surface` 토큰은 Light에서 `#FFFFFF`, Dark에서 `#1F1F1F`로 정의하여 동일한 의미의 색상이 테마에 따라 다르게 표시됩니다 [2]. 상태색은 **정보(Information)**, **성공(Success)**, **경고(Warning)**, **위험(Danger)** 등의 의미 기반으로 정의하고 Hover/Pressed 같은 상호작용 변형 토큰을 제공합니다 [6] [7]. 중립색의 경우 밝은 테마/어두운 테마 각각에 최적화된 중립계열 램프를 사용해 충분한 대비를 확보합니다 [8]. (예: Light 중립 `gray500=#6B7280`, Dark 중립 `gray500=#9CA3AF` 등)

- **타이포그래피:** 기본 폰트는 시스템 폰트 또는 범용 웹폰트(예: Noto Sans KR 등)를 사용합니다. **폰트 패밀리와 가중치**(Regular 400, Medium 500, Bold 700 등)를 명시하고, **계층별 폰트 크기/라인높이** 스케일을 정의합니다. 예컨대 본문 기본 14px (line-height ~20px), 작은 텍스트 12px, 제목 Large 18px 등 1.2×~1.5× 배수로 크기를 산정합니다. 각 텍스트 스타일은 크기에 따라 상하 패딩이나 letter-spacing을 약간 조정해 가독성을 최적화합니다 [9]. (헤드라인은 letter-spacing 좁게, 대문자만 쓰는 경우 약간 넓게 등)

- **공간/레이아웃: 4px/8px Grid**를 채택하여 여백과 크기를 배수 단위로 맞춥니다 [10]. 기본 간격 단위는 8px이며, 필요 시 4px의 절반 단위를 보조로 사용합니다. 디자인 토큰으로 `space.100=8px`, `space.200=16px`, `space.300=24px` … 식으로 정의하고 컴포넌트 간 패딩/마진에 적용합니다 [11]. 레이아웃 그리드는 12열 체계를 사용하며, 반응형으로 열과 거터가 재계산됩니다 [12]. 예를 들어 1024px 이상에서는 12컬럼 및 거터 16px, 마진 32px를 기본으로 하고, 768px 이하 모바일에선 6컬럼/거터 12px로 단순화합니다 [12]. 주요 **반응형 구간**은 `<768px` (모바일), `≥768px` (태블릿), `≥1024px` (소형 데스크톱), `≥1280px` (기본 데스크톱), `≥1440px` (와이드 데스크톱)로 구분하여 레이아웃 전환점으로 삼습니다. 각 구간에서 사이드바 표시/숨김, 컨테이너 최대너비 등을 조절합니다. (예: 1280px 이상부터 사이드바 고정 전개, 1024px 미만에서는 사이드바 오버레이 등)

- **쉐도우/모서리/블러:** 컴포넌트의 그림자 깊이와 모서리 곡률을 토큰으로 관리합니다. 예컨대 `radius.sm=4px`, `radius.md=8px`, `radius.lg=12px`로 모달이나 카드 등에 적용하고, `shadow.sm`, `shadow.md` 등을 Elevation 단계별로 정의합니다 (Material Design의 elevation 대응). Light 모드에서는 투명한 검정 그림자(`rgba(0,0,0,0.2)`)를, 다크 모드에선 투명한 흰색 그림자를 써서 어두운 배경에서도 보이도록 합니다. 모달의 경우 배경 블러 토큰(`backdrop.blur`)과 오버레이 투명도(`overlay.opacity`)를 정의하여 모달 뒤 콘텐츠를 흐리게 처리하되 WCAG 2.2를 만족하도록 대비를 유지합니다.

- **대비 기준: 텍스트 및 아이콘 대비**는 WCAG 2.2 AA 기준 이상으로 합니다 (일반 텍스트 4.5:1, 큰 텍스트 및 아이콘 3:1 이상) [13]. 예외적으로 **비활성 상태** 요소는 중요 정보가 아니므로 이 대비 기준에 못 미쳐도 됩니다 [14]. (예: 비활성 버튼은 배경과의 명도 대비가 낮게 설정하여 비활성 상태임을 표현) 또한 **경고 배너 등** 강한 배경색 위의 텍스트는 가독성을 위해 inverse 토큰(예: `color.text.warning.inverse`)을 사용하여 대비를 높입니다 [15].

어떻게: 디자인 토큰은 JSON 등의 포맷으로 정의됩니다. 예시:

```
{
 "color": {
  "bg": {
   "surface": { "light": "#FFFFFF", "dark": "#1B1B1D" },
   "surface.secondary": { "light": "#F7F7F7", "dark": "#2A2A2D" }
  },
  "text": {
   "default": { "light": "#172B4D", "dark": "#F4F5F7" },
   "muted": { "light": "#6B7280", "dark": "#9CA3AF" },
   "success": { "light": "#00875A", "dark": "#57D9A3" },
   "danger": { "light": "#AE2E24", "dark": "#F5A099" }
  },
  "state": {
   "info.bg": { "light": "#EAF4FF", "dark": "#24364E" },
   "info.text": { "light": "#1D7AFC", "dark": "#69C" }
  }
 },
 "radius": {
  "sm": "4px", "md": "8px", "lg": "12px"
 },
 "shadow": {
  "sm": "0 1px 2px rgba(0,0,0,0.08)",
  "md": "0 2px 4px rgba(0,0,0,0.10)"
 },
 "spacing": {
  "space.50": "4px",
  "space.100": "8px",
  "space.200": "16px",
  "space.300": "24px"
 }
}
```

위 예시에서 `color.text.muted.light`와 `.dark` 값을 분기해 Light/Dark 테마별 값을 지정한 것처럼, 모든 색상 토큰을 테마에 따라 분리 제공합니다 [16]. 디자인 시스템에서 이러한 토큰을 사용하면, 테마 전환 시 별도 코드 변경 없이 토큰 값만 교체되어 전체 UI의 색상과 스타일이 일관되게 바뀝니다 [1].

2. 레이아웃 & 내비게이션 명세

왜: B2B 콘솔은 복잡한 정보를 다루므로, 사용자가 효율적으로 탐색할 **전역 레이아웃 구조**가 중요합니다. 일관된 레이아웃은 사용자가 기능 위치를 예측하게 해 학습을 돕고, 다양한 해상도에서도 안정적인 사용성을 제공합니다. 또한 다크 모드 지원을 위해 레이아웃 배경 및 색상 조합을 조정해야 합니다 [5].

무엇: - **전역 레이아웃 구조:** 상단에는 **Topbar 헤더**(서비스 로고, 현재 프로젝트/환경 선택, 사용자 프로필 메뉴, 글로벌 검색/알림 아이콘 등), 왼쪽에는 **사이드바 내비게이션**이 배치됩니다. 사이드바에는 주요 메뉴(예: 대시보드, 거래내역, 정산, 설정 등)가 아이콘+레이블 형태로 계층적으로 표시되며, **축소(collapsed)** 모드에서는 폭 48~64px 아이콘만 보여주는 형태를 지원합니다 [17]. 기본 확장 시 사이드바 폭은 약 **240~280px**로 설정하여 내용이 잘려보이지 않도록 합니다 [17]. 사이드바 축소 시에는 **툴팁**을 제공하여 아이콘 의미를 표시해 접근성을 보완합니다 [18]. 오른쪽에는 상황에

따라 표시되는 **Right Utility 패널**을 둡니다 (필터 패널이나 도움말/설정 패널 등). 이 패널은 필요 시 슬라이드오버 방식으로 나타나며, 디폴트는 숨겨져 있습니다.

- **헤더 & 브레드크럼:** Topbar 아래 콘텐츠 영역 상단에 **페이지 헤더**를 두어 현재 페이지 제목과 작업 컨텍스트를 명확히 합니다. 여기에 **Breadcrumbs**를 배치해 사용자의 위치를 나타내고 상위 메뉴로 이동을 돕습니다 [19]. (예: 거래 관리 > 정산 > 상세) 페이지 헤더 우측에는 해당 페이지의 주요 액션 버튼들을 놓고, 보조정보(업데이트 일시, 필터버튼 등)를 함께 표시합니다.

- **글로벌 검색 & 공지:** Topbar에 **전역 검색창**(돋보기 아이콘)과 **공지/알림 배너 트리거**를 배치합니다. 공지사항 배너는 Topbar 아래 global 영역에 표시되며, 중요 업데이트나 시스템 점검 알림을 **info/banner 컴포넌트**로 나타냅니다. 배너는 닫기 X 버튼과 `role="alert"` 또는 라이브영역 속성으로 구현하여 시각적/스크린리더 모두 주목할 수 있게 합니다.

- **반응형 레이아웃 규칙:** 레이아웃은 **유동적 + 고정폭 혼합**을 채택합니다. 큰 화면(데스크톱)에서는 컨텐츠 최대폭을 적절히 제한하여 가독성을 높이고, 작은 화면(모바일)에서는 전폭을 활용합니다. 주요 Breakpoint는 다음과 같습니다:

- **≥1440px:** 와이드 데스크톱 - 콘텐츠 최대폭 1280px 정도로 중앙정렬. Left Sidebar 항상 펼침(240px 고정폭), Right 패널은 여유공간에 표시 가능.
- **≥1280px:** 일반 데스크톱 - 콘텐츠 최대 1152px 정도. Left Sidebar 펼침(240px). Right 패널 토글 표시.
- **≥1024px:** 태블릿 가로/작은 데스크톱 - Left Sidebar 축소(collapsible) 가능: 아이콘만 표시하거나, 햄버거 메뉴로 숨김. 화면 공간에 따라 콘텐츠 12컬럼 그리드 사용 [12].
- **≥768px:** 태블릿 세로/패블릿 - Left Sidebar 기본 숨김(오프캔버스 메뉴 버튼 제공). 상단 Topbar 고정. 콘텐츠 6~8컬럼 레이아웃.
- **<768px:** 모바일 - Topbar만 상단 고정, 햄버거 버튼으로 전체 메뉴 드로어 오픈. 콘텐츠는 단일 열 흐름.

Atlassian 디자인 시스템처럼, 6가지 breakpoint와 12컬럼 그리드를 조합하여 유연한 레이아웃을 구현합니다 [20]. 작은 뷰포트에서 사이드바는 Off-canvas로 동작하고, `Drawer` 컴포넌트로 구현해 스크린 리더에 접근되지 않도록 숨겼다가 활성화시킵니다.

- **사이드바 동작:** 사이드바는 **아코디언** 형태로 2단계 메뉴를 표시하고, 현재 페이지에 해당하는 항목을 강조(활성 하이라이트)합니다. Collapse된 사이드바에서는 현재 선택 메뉴에 대해 아이콘 강조 및 떠있는 툴팁 레이블을 제공합니다. 사이드바 하단에는 **계정 전환** 드롭다운(테넌트나 가맹점 계정 스위처)을 배치해 여러 계정을 다루는 사용자의 편의를 높입니다 [21]. (예: Stripe 등에서 계정 전환 기능) Collapse 상태에서도 이 전환 메뉴는 식별 가능한 아이콘(예: 회사 로고)으로 남기고 클릭 시 풀사이드 메뉴로 확장합니다.

- **다크모드 전환:** 다크모드는 전체 UI 명도/채도를 조정한 별도 테마로 제공합니다. **명도 조정 규칙:** 배경색은 기본적으로 Dark Theme 기준 `#121212` 계열로 설정하여 명암 대비를 확보하고 [22], 텍스트와 아이콘은 밝은 색(#FFF, #CCC 등)으로 설정합니다. **상태색 보정:** 어두운 배경에서 원본 브랜드색이 충분히 보이지 않을 경우, Material Design 권고처럼 동일한 투명 오버레이를 적용하거나 색상을 밝게 조정합니다 [23]. 예를 들어 Light 모드의 Primary Blue(예: #0052CC)는 Dark 모드에선 조금 더 밝은 Blue (#4C9AFF)로 사용하는 식입니다. 또한 다크모드에서는 강조효과(hover 등)를 나타낼 때 **흰색 8%** 오파시티 오버레이를 사용하고 [24], Light 모드에선 검정 8% 오버레이를 사용해 일관된 contrast 변화를 줍니다. 다크모드 토글은 사용자 설정에 따라 `<html data-theme="dark">` 등을 토글하며, OS `prefers-color-scheme` 미디어쿼리에 따라 초기값을 설정합니다 [25]. 테마 전환 시 번쩍임을 막기 위해 CSS Variables와 `color-scheme` 메타를 활용하고, 전환 애니메이션(페이드)으로 부드럽게 처리합니다.

3. 핵심 컴포넌트 스펙 시트

각 UI 컴포넌트의 크기, 상태, 상호작용, 접근성 요구사항을 정리합니다. (치수 단위 px는 8px 그리드 기준)

버튼 (Button)

- **크기 (Size):** `xs` / `sm` / `md` / `lg` 네 가지로 정의합니다. 터치 타깃은 모든 경우 최소 **40px 이상** 확보하며 [26], `lg`≈48px 높이, `md`≈40px, `sm`≈32px, `xs`≈24px로 합니다. (참고: Ant Design은 small≈24px, default≈32px, large≈40px를 사용 [27]) `xs`는 주로 아이콘-only 버튼이나 테이블 내 액션에 쓰이고, `lg`는 모달 확인 버튼처럼 중요 액션에 사용합니다. 버튼 간 내부 패딩은 가로 16px(아이콘 버튼은 8px), 세로 패딩은 사이즈별 다르게(예: md 버튼 8px 상하 패딩) 적용하여 총 높이가 위 규격에 맞게 합니다.

- **스타일 & 상태:**

- **기본(Base):** Primary (브랜드색 배경 흰색 글자), Secondary (중립 배경 짙은 글자), Text (투명 배경 밑줄 없음 링크 스타일) 등 유형별로 색상 토큰 적용. **폰트 크기:** 기본 14px (sm=14px, md=14px, lg=16px, xs=12px) [28].
- **호버(Hover):** 마우스오버 시 배경 명도를 8~12% 조정하거나 투명오버레이를 입혀 강조합니다 [24]. Primary의 경우 약간 더 진한 색 또는 어두운 배경에선 opacity 0.08의 흰색 레이어를 추가 [24]. Secondary는 경계선 색상을 진하게 변경합니다.
- **액티브(Pressed):** 클릭시 배경을 더욱 짙게 변화시키고, 눌림 효과를 위해 약간 아래쪽으로 그림자를 옮기거나 없앱니다. (예: AntD 기본버튼 active 상태에서 border-color를 primary색으로 변경 [29])
- **비활성(Disabled):** 배경은 중립계열으로 희미하게 (예: #F5F5F5), 텍스트는 `color.text.disabled` 토큰(예: #A0A0A0)으로 표시합니다 [30]. 커서는 기본/hover 효과 없이 `not-allowed`로 처리하고, 포커스 불가 (`tabindex="-1"` 적용). 비활성 상태는 4.5:1 대비를 면제하여 연한 표시를 허용합니다 [14].

- **로딩(Loading):** 버튼 안에 `<Spinner>` 로딩 인디케이터를 배치하고, 콘텐츠는 aria-hidden 처리합니다. 사용자에게는 `aria-live="polite"` 영역에 "로딩 중" 등 안내를 합니다. 로딩 상태에서는 버튼 width가 바뀌지 않도록 미리 공간을 확보해 둡니다.

- **아이콘 버튼:** 아이콘만 있는 버튼의 경우 접근성을 위해 `aria-label`을 필수로 넣어 목적을 설명해야 합니다 [31]. (예: 돋보기 버튼 `aria-label="검색"`) 아이콘과 텍스트가 함께 있는 경우 아이콘은 장식(`aria-hidden="true"`)으로 처리하고 텍스트만 스크린리더가 읽도록 합니다. 아이콘과 텍스트 사이 간격은 8px (`iconGap` 토큰)으로 유지합니다 [32]. 아이콘-only 버튼은 기본 원형 버튼 형태로 40px(또는 sm일 경우 32px) 정사각형 영역을 갖추어 터치 영역을 확보합니다 [33].

- **포커스 표시:** 키보드 포커스시 **뚜렷한 포커스 링**이 나타나야 합니다. 기본으로 **아웃라인 2px** (또는 3px) 굵기의 테두리를 지정된 포커스 컬러(예: 파란색 #2684FF)로 그립니다 [34] [35]. Focus-visible만 표시하도록 CSS `:focus-visible` 사용하고, hover 상태와 중복될 경우 최전면에 표시합니다. (Atlassian은 포커스시 `focus ring` 컴포넌트로 1px 실선+1px 투명 outline 이중처리를 권장 [36]).

접근성: - 역할: `<button>` 태그 사용 (혹은 `role="button"`), 키보드 `Enter` / `Space`로 활성화 가능해야 함 [37]. - 아이콘 버튼은 **보이지 않는 레이블**을 포함 (시각적으로는 `visually-hidden` 클래스로 숨김)하여 스크린리더에 용도를 전달 [38]. - 상태 변화는 `aria-pressed` (토글버튼 경우) 등으로 AT에게 알립니다 [39] [40]. (예: 패스워드 보이기 버튼은 `aria-pressed` 토글로 현재 상태를 표시 [41] [38]) - 비활성 시 `disabled` 속성 및 `aria-disabled="true"` 지정하고 포커스 제외. - 긴 텍스트를 가진 버튼은 폭을 유동적으로 조정하되 최소 너비(예: 80px)토큰으로 정의하여 일관되게 합니다.

버튼 종류	예시	크기 (높이)	설명
Primary	**확인**	md (40px)	주요 액션, 브랜드색 배경
Secondary	취소	md (40px)	보조 액션, 중립 배경 테두리

버튼 종류	예시	크기 (높이)	설명
Icon-only	🗑 (휴지통 아이콘)	sm (32px)	아이콘만, aria-label="삭제" 필요
Disabled	**확인** (비활성)	md (40px)	비활성 표시, 회색톤, 클릭불가

입력 필드 (TextField, Select, etc.)

- **크기:** 한 줄 짜리 Input의 높이는 버튼과 조화를 맞춰 `md≈40px` (large=48px, small=32px) 정도로 합니다 [27]. 좌우 패딩은 8px~12px, Placeholder 존재 시 이 공간을 감안합니다. Label은 폰트 14px 볼드, Input 텍스트는 14px 보통. Select, Combobox 등도 동일 높이 규격을 따릅니다.

- **레이블 & 도움말:** 모든 입력에는 상단에 레이블(`<label>`)을 명시해 맥락을 줍니다. 레이블 옆에 `(필수)` 표시나 `*`를 붙여 필수 여부를 나타내고, 시각적으로 구분합니다. 레이블은 포커스시 해당 입력에 포커스를 전달하도록 `for`/`id`를 연결합니다. **헬퍼텍스트**(placeholder나 별도 안내 문구)는 짧게 Placeholder로 제공하거나, 하단에 작은 회색글씨로 영구 표시합니다 (예: “예: ”).

- **즉시 검증 & 오류 표시:** 사용자가 입력을 종료하거나 포커스를 벗어날 때, 검증을 수행합니다. 잘못된 값이면 **오류 메시지**를 필드 아래에 붉은색 작은 글씨로 표시합니다. 이때 기존의 도움말 문구가 있다면 오류 문구로 **대체**하여 한 번에 하나의 메시지만 보이도록 합니다 [42]. (형식 가이드 → 오류 메시지 교체) 오류 메시지 앞에는 오류 아이콘(⚠)을 함께 표기하고 `aria-live="polite"` 속성으로 화면낭독기에 읽히게 합니다. 또한 입력 필드의 테두리를 `color.border.danger` 토큰(빨강 계열)으로 강조하고, 잘못된 필드에 `aria-invalid="true"`를 추가합니다. 유효성 즉시 검증은 가능한 실시간(입력하면서)보다는 **포커스 아웃** 또는 **제출 시점**에 수행하여 사용성을 저해하지 않도록 합니다 [42]. 단, 명백히 틀린 형식(예: 이메일에 @ 없음)은 입력 중에도 동적으로 클래스 표시하여 힌트를 줄 수 있습니다.

- **제약 조건 표시:** 글자 수 제한, 입력 형식 등 제약이 있으면 이를 **placeholder**나 **헬퍼 텍스트**로 명시합니다. (예: “최대 20자”, “하이픈 없이 숫자만 입력”). 필요한 경우 입력 우측에 **카운터**를 배치해 (`<span>12/20</span>`) 현재 글자수를 표시합니다. 0/초과 등 임계상황 되면 색상을 경고색으로 바꿉니다.

- **복사 버튼 & 마스킹 토글:** API Key나 비밀번호처럼 민감한 입력값은 기본 **마스킹**(`type="password"`) 처리합니다. 우측에 **“보기/숨기기” 토글 아이콘**(눈 모양)을 배치하여 클릭 시 평문 표시(`type="text"`)로 전환합니다. 이 토글 버튼은 **토글 상태**를 `aria-pressed`로 표명하고, `aria-controls`로 대상 input을 지정합니다 [41] [38]. (예: aria-pressed="true"면 “현재 보이는 상태”) 또한 해당 버튼에 `aria-label="비밀번호 표시"` / `"비밀번호 숨기기"` 등 상태에 따른 라벨을 동적으로 변경하거나, 시각적으로는 아이콘 변화 + `aria-pressed`로만 표현하고 스크린리더엔 `aria-label="비밀번호 표시 (선택됨)"` 식으로 처리합니다 [39] [40]. **복사 버튼:** 긴 키 값 옆에는 Copy 아이콘 버튼을 두어 클릭 시 클립보드에 복사합니다. 이 버튼도 `aria-label="API 키 복사"`를 갖게 하고, 성공 시 툴팁이나 토스트로 "복사 완료" 피드백을 제공합니다. 민감정보 노출 이벤트는 감사로그에 기록하며, 복사 버튼 클릭 시에도 사용자 ID, 시각 등을 로깅합니다.

- **상태 스타일:**

- **포커스:** 입력 포커스되면 아웃라인 또는 하단 border를 **브랜드색 굵은 선**으로 표시합니다. 또한 Placeholer 텍스트는 투명도 낮춰주고 (또는 숨김) 실제 입력값이 잘 보이도록 합니다.
- **유효(Valid):** (선택사항) 성공적으로 검증된 경우 테두리를 초록색으로 표시하거나 아이콘(✔)을 우측에 띄워줄 수 있습니다. 하지만 과도한 장식은 피하고, 오류 상황만 명확히 표시하는 것을 기본으로 합니다.
- **비활성/읽기전용:** 배경을 살짝 회색(#F0F0F0)으로, 커서는 기본 화살포인터로 바꾸고 편집 불가를 나타냅니다. `aria-readonly="true"` 또는 `disabled` 속성 사용 [43].

접근성: - 레이블은 `<label for="inputId">`로 연결하고, placeholder만으로 의미를 전달하지 않습니다(placeholder는 보조 힌트로만 사용) [44]. - 오류 메시지는 해당 input과 `aria-describedby`로 연결하여 SR이 읽을 수 있게 합니다. (ex: `<div id="emailError" role="alert">잘못된 이메일 형식입니다.</div>` 그리고 input에 `aria-describedby="emailError"`) - 필수 입력은 레이블에 반드시 표시하고, `aria-required="true"`를 추가합니다. - 그룹으로 묶인 입력 (예: 전화번호 국번 등)에는 `<fieldset><legend>` 구조로 그룹 레이블을 제공하고, 안의 각각 input에는 개별 레이블 혹은 `aria-label`로 서술합니다. - Select/Combobox에는 현재 선택값을 시각적으로 보이게 하고, 키보드로 `Alt+↓` 등으로 목록 열기 가능하도록 JS로 제어합니다. 옵션 리스트는 `<ul role="listbox">`와 `<li role="option">`으로 구성하거나 native `<select>`를 사용합니다. - 날짜 선택(DateRangePicker)은 `<input type="text">`로 날짜를 입력하거나, 달력 아이콘 버튼을 통해 팝업 캘린더(`<div role="dialog">` 내에 `role="grid"`)를 띄웁니다. 달력에는 `aria-label`로 월/년도, 오늘 표시 등을 제공하고 `aria-selected`로 선택일을 나타냅니다.

카드 (Card)

- **유형:** 정보 카드(텍스트+아이콘), 지표 카드(숫자 통계), 경고 카드(주의를 환기하는 배경색 박스) 등으로 사용됩니다. 모든 카드는 통일된 스타일 규칙을 따릅니다:
- **레이아웃:** 카드 구성은 **Header / Body / Footer** 3부분으로 표준화합니다 [45]. Header에는 카드 제목과 요약 액션이, Body에는 본문 내용, Footer에는 하단 부가정보나 액션 버튼들이 들어갑니다. 이러한 구조를 일관되게 유지하면 사용자가 카드별 위치에서 기대하는 요소(제목, 버튼)를 쉽게 찾습니다 [46].
- **간격:** 카드 내부 패딩은 상하 16px, 좌우 20px (8px grid 준수)로 여유 있게 합니다. Header와 Body 사이, Body와 Footer 사이에는 12~16px의 공간을 둬 시각적 구분을 줍니다. 여러 섹션이 있는 Card Body에서는 **구획 간 24px** (`space-200`) 정도 간격을 주고 구분선은 지양하여 여백으로 그룹짓습니다 [47].
- **Header: 카드 제목**은 Bold 14~16px로 나타내고 [48], 카드 전체 내용을 대표하는 적당한 명사를 사용합니다. (예: "정산 계좌", "월간 거래현황") 카드 우측 상단에는 **Header actions**로서 편집 또는 바로가기 등의 아이콘 버튼을 둘 수 있습니다 [49]. 이 경우 아이콘-only 버튼에는 툴팁을 제공해 무슨 동작인지 설명합니다 [50]. Header action은 카드 전체를 대상으로 하는 동작만 배치하고, 개별 항목에 대한 액션은 Body 내부 해당 항목 옆에 배치해야 합니다 [51].
- **Body:** 주요 콘텐츠 영역입니다. 여기에는 텍스트, 표, 그래프, 리스트 등 다양한 요소가 들어갈 수 있습니다. Body를 **Section**으로 나눌 때 각 Section 별로 소제목(예: `<h6>`)을 붙일 수 있고, 그렇지 않을 땐 Card 제목만으로 내용을 파악할 수 있게 작성합니다 [52] [53]. 예: "결제수단 현황" 카드 안에 "신용카드 50%, 계좌이체 30%..." 등 섹션별 통계를 표기하는 경우 섹션 제목 "신용카드" 등을 두지만, 한 종류만 있을 땐 중복 제목 생략.

- **Footer:** 부가 링크나 **CTA(Call-to-Action)** 버튼을 배치합니다. 예를 들어 "전체 보기" 링크, "자세히" 아이콘 등이 Footer 우측 정렬로 나타날 수 있습니다. Footer가 있을 때는 상단에 1px 연한 경계선(#EFEFEF)이나 16px 패딩으로 Body와 분리합니다. 위험 작업용 카드(ex: 연결 해지 경고)에서는 Footer에 취소/확인 버튼을 두어 사용자가 바로 조치하게 합니다.

- **모서리/그림자:** 카드 컨테이너는 `radius.md` (8px) 모서리 둥글기를 적용하고, 기본 그림자 `shadow.sm` (아주 옅은 그림자)로 부양감을 줍니다. 다크 모드에서는 그림자 대신 약간 밝은 테두리(예: 투명한 하얀 경계 1px)를 사용해 카드 영역이 배경과 구분되게 합니다.

- **경고/정보 카드:** 특별한 맥락의 카드 (예: 서류 미제출 경고)는 배경색을 상태 색의 아주 옅은 톤으로 설정합니다 (정보=파랑계열 #EAF4FF, 경고=노랑계열 #FFF7D6 등 [54]). 아이콘(i, ⚠)과 함께 메시지를 넣고, **닫기 버튼(X)**을 우측 상단에 제공하여 카드 형태의 알림을 Dismiss할 수 있게 합니다. 이 버튼은 `aria-label="닫기"`로 SR에 제공됩니다.

접근성: - 카드의 제목은 `<h3>` 등으로 마크업하여 문서 내 구조를 돕습니다. 여러 카드가 연달아 있으면 제목 레벨을 동일하게 유지해 SR 사용자가 헤딩 목록으로 쉽게 파악하도록 합니다. - 카드 내에 **리스트나 테이블**이 있으면 해당 컴포넌트 규칙(표의 scope 등)에 따릅니다. - Header의 아이콘버튼은 툴팁을 통해 보조 라벨을 제공하므로, `<button`

`aria-label="편집">✎</button>` + `<div role="tooltip">편집</div>` 식으로 구현합니다 [49] [50] . - 중요도가 높은 카드는 `role="region"` 과 `aria-label` 을 줄 수도 있습니다 (예: "월간 요약 섹션"). - 키보드: 카드 자체는 인터랙티브 요소가 아니므로 Tab포커스를 갖지 않고, 내부의 버튼/링크만 포커스 가능합니다. 그러나 **카드 간 keyboard navigation**이 필요하면 각 카드를 `tabindex="0"` 주어 Up/Down으로 카드 블럭 간 이동시키는 ARIA 그리드 혹은 listbox 패턴을 고려할 수 있습니다.

모달 / 슬라이드오버 (Modal Dialog & Drawer)

- **등장 방식:** 전역 모달은 `<div role="dialog" aria-modal="true">` 로 구현하며, 페이지 위에 오버레이로 등장합니다 [55] . 슬라이드오버 (우측 Drawer 패널)는 `<aside>` 또는 `<div role="dialog">` 로 간주하고 aria-modal은 false 처리(메인과 동시에 접근 안되게 백드롭 처리)하거나, 상황에 따라 modal로 취급합니다.

- **크기:** 모달 너비는 내용에 따라 **small (~400px)**, **medium (~600px)**, **large (~800px)** 세 종류를 둡니다. 작은 확인 대화창(예: "정말 삭제할까요?")은 small, 폼 입력이 많은 경우 medium~large를 사용합니다. 세로 길이는 최대 90vh로 제한해 화면을 넘지 않도록 하고, 내용이 길면 내부 스크롤을 적용합니다.

- **Header/Body/Footer 구조:**

- **Header:** 모달 상단에 **제목**(굵은 16px)을 표시하여 맥락을 명확히 합니다. 예: "정산 계좌 변경". 우측 상단에 **닫기(X) 버튼**을 배치하며, 이 버튼은 `aria-label="닫기"` 를 가져야 합니다.
- **Body:** 주요 내용 영역. 텍스트 설명, 폼 요소 등이 위치. 충분한 패딩(20px)과 적절한 line-height로 가독성을 확보합니다.

- **Footer:** 하단에 액션 버튼들을 배치합니다. 기본적으로 오른쪽 정렬로 Primary 액션(확인) 버튼, Secondary(취소) 버튼을 두며, 2개를 초과하지 않도록 합니다 [56] . (모달은 집중된 작업을 위한 것이므로 버튼이 너무 많으면 안 됨) 필요한 경우 좌측에 부가링크 (예: "자세한 도움말") 등을 둡니다.

- **동작 규칙:**

- **열기/닫기:** 모달 열릴 때 배경 콘텐츠는 **스크롤 락**을 걸어(`overflow: hidden` on body) 사용자 스크롤이 모달 내로 제한됩니다 [55] . 닫힐 때 body scroll을 복원합니다.
- **닫기 조건:** 사용자가 **배경을 클릭**하거나 **ESC 키를 누르면** 모달을 닫습니다 [56] [57] . (단, 중요한 확인이 필요한 경우 이러한 닫기를 막고 반드시 버튼 클릭을 유도할 수도 있음)
- **포커스 트랩:** 모달이 떠있는 동안 Tab키 이동은 모달 내부에 한정됩니다 [57] . 첫 포커스는 기본적으로 모달의 첫 인터랙티브 요소 (없으면 닫기 버튼)에 맞추고, Shift+Tab으로 앞 요소가 없으면 마지막으로 루프백됩니다 [57] . 모달 닫힐 때는 **원래 트리거** 요소로 포커스가 돌아갑니다 [58] .

- **위험 액션 이중 확인:** API 키 삭제나 계좌해지처럼 위험한 액션의 모달에서는, **2단계 확인**을 적용합니다. 예: "'삭제'를 입력하여 확인:" 입력필드와 확인버튼을 두어 오동작을 방지합니다. 또한 이러한 파괴적 행동 모달의 Primary 버튼은 **Destructive style**(빨간색)로 표시합니다. Secondary는 "취소"로 명확히 합니다. 이때 사용자가 확인 입력을 마치기 전까지 Primary 버튼을 disabled로 두었다가, 정확히 일치하면 활성화합니다.

- **애니메이션:** 모달은 페이드인(+약간 scale in)으로 띄우고, 닫을 때는 반대로. Drawer는 오른쪽에서 슬라이드 인/아웃. 애니메이션은 200~300ms로 부드럽게 처리합니다 (easing: ease-out). 애니메이션 중에도 ESC로 즉시 닫기 동작은 받습니다.

접근성: - `role="dialog"` 와 `aria-modal="true"` 를 모달 최상단에 명시하여 보조기기에 모달 컨텍스트임을 알립니다 [59] [60] . - 모달 제목 요소에 `id` 를 부여하고 다이얼로그에 `aria-labelledby` 로 연결해 SR이 제목을 읽게 합니다 [61] . 본문이 길면 요약을 `aria-describedby` 로 연결하거나, 첫 포커스 요소를 설명문단에 두어 읽도

록 합니다 [62] [63] . - **포커스트랩** 구현은 JavaScript로 관리: 모달 열릴 때 `document.activeElement` 저장 후 첫 포커스로 이동, 키보드 이벤트로 Tab 순환 제어, 닫힐 때 저장한 위치로 포커스 복귀 [58] . - 모달 열리면 배경은 inert 처리: HTML5엔 inert 속성 활용하거나, 모달 외 요소에 `aria-hidden="true"` 적용. - 스크린리더는 aria-modal로 배경 읽기를 막지만, 모든 AT가 완벽히 지원하지 않을 수 있어 위처럼 중복 조치합니다 [64] . - 닫기 버튼은 키보드 접근 가능해야 하며, 모달 안에 반드시 하나 존재하도록 합니다 [65] . (닫기 X 또는 "취소" 버튼) - **라이브영역:** 모달이 뜰 때 `<div role="alertdialog">`로 하면 SR이 즉각 안내합니다. 혹은 `role="dialog"` +JavaScript로 "대화상자가 열렸습니다" 메시지를 aria-live로 읽게 할 수도 있습니다.

테이블 (Data Table)

- **레이아웃 & 기능:** 대량 데이터의 효율 표시를 위해 **가상 스크롤(virtual scroll)**, **페이징**, **정렬/필터링** 등을 지원합니다.
- **가상화:** 수백~수천 행이 넘는 경우 DOM에 필요한 부분만 렌더링하는 **행 가상화**를 적용하여 성능을 유지합니다. TanStack Virtual 등 라이브러리를 활용하여 수천 행도 부드럽게 스크롤 가능하게 합니다 [66] . 스크롤 성능 목표는 60fps입니다.
- **페이징:** 서버사이드 페이지네이션을 기본 적용합니다 (한 페이지 50행 등). 표 하단에 페이지 번호와 앞뒤 화살표로 페이지를 이동하며, 한 번에 1000건 이상 볼 경우 Lazy Load(가상스크롤)로 전환 고려.
- **정렬:** 컬럼 헤더를 클릭하면 그 컬럼으로 정렬합니다. 정렬 상태는 헤더 텍스트 우측에 ↑/↓ 아이콘으로 표시하고 `aria-sort="ascending|descending"`을 적용합니다. Shift+Click으로 다중컬럼 정렬도 가능하게 합니다.
- **필터:** 각 컬럼 헤더나 상단에 필터 도구(UI 칩 또는 드롭다운)를 제공합니다. 일반 텍스트 검색, 날짜 범위 선택, 드롭다운 필터 등을 지원하고, 자주 쓰는 필터 조합은 "즐겨찾는 필터"로 저장/불러오기 가능하게 설계합니다. 필터 적용 중임을 표 상단에 배지 또는 작은 칩으로 보여주고, '필터 지우기' 기능을 제공합니다.

- **컬럼 설정:** 사용자가 **컬럼 보이기/숨기기**를 토글할 수 있는 설정 메뉴를 테이블 우상단에 둡니다. 또 **컬럼 리사이즈**(헤더 경계 드래그) 및 **컬럼 고정**(좌측 고정) 기능도 지원합니다. (대부분 AG Grid 등에서 제공) 컬럼 리사이즈는 최소/최대 폭 제한을 두어 레이아웃 파괴를 막습니다. 고정된 컬럼은 스크롤시 항상 보이게 하며, 고정 구역과 본 테이블 경계에 분리선(shadow) 표시로 구분합니다.

- **내용 표시:**

- **정상 상태:** 셀 내 텍스트는 기본 14px 좌측 정렬(숫자는 우측 정렬). 중간 줄바꿈이 필요한 경우 `<br>` 대신 적절한 너비를 주거나 툴팁으로 전체 내용 표시합니다. 긴 내용(예: 거래 설명)은 말줄임표 처리하고 hover 시 툴팁으로 풀텍스트 제공합니다.
- **상태 뱃지:** 거래 상태 등은 컬러 뱃지로 표시합니다. 예: “승인”은 녹색 배경+흰글씨 뱃지, “보류”는 주황 테두리 뱃지 등 [67] . 이 뱃지들에는 `aria-label="승인 상태"` 등으로 SR 안내를 보강합니다. (Alternatively, `<td><span class="badge badge-success">승인</span></td>`)
- **금액 포맷:** 통화는 로케일별 통화 기호와 천단위 구분을 적용합니다. (예: KRW ₩12,345, USD $12,345.00). Intl API나 CLDR 데이터를 이용하여 통화 포맷과 반올림 자릿수를 맞춥니다 [68] . 음수 금액은 `-₩1,234`처럼 기호로 표시하되, SR에 "마이너스 1,234원" 읽히도록 visually-hidden 문구 추가 가능합니다.
- **민감 정보 마스킹:** 카드번호, 주민번호 등은 기본 `****-****` 형태로 일부만 보여줍니다. 사용자가 **역마스킹** 권한이 있고 해당 셀에 마우스 호버 시, “보기” 아이콘 버튼을 떠서 클릭하면 5초간 평문 표시 후 다시 마스킹합니다. 이때 해당 이벤트(누가 언제 어느 데이터를 열람)를 감사 로그로 남기고, 사용자에게 상단에 배너 또는 토스트로 "민감정보가 일시 노출됩니다" 경고를 표시합니다.

- **오류 툴팁:** 값이 “실패” 등 에러 상태인 경우, 셀에 붉은 아이콘(❗)이나 강조를 주고, 마우스오버/포커스 시 **툴팁**으로 실패 사유를 표시합니다. (예: “출금 실패 - 잔액부족”) 이 툴팁은 `<div role="tooltip">`로 구현하고 내용에 기술적인 용어를 피하며 필요한 경우 링크(“자세히”)를 포함할 수 있습니다.

- **빈 상태/로딩/에러:**

- **빈 상태:** 필터 결과가 없거나 데이터가 아예 없을 경우, 테이블 대신 **Empty state** 일러스트와 메시지를 표시합니다 (예: "거래 내역이 없습니다"). 이는 별도 컴포넌트(`<EmptyState>`)로 구현하고, 필요한 경우 주요 액션(예: "새 거래 추가") 버튼을 함께 제시합니다. SR에게도 해당 메시지를 읽혀야 합니다.
- **로딩:** 데이터 로딩 중에는 테이블 셀 대신 **스켈레톤 로더**(회색 셰이프 애니메이션)로 자리표시를 합니다 [69]. 예: 행 5개 정도 회색 바를 표 형태로 깜박여 로딩 중임을 표현. 또한 `<table aria-busy="true">`로 지정해 SR에 알려줍니다.

- **에러:** 데이터 불러오기 실패 시에는 표 대신 에러 아이콘(⚡)과 "데이터를 불러오지 못했습니다. [재시도]" 메시지를 표시합니다. 재시도 버튼에 `onClick`으로 재호출을 연결합니다. 또한 SR-friendly하게 `role="alert"`로 즉시 읽히게 합니다.

- **대용량 최적화:** 테이블이 화면에 여러 개 있을 경우 가시 영역의 테이블만 렌더하거나, 가상화 외에도 **pagination**을 적극 사용합니다. 또한 열의 수가 많다면 가로 스크롤을 허용하되, 첫번째 중요 열들을 좌고정해 가독성을 유지합니다. 사용자 브라우저가 느릴 경우 (fps <30 감지)에는 한 번에 표시 행수를 자동 줄이는 것도 고려합니다.

접근성 (ARIA 그리드): 표는 시맨틱 `<table>` 태그로 작성하는 것이 가장 이상적이며, `<th>`에 `scope="col"`/`scope="row"`를 지정해 관계를 명시합니다. 그러나 인터랙티브한 기능 (정렬, 셀 포커스 등)이 많다면 ARIA Grid 패턴을 적용할 수 있습니다: - 컨테이너에 `role="grid"`를 주고, 행은 `role="row"`, 셀은 `role="gridcell"`로 합니다 [70]. 칼럼 헤더는 `role="columnheader"`, 행 헤더는 `role="rowheader"`로 표기합니다 [70]. 그리고 `aria-colcount`/`aria-rowcount`로 전체 크기를 알립니다. - 키보드 내비게이션: grid에 포커스되면 **방향키**로 셀 간 이동을 지원합니다 [71] [72]. →/←는 좌우 이동, ↑/↓는 상하 이동입니다. Home/End는 행의 처음/끝, Ctrl+Home/End는 전체 첫 셀/마지막 셀로 이동합니다 [73]. 이때 셀에 포커스가 가면 `aria-selected` 등을 이용해 현재 선택상태를 알립니다. - **행 선택:** 멀티 선택 가능 시 `aria-multiselectable="true"`를 grid에 지정합니다 [74]. 행 선택은 보통 체크박스를 활용하지만, 키보드만으로는 **Shift+Space**로 현재 행 토글 선택 [75], **Ctrl+A**로 전체 선택 등을 지원합니다 [76] [75]. 이런 단축키 안내는 필요 시 툴바의 도움말(?) 아이콘에 설명서 툴팁을 넣습니다. - **포커스 관리:** 셀 내부에 버튼/링크 등 포커스 가능한 요소가 있을 때, **화살키**와 **Tab키** 동작을 분리합니다. 화살키는 셀 간 이동에 쓰이고, 특정 셀에서 `Enter` 또는 `F2` 등을 누르면 "셀 편집 모드"로 전환하여 Tab으로 내부 요소를 탐색하게 하는 패턴을 적용합니다 [77] [78]. 이는 스프레드시트형 상호작용에서 권장되는 방식입니다. 기본 폼이 아닌 그냥 데이터 표일 경우에는 셀 내 포커스 요소가 없으므로 단순 arrow navigation이면 됩니다. - **ARIA 속성:** 정렬된 컬럼 헤더에는 `aria-sort="ascending"`/`"descending"`를 적용하여 SR이 정렬 상태를 파악하게 합니다. 필터링이 적용된 경우 표에 `aria-live="polite"` 영역에 "N건 필터됨" 메시지를 업데이트하여 알려줍니다. - HTML `<table>` 사용 시에는 `summary`나 `<caption>`을 활용해 표 목적을 서술합니다. (예: `<caption>최근 1주일간 거래내역</caption>`)

폼 (온보딩/KYB 프로세스)

- **다중 스텝 (Stepper):** 신규 파트너 온보딩이나 KYB 제출 과정은 **단계별 폼**으로 구성합니다. 상단에 **Step Indicator**를 표시해 현재 단계를 시각화합니다 (예: 1. 정보입력 → 2. 사업자등록증 → 3. 정산계좌 → 4. 완료). 진행상황 바 또는 숫자/레이블 조합으로 단계를 나타냅니다. Stepper는 Linear 모드로 이전 단계를 완료해야 다음 단계로 진행 가능하게 하고 [79], 완료된 단계는 체크표시 또는 회색표기로 표시합니다. 사용자가 중간까지 작성 후 나갔다 돌아올 경우를 대비해 **저장 후 이어하기** 기능을 지원하며, 다시 진입 시 완료된 단계까지는 스킵하거나 읽기전용 검토상태로 보여줍니다.

- **단계별 화면:** 한 단계에는 관련된 입력 필드 그룹들이 있습니다. 예를 들어:

- **개요 정보** – 업체명, 담당자 연락처 등
- **사업자 등록증** – 사업자번호, 법인명, 증빙 파일 업로드
- **정산 계좌** – 계좌번호, 은행명, 예금주

- **제출 확인** – 전체 입력 검토 및 전송 버튼 각 단계 폼 하단에는 “다음으로” / “이전으로” 버튼이 있어 순차 진행합니다 [80]. 마지막 단계에는 "제출" 버튼. 중간 단계에서 **임시저장** 옵션도 하단에 별도 배치합니다.

- **필수/선택 필드:** 필수 필드는 레이블에 `*`를 표시하고, `required` 속성을 부여합니다. 선택 사항은 "(선택)" 표시를 레이블 뒤에 추가합니다. 폼 제출시 누락된 필수값이 있으면 해당 필드로 포커스를 주고, 오류 메시지를 표시합니다 (예: "이 필드는 필수입니다"). 동시에 상단 또는 해당 위치에 "필수항목을 입력해주세요" 배너를 띄워줄 수도 있습니다.

- **실시간 검증:** 가능한 한 **입력 중 즉각 검증**을 실시하여 사용자 오류를 줄입니다. 예: 사업자번호 10자리 다 채우면 바로 형식 체크, 중복 검사 API 호출 등. 단, 너무 잦은 경고는 피하고, 입력 완료 시점에만 표시합니다 [42]. 파일 업로드 시 허용 확장자/용량도 선택 직후 체크하여 문제 시 오류 표시합니다.

- **파일 업로드 UI:** KYB 제출 시 필요 서류(예: 사업자등록증 사본 등)는 **드래그앤드롭 영역**과 "파일 선택" 버튼 둘 다 제공해 업로드합니다. 업로드 컴포넌트는 개별 파일 항목마다 상태를 표시합니다:

- **대기중:** 파일명.jpg (0%) – 업로드 진행바 표시.
- **완료:** 파일명.jpg – 첨부 크기, 썸네일(이미지일 경우) + 아이콘.
- **실패:** 파일명.jpg – **!**아이콘 및 "업로드 실패. 재시도" 링크.

- **삭제:** 각 파일 오른쪽에 X버튼 (aria-label="파일 삭제")로 목록에서 제거 기능. 또한 파일 종류별 아이콘을 표시하고, 허용 형식 아니면 즉시 오류 메시지 ("PDF 또는 JPG만 업로드 가능")를 표시합니다. 용량 초과시 ("최대 5MB")도 마찬가지로 표시합니다. 여러 파일을 한 번에 선택 가능하며, 순서 상관없이 나열합니다.

- **폼 UX 개선:** 입력 도중 페이지 이탈 시 **경고 다이얼로그**("저장되지 않은 변경사항이 있습니다. 정말 나가시겠습니까?")를 띄워 사용자 확인을 받습니다. 임시저장 기능 사용 시 마지막 저장 시각을 표시하여安心감을 줍니다. 또한 진행 중 이미 입력한 값을 바탕으로 다음 단계에 미리 채울 수 있는 부분은 프리필(예: 이전 단계 입력한 이름을 다음 단계 리뷰 화면에 표시) 합니다.

- **검수/예외 상황:** 제출 후 서류 검수 단계에서는 **진행 상태**를 대시보드에 표시합니다. ("심사중", "반려 - 사유: XXXX" 등) 반려 시 사용자는 해당 단계로 이동하여 수정 업로드 할 수 있도록 UI를 제공합니다. 재제출 후에는 상태 갱신. 만약 API 오류 등으로 제출 실패하면, 사용자에게 오류 다이얼로그/배너로 알리고 재시도 안내를 합니다. 가능한 오류 케이스 (네트워크 다운 등)마다 친절한 메시지를 준비합니다.

접근성: - Step Indicator는 `<nav aria-label="온보딩 진행단계">`와 내부 리스트 (`<ol role="tablist">` + `<li role="tab" aria-selected>` 등)로 구현하여 SR이 단계 수와 현재 단계를 인지하게 합니다. 현재 단계 탭에는 `aria-current="step"` 적용도 고려합니다. - 각 단계 폼은 하나의 섹션으로 `<fieldset>`으로 감쌀 수 있고, 단계 제목을 `<legend>`로 두면 SR에 각 단계의 주제 전달됩니다. - 폼 요소들은 앞서 input 접근성 원칙을 따릅니다. 오류 메시지는 `role="alert"`로 실시간 읽히도록 해 사용자가 바로 인지하게 합니다. - 파일 업로드 컴포넌트는 `<input type="file" aria-describedby="지원형식안내">`로 만들고, DnD 영역에도 `role="button"`과 키보드 지원(on Enter키) 제공해 키보드만으로도 파일 선택 대화상자를 띄울 수 있게 합니다. 업로드 진행 상태는 시각적으로뿐만 아니라 `aria-live="polite"` 영역에 텍스트로 업데이트("파일 1 업로드 50% 완료") 합니다. - 폼 완료 후 "제출" 버튼에는 `aria-disabled` 적용 (검증오류 있을 경우) 또는 활성. 제출 성공시 피드백으로 성공 알림을 화면에 보여주고 `role="status"`로 읽히도록 합니다.

알림 (Notifications)

- **토스트 vs 인라인 vs 배너:**
- **토스트(Notification Toast):** 사용자의 작업 피드백이나 일시적 알림은 화면 우측 하단에 토스트로 띄웁니다. (예: "복사 완료", "저장되었습니다") 토스트는 자동으로 3~5초 후 사라지며, 여러 개 중복시 스택으로 표시합니다.

- **인라인 메시지:** 폼 내 특정 섹션에 대한 안내/경고는 해당 위치에 **Inline Message**로 표시합니다 [81] . (예: "비밀번호는 8자 이상이어야 합니다" - 필드 바로 아래 붉은 텍스트)

- **페이지 배너:** 전체 시스템 공지나 높은 심각도의 에러(예: 권한 부족으로 콘텐츠 표시 불가)는 페이지 상단에 **Banner**로 띄웁니다 [56] . 배너는 색상으로 구분 (정보-파랑, 경고-노랑, 위험-빨강)하며 아이콘과 함께 고정된 영역에 노출됩니다.

- **심각도 매핑:** 알림은 **정보(I)**, **성공(S)**, **경고(W)**, **위험(E)** 4단계로 구분합니다. 심각도에 따라 색상과 아이콘을 적용합니다:

- 정보: 파란색 계열, i 아이콘 (예: 업데이트 소식)
- 성공: 녹색 계열, 아이콘 (예: 저장 완료)
- 경고: 노랑/주황 계열, ⚠ 아이콘 (예: 유효기간 임박)

- 위험: 빨강 계열, ⛔ 아이콘 (예: 액션 실패, 권한없음) 배경은 연한 색, 아이콘/텍스트는 어두운색으로 대비 충분히 합니다. (다크모드에서는 반대로 연한 배경+밝은 텍스트)

- **토스트 상세:** 토스트 내용은 짧고 명확하게 작성하며, 필요시 **Undo** 같은 액션 버튼을 포함할 수 있습니다. (예: "삭제했습니다. [실행취소]") 토스트 여러개가 쌓이면 최신것이 위에 오며, 5개 이상 쌓이면 오래된 것부터 즉시 사라집니다. 토스트에는 `role="status"` 를 줘서 SR이 읽도록 합니다 (사용자 조작 결과 알림이므로 polite하게).

- **라이브영역 규칙:**

- 중요 오류(위험 배너)는 `role="alert"` 으로 즉시 읽도록 합니다. (예: "서버와 연결이 끊어졌습니다" 배너)
- 정보성 토스트는 `role="status"` 나 `aria-live="polite"` 로 설정 [81] .

- 상태 업데이트 빈번한 영역 (예: 처리 진행률)은 `aria-live="polite"` 를 써서 적당히 읽히게, `assertive` 는 꼭 필요한 경우만.

- **사용자 조작:** 배너/토스트에 **닫기(X)** 버튼을 제공하여 사용자가 명시적으로 닫을 수 있게 합니다. 이 버튼은 `aria-label="닫기 알림"` 을 가져야 합니다. 키보드로 토스트에 포커스 가면 자동닫힘 타이머는 일시 정지합니다.

접근성: - SR 사용자를 위해, 중요한 시스템 배너는 페이지 진입시 또는 발생시 즉시 알림이 가야 합니다. 그러므로 DOM 상 랜더 순서를 상단에 두거나 `aria-live="assertive"` 로 설정합니다. - 알림 컴포넌트는 화면상 위치와 관계없이 논리적으로는 페이지 끝에 두되, 시각적으로 fixed 포지셔닝 해도 무방합니다. Off-screen으로 SR용 텍스트 (`.visually-hidden[role=status]`)를 두어도 좋습니다. - 다중 알림이 연달아 뜰 경우 SR이 겹쳐 읽지 않게, 한 알림이 끝난 후 다음을 live 영역에 넣는 식으로 관리합니다. - 애니메이션 등장시 SR announce Timing을 위해 약간 지연을 주거나, DOM 추가 순서에 주의합니다. - 색상/아이콘에만 의존하지 않고 텍스트로 내용과 심각도를 함께 전달합니다 ("성공: 변경사항이 저장되었습니다" 등). 아이콘에는 `aria-hidden="true"` 처리하고 SR엔 "성공" 단어를 포함시키는 방식.

차트 (Charts: 선/막대/영역/도넛)

- **차트 구성:** 대시보드용 통계 차트들은 Line, Bar, Area, Donut 형태 등으로 구현합니다.
- **선(Line) 차트:** 시간에 따른 추이를 나타낼 때 사용. 예: 일별 거래금액. X축은 시간 (요일/날짜), Y축은 금액. 선은 2px 굵기로, 주요 포인트에만 마커(dot)를 표시.
- **막대(Bar) 차트:** 항목별 비교. 예: 결제수단별 건수. X축 범주(카드/계좌/포인트…), Y축 수치. 막대 간 간격 4px, 막대 너비는 대응 데이터 개수에 따라 자동.

- **영역(Area) 차트:** 누적량이나 범위 표시. Line 차트와 유사하나, 선 아래 면적을 채워 총량을 보여줌. 투명도 0.2~0.3로 채웁니다.
- **도넛(Donut/Pie) 차트:** 구성 비율 표시. 예: 거래 상태별 비중. 각 조각에 대비색 지정하고, 중앙에 총합 또는 선택된 항목 정보를 표시.
- **축/단위:**
- X/Y **축 라벨**은 명확히 표시하고 단위도 첨언합니다 (예: "금액 (만원)" 이런 식). Y축 눈금은 3~5단계로 자동 생성하되, 인간 친화적으로 (예: 100K 대신 100,000원).
- **단위 표기:** 데이터 값에는 국제화 포맷 적용 (위 통화/숫자 규칙). 천단위 구분, %는 한글 locale 시 “%”붙여쓰기 등 CLDR 양식 준수 [68] .
- **툴팁/범례:** 차트 상호작용으로 특정 데이터의 값을 볼 수 있게 합니다. 마우스 hover 또는 포커스 시 **툴팁**을 띄워 해당 지점의 구체 값(예: "2025-01-03: ₩5,300,000")을 보여줍니다. 범주형 차트(막대 등)는 범례(legend)를 차트 하단에 제공하여 색상과 항목을 대응시킵니다. 범례 항목을 클릭하면 해당 데이터 세트 토글 (보이기/숨기기) 기능도 고려합니다.
- **결측 처리:** 데이터가 없는 날 등의 구간은 0으로 간주하여 선을 끊지 않고 0점으로 그리거나, 선을 점선으로 표시합니다. `NaN` 등은 "데이터 없음"으로 처리. 차트 라이브러리 설정에서 null값 처리를 지원하면 활용합니다.
- **데이터 다운로드:** 차트 우상단 “↓” 아이콘 버튼을 통해 현재 차트를 이미지(PNG) 또는 CSV 데이터로 다운로드 가능하게 합니다. 이미지 저장은 chart.js 등 라이브러리의 toDataURL 기능을 이용하거나 서버에서 생성. CSV 다운로드는 차트 데이터 배열을 문자열로 변환해 다운 트리거. (AG Grid 등 표에서는 CSV Export 기본 제공 [82])
- **반응형:** 좁은 화면에서는 범례를 차트 위에 오버레이하거나, 숨기고 탭으로 바꿀 수 있습니다. 축 라벨 겹칠 경우 자동 회전(45°) 또는 간격 축소. 중요한 수치 위주로 간소화합니다.

접근성: - 차트는 canvas/SVG로 그려지므로 시각장애인을 위해 **대체 텍스트**를 제공합니다. 차트마다 `<div role="region" aria-label="월별 거래 추이 그래프"`와 `aria-describedby="chart1-desc"`를 두고, 숨긴 `<div id="chart1-desc">2023년 1월부터 12월까지 거래금액 추이: 1월 5백만, 2월 7백만, ...</div>`를 넣어 데이터 개요를 전달합니다. 또는 간략히 "3분기 이후 상승 추세" 같은 해석을 넣을 수도 있습니다. - 범례 역시 키보드로 집중 가능하게 `<ul>`목록으로 제공하고, 각 항목에 `tabindex="0"`와 `aria-pressed` (토글 가능 시)를 지정합니다. - 툴팁은 시각적으로만 보이므로, 키보드 사용자를 위해 차트 데이터 점들을 포커스 가능하게 만들어 `focus`시 동일 툴팁 내용을 화면에 별도 표시하거나 SR에게 읽히게 합니다. (예: SVG circle 요소에 `tabindex="0"`, `aria-label="1월: 5백만 원"`) 차트 라이브러리에 따라 aria 속성 지원이 제한적이면, 데이터 표(Table)을 숨겨서 제공하는 방법도 고려합니다. (예: 차트 아래 `<table class="sr-only">`로 동일 데이터를 표 형태 제공) - 색상 구분 외에 **패턴/모양**으로도 구별을 줍니다. (ex: 파란색 실선 vs 초록색 점선, 노란색 채움 vs 빨간색 사선패턴 채움) 이는 저시력/색약 사용자를 배려한 디자인입니다. - 애니메이션은 자동 재생되더라도 5초 이내 끝나도록 하고, 사용자 입력 없이 반복되지 않게 WCAG 준수합니다. 필요한 경우 애니메이션 비활성 옵션도 제공합니다.

4. 정보 구조 & 사용자 흐름 (IA/Flow)

왜: 파트너 콘솔의 효율적 사용을 위해, 주요 업무 시나리오별로 흐름을 최적화하고 문제없는 예외 처리가 중요합니다. 정보구조(IA)를 명확히 정의하면 메뉴 구성과 권한 체계가 명료해지고, 사용자 흐름도(Flow)를 준비하면 개발 단계에서 누락 없이 구현할 수 있습니다.

무엇: 대표적인 사용자 시나리오별 IA와 Flow를 설계합니다:

- **온보딩 (파트너 등록 프로세스):** 새 제휴사가 콘솔을 처음 사용할 때의 가입 절차입니다. **IA:** 초대 이메일 수신 → 콘솔 회원가입 (비밀번호 설정 등) → 제휴사 정보 입력(회사명, 사업자등록) → 심사 대기 → 승인 후 정식 사용. **Flow:**
- **가입**: 초대링크 → 이메일/임시비번 인증 → 약관동의 → 계정 생성 완료 (성공 화면/로그인 이동).
- **정보 입력(KYB)**: 로그인 후 즉시 온보딩 마법사 시작 → 단계별 입력 (앞서 Section 3-폼) → 제출 → "심사 중" 상태 대시보드 표시.
- **관리자 승인:** 내부 운영자가 제출 서류 검토 → 승인 처리 시 파트너에게 이메일 알림 + 대시보드 상태 "승인됨" 업데이트. 반려 시 사유와 재제출 Flow 안내.
- 승인 완료 후, 파트너는 콘솔 내 모든 메뉴 접근 가능.

예외: 서류 반려 → 파트너가 **다시 제출** (심사 프로세스 반복). 승인 지연 시 N일 후 리마인드 배너 노출 등.

- **API 키 / 웹훅 설정: IA:** 메뉴 구조상 "개발자 설정 > API Keys & Webhooks". **Flow:**
- **API Key 발행:** 운영자는 콘솔에서 "새 API 키 발행" 버튼 클릭 → 모달에서 키 설명 입력/권한 선택 → 생성 → **한번만 노출**되는 Secret 키값 표시 83 (이때 "다시 볼 수 없으므로 복사하십시오" 경고) → 사용자가 복사. 키 목록에 새 키 추가(이름, 생성일, 마지막 사용일).
- **API Key 폐기:** 키 목록에서 폐기 아이콘 클릭 → 확인 모달 (키 이름 입력 확인, 취소/확인 버튼) 84 → 폐기 완료 (리스트에서 삭제, 토스트 "폐기 완료"). 폐기된 키는 DB상 disabled 처리 후 X시간 뒤 완전 삭제.

- **Webhook 설정:** 사용자 흐름 - URL 입력 + 이벤트 선택 → "테스트 전송" 버튼으로 샘플 페이로드 보내 확인 → 저장. 웹훅 비밀키는 `xxxxx`로 마스킹되어 표시되며 "재발급" 버튼으로 새로 생성 가능 (재발급 시 옛 키는 효력 상실). 재발급 역시 확인 다이얼로그를 거칩니다. 예외: URL 검증 실패 (잘못된 HTTPS 등) → 인라인 오류 표시. Webhook 응답에 대한 로그 탭도 제공하여 전달 실패 시 원인을 보여줍니다 (예: "응답 500, 재시도 예정").

- **거래 목록 탐색: IA:** "거래 관리 > 거래 내역" 화면은 테이블로 구현, 상단에 **빠른 필터**(오늘/어제/이번달)와 **고급 필터**(드롭다운 패널) UI를 갖습니다. **Flow:**

- 기본 진입 시 최근 7일간 거래를 날짜 내림차순 표시.
- 사용자가 상단 빠른 필터 "이번 달" 클릭 → 날짜필터 적용 → 테이블 새로고침 (API 호출) → 결과 표시, 해당 버튼 강조상태.
- 추가로 고급 필터에서 상태:"실패", 결제수단:"카드" 선택 → "필터 적용" → 조건 해당 거래만 표시. 이때 상단에 작은 필터태그 2개 노출 ("상태: 실패", "결제: 카드")와 (X 모두지우기).
- 특정 거래 클릭 → **거래 상세 페이지**로 이동 (라우트 분기). 상세페이지에는 거래 정보, 로그, 연관결제 등 탭 구성.

- 상세에서 "환불" 버튼 누르면 모달 열려 환불 금액 확인 → 수행 시 API 호출 & 성공 토스트, 해당 거래상태 "환불됨"으로 즉시 갱신. 예외: 필터 결과 0건이면 Empty State로 "조건에 해당하는 거래 없음" 표시 + 필터초기화 버튼 제공. API 오류 시 배너 경고 표시.

- **정산 리포트 다운로드: IA:** "정산 > 정산 리포트" 메뉴. **Flow:**

- 월별 정산 목록표 (정산월, 금액, 상태, 다운로드 링크) 제공. 기본 현재 연도 리스트.
- 사용자가 특정 월 행의 "다운로드" 아이콘 클릭 → 서버에 엑셀 생성 요청 → 즉시 `<a>` 다운로드 트리거 or "준비중" 인디케이터 표시 후 완료되면 자동 다운로드.
- 상태가 "계산중"인 경우 (정산일 이전) 다운로드 버튼 disabled.

- 과거월 선택하여 "다시보기" 가능. 예외: 파일 생성 실패 (네트워크 등) → 토스트로 실패 안내 및 재시도 유도.

- **알림 & 이벤트 로그: IA:** "시스템 > 감사 로그" 등. **Flow:**

- 알림센터: Topbar 종 아이콘 클릭 → 우측 사이드패널로 최근 알림 표시 (읽지 5개 bold 표시). 각 알림항목 클릭 시 관련 화면으로 이동 또는 팝업. “모두 읽음” 버튼으로 상태 갱신.
- 감사 로그: 사용자 활동/민감정보 열람 등의 로그표를 관리자에게 제공합니다. 필터 (날짜, 사용자, 이벤트 타입 등) 제공하여 검색. 기본은 최근 30일.

- 로그 항목 클릭 시 세부정보 (IP, UserAgent 등) 모달로 표시. 예외: 로그 데이터 방대할 경우 서버페이징 + 가상 스크롤 적용. 권한 없는 사용자는 접근 시 “권한 없음” 페이지를 보여주며, 로그는 서버에서 거르므로 UI에서 노출되지 않습니다.

- **권한/역할 관리: IA:** “설정 > 사용자 및 권한”. **Flow:**

- 오너 계정이 부관리자 계정을 초대 -> 역할 지정 (읽기/쓰기 권한 분리).
- 초대받은 사람은 이메일 통해 가입 -> 해당 권한으로 로그인.
- 권한 편집: 리스트에서 사용자 선택 -> 역할 드롭다운 변경 -> 저장. 본인이 오너일 경우 변경 불가 처리.
- 사용자 삭제: 목록에서 삭제 -> 확인 다이얼로그 -> 삭제 시 그 사용자 로그인 불가, 리스트 제외. 예외: 최소 1명 오너 필요 -> 마지막 오너 삭제 시 경고 및 차단. 권한 없는 사용자가 해당 설정 메뉴 접근시 “권한 없음” 메시지.

예외 처리: - **서류 반려:** 사용자가 서류 재업로드 후 제출하면 상태 "재심사 중"으로 업데이트. 내부 검수자가 이를 특별 표시 (재제출)로 보고 우선 처리. - **API 오류 공통 처리:** 글로벌 axios 인터셉터 등으로 401(세션만료)시 자동 로그아웃 + “세션이 만료되었습니다, 다시 로그인” 알림, 500시 “일시적 오류” 토스트. - **저장 실패 재시도:** Form 제출 실패시 입력값 보존 + 배너로 "전송 실패, 네트워크 확인 후 재시도" 안내. - **권한 오류:** 사용자가 URL로 권한 없는 페이지 접근 시, “접근 권한이 없습니다” 배너와 홈으로 돌아가기 링크 제공. - **중복 작업 방지:** 중요 액션 버튼은 누르면 즉시 disabled+로딩처리하여 중복 submit 막고, 완료/실패 시에만 다시 활성화.

5. 국제화/현지화 규칙

왜: 콘솔은 다국어(한국어/영어) 및 다중 통화(KRW/USD)를 지원해야 합니다. 국제화(i18n) 원칙을 준수하면 한 개 소스 코드로 다양한 언어와 문화권에 대응 가능합니다. 또한 금전/날짜 포맷 등의 현지화(L10n)을 정확히 처리해 혼동을 없앱니다.

무엇: - **통화 표시:** 통화는 로케일별 포맷에 따라 표시합니다. `₩`(원)과 `$`(달러)를 금액 앞에 붙이고, 세자리마다 콤마를 삽입합니다 (₩12,345, $12,345.00 등). 소수점 자리: - KRW: 소수점 없음 (원화는 일반적으로 0 decimal). - USD: 소수점 두 자리 (cents)까지 표시. - 반올림은 금액 산출 단계에서 하고, 표시 시에는 반올림된 값을 그대로. - 음수 금액: 통화기호 앞에 `-` 붙여 `-₩1,000` 형태. 음수인 것을 빨간색 등으로 표시할 수도 있음.
이러한 포맷은 **Unicode CLDR**의 통화 패턴을 따릅니다 [68]. 예컨대 CLDR에 정의된 KRW 패턴은 `"¤#,##0"`, USD는 `"¤#,##0.00"`입니다. 금액을 JS Intl API로 포맷하거나 서버에서 포맷된 문자열을 받습니다. - **불변 소수점 자리:** 환율이나 단가처럼 소수점 고정 자리 필요 시 (예: 1.2345 BTC) 설정된 자리수까지 표시하고 잘립니다.

- **날짜/시간 표기:**
- 기본 표준 시간대는 `Asia/Seoul` (KST)로 합니다. DB 등은 UTC로 저장하되 UI에서는 KST로 변환 표시. 사용자의 지역(영문 사용자 등)에 따라 UTC 또는 선택 TZ 표시도 추후 고려.
- **표기 형식:** 한국어: `YYYY년 M월 D일 (요일) HH:mm` (24시간제) 형태를 기본으로 합니다. 예: 2025년 1월 3일 (금) 15:30. 영어: `MMM D, YYYY hh:mm a` (Jan 3, 2025 3:30 PM) 형태로 합니다.
- 시간대 표시는 KST 기준임을 명시적으로 표시합니다 (예: "2025-01-03 15:30 KST"). UI에는 통상 KST만 쓰지만, 영문 사용자를 위해 "KST" 혹은 GMT+9 표기를 병기합니다.
- **상대시각 지양:** "5분 전", "어제" 같은 상대적 표현은 혼란 야기 가능하므로 콘솔에서는 **절대시각**을 기본으로 합니다. 필요 시 툴팁으로 상대 시간을 보조 표시하는 정도로 제한합니다. (이력 로그 등에서는 “2시간 전” 정도 표시 가능하나 역시 hover 시 실제 시각 제공).

- 지역화: 영어권에서는 월/일 순, 12시간제 등 선호가 달라, 로케일별로 date-fns/Intl.DateTimeFormat 등을 통해 형식 지정합니다.

- **숫자 포맷:** 천 단위 `,` 구분과 소수 `.` 구분은 로케일에 따라 다릅니다. 한국/미국은 `1,234.56`이지만 독일 등 유럽은 `1.234,56` 형식입니다. 현재 대상 로케일 ko-KR/en-US에 맞춰 세팅 (둘 다 그룹 구분자 콤마, 소수점 마침표). 다국어 확장시 CLDR의 숫자 패턴을 사용하도록 구현합니다 [85].

- **텍스트 리소스 관리:** 모든 UI 문구는 소스코드 하드코딩이 아닌 **i18n 리소스 키**로 관리합니다. 예: `"dashboard.welcome_message": "환영합니다, {name}님"` 형태로 별도 JSON에 저장. UI에서는 `t('dashboard.welcome_message', { name: 사용자명 })`으로 호출. **메시지 포맷**은 ICU MessageFormat을 사용해 복잡한 문법(복수형, 성별)도 처리합니다 [86].

- **복수형 처리:** 예: "{count}건의 알림" vs "{count}건의 알림들"과 같은 한국어 복수형은 차이 없지만 영어는 "1 alert" vs "2 alerts"로 S 첨가 필요. MessageFormat의 `plural` 기능으로 구현합니다 [87].
- **성별 등격식:** 필요 시 Placeholder로 성별에 따라 문구가 바뀌도록 (예: "{user}님이 자신의 프로필을 수정했습니다" vs "그녀/그가 ...") 할 수 있으나, 일반적으로 콘솔 문맥에서는 성별에 따른 표현은 없음.

- **형식 일관성:** Key 네이밍은 도메인별로 그룹화 (예: navigation.home, navigation.settings; error.network_timeout 등). 영문 번역은 전문 번역가가 수행하며, UI 상 긴 텍스트 줄바꿈 등은 번역 과정에서 고려합니다.

- **문화권 차이:**

- 주소 입력순서: 한국은 "우편번호, 주소", 영문은 "Street, City, State, ZIP". 폼에서는 나라에 따라 필드 순서/레이블을 스위치합니다.
- 달력 시작요일: 기본 월요일 시작 (ISO주), 필요시 locale 기반 Sunday start.
- **시간제:** 한국어 UI는 24시간제, 영어(US)는 12시간제 AM/PM [88].
- 요일 표기: 다국어 약칭 적용 (Mon/Tue vs 월/화).
- 기타: 이름 표시 (영문 Last Name + First Name vs 한글 이름), 화폐단위 (달러 기호, 원화 표시) 등 모두 로케일 컨벤션 준수.

어떻게: - React Intl, i18next 등의 라이브러리를 사용하여 다국어 지원 구현. - 런타임에 사용자 프로필의 language 설정에 따라 해당 리소스 불러오기 (코드 스플리팅으로 각 언어 JSON 분리). - 번역 부재 키는 fallback (영어)로 표시하고, 콘솔 내 언어전환 드롭다운을 제공하여 즉시 locale 변경을 가능케 합니다. - 금액/날짜 등은 Intl API 사용. 예: `new Intl.NumberFormat('ko-KR', { style: 'currency', currency: 'KRW' }).format(val)` 형태로. - ICU 메시지 활용 예:

```
"alerts.count": "{count, plural, one {새 알림 1건} other {새 알림 #건}}"
```

- count=1이면 "새 알림 1건", count=5이면 "새 알림 5건" 출력 [86].

- CLDR 데이터 업데이트 및 ICU Plural Rules는 정기적으로 검토/반영 (예: 통화 기호 변경 등)합니다.

6. 보안/신뢰 UX

왜: 금융 콘솔은 민감한 정보와 중요한 기능을 다루므로, **보안에 대한 신뢰**를 사용자에게 주어야 합니다. 정보 노출을 최소화하고, 불가피한 노출 시에는 투명하게 기록하고 알리는 UX가 필요합니다. 또한 위험한 작업은 실수로 발생하지 않도록 사용자 확인 절차를 두어 안전망을 제공합니다.

무엇: - **민감정보 기본 마스킹:** 앞서 언급한 대로 계정번호, API Secret 등은 기본 숨김 표시(●●● 또는 ***)로 처리합니다. UI 어디서든 민감정보(개인식별번호, 카드 CVV 등)는 *최소한**만 노출 (마지막 4자리 등)하고, 전체보기가 꼭 필요할 때만 제한적으로 허용합니다. - **역마스킹 일시 허용:** 권한 있는 사용자(예: 오너/관리자)는 필요 시 해당 값을 **일시적**

으로 볼 수 있습니다. 방법: “눈” 아이콘 토글 → 5초간 평문 표시 후 자동 다시 마스킹. 이 기능 사용 시 곧바로 **감사 로그에 기록**하고, UI 상에도 노출 중임을 배너나 토스트로 경고합니다. 예: “민감 정보가 표시됩니다 - 5초 후 자동 가림” 배너 [39] [40]. 해당 아이콘 버튼에는 `aria-pressed`로 현재 보임/숨김 상태를 전달합니다. - **감사 로그 필수 기록:** 모든 민감데이터 열람 행위(역마스킹, 파일 다운로드 등)는 `audit.log`에 남깁니다 (사용자, 시각, IP, 대상 데이터 식별자 등). 또한 UI에도 "열람 이력" 버튼을 제공해 누가 마지막으로 봤는지 표시해 신뢰성을 높입니다. (예: "이 API키는 2025-01-03 15:32 홍길동님이 조회함" 작은 폰트로 표시). - **위험 액션 2단 확인:** 앞서 설명한대로, 되돌릴 수 없는 **위험 동작**(API 키 삭제, 정산계좌 변경 등)은 **Confirm 모달**을 거칩니다. 모달 메시지에는 "이 작업은 취소할 수 없습니다" 등의 경고 문구를 붉은색으로 강조합니다. 단순 확인 클릭 외에, 추가로 사용자에게 특정 단어나 해당 대상명을 입력하도록 요구합니다 (이를 **확인 텍스트**라 부름) [84]. 예: “삭제하려면 OK를 입력하십시오: [OK]”. 이중 확인은 사용자의 의도를 확실히 하고, 실수 방지를 위함입니다. 또한 여기서도 `aria-required` 등을 활용해 SR 사용자가 이 입력 없이는 진행 안 됨을 알게 합니다. - **세션 관리:** 보안을 위해 **자동 로그아웃** 정책 적용. 일정 시간(예: 15분) 사용자 인터랙션 없으면 경고 모달("곧 로그아웃됩니다, 연장하려면 확인 클릭")를 띄우고 [26], 추가 1분 후에도 반응 없으면 세션 만료 처리합니다. 만료 시에는 로그인 페이지로 리디렉션하고, "세션이 만료되어 로그아웃되었습니다" 알림 배너를 보여줍니다. - 또한 동일 계정 중복 로그인 발생 시 이전 세션 무효화 정책이나 알림을 둘 수 있습니다 (관리자 판단). - **권한 분리:** 운영자(role: admin)와 파트너사 사용자(role: partner)가 볼 수 있는 메뉴/데이터를 엄격히 구분합니다. UI에서는 권한없는 메뉴는 처음부터 숨기며 (또는 비활성화), URL 접근을 시도해도 "권한 없음" 메시지. 중요한 쓰기 동작(정산계좌 변경 등)은 관리자만 버튼이 노출되고, 일반사용자는 해당 UI를 읽기전용으로 표시하거나 아예 제외합니다. - **HTTPS와 MFA:** 콘솔 접근은 항상 HTTPS로만 허용하고 (HSTS 적용 권장), 관리자 계정은 2차 인증(MFA) 옵션을 제공합니다. UI적으로 MFA 설정 메뉴를 제공하고, 활성화 시 로그인 프로세스에 OTP 입력 화면 등을 추가합니다. - **비밀번호 입력 UX:** 새 비밀번호 생성/변경시에 **강도 표시기**(예: 약함/강함 progress bar)를 제공해 안전한 비밀번호를 유도합니다. 입력한 비밀번호는 기본 *** 표시지만, 필요시 show/hide 토글 기능으로 볼 수 있게 (이 역시 aria-pressed로 상태 표시) 합니다.

어떻게: - 클라이언트 측에서 민감정보를 가리지 않고 받아야 할 경우에도 (예: AES 복호화 후), 상태 토글에 따라 UI에만 가시화하고 DOM상 텍스트는 필요시 `text-security: disc` (안전하지 않은거라 대부분 쓰지 않지만) 등을 활용하거나, 전송 직후 문자열을 메모리에서 폐기하는 등 신경씁니다. - 모든 API 호출에는 인증토큰 외에 CSRF 방어가 적용된 상태로 보내고, 중요 액션에는 서버에서 재확인(2차 토큰이나 MFA) 절차를 요구할 수 있습니다. UI에서는 이를 고려해 UX 흐름을 설계합니다. (예: 계좌변경 -> OTP 확인 모달 -> 완료). - 사용자에게 보안팁을 주기 위해, 로그인 화면 등에 "공용 PC 사용 시 로그아웃을 확실히 하세요" 등의 안내문구를 표시합니다. - **신뢰 배지:** 콘솔 하단에 보안 인증 배지(예: ISO27001)나 버전 정보, 마지막 로그인 IP 등을 보여줘 투명성을 높입니다. (마지막 로그인 정보는 UI/로그인 시 배너로도 표시 가능: "마지막 접속: 2025-01-01 12:00, IP 1.2.3.4") - **로그인 보호:** 일정 횟수 이상 로그인 실패시 CAPTCHA 또는 일정시간 계정 잠금 기능을 UX 흐름에 포함합니다. 이때 잠금 풀리는 시점을 사용자에게 안내합니다.

7. 성능/기술 가이드 (React 기반)

왜: 대시보드 화면과 대용량 데이터 테이블 등 성능에 민감한 요소가 많습니다. React 기반으로 개발 시 **최적화 전략**을 명시하여 초기 로딩과 런타임 성능 목표를 충족해야 합니다. 또한 모듈/라이브러리 선정과 코드 구조를 가이드하여 개발자가 공통 기준으로 작업하도록 합니다.

무엇: - **상태 관리 & 캐싱:** 서버 데이터는 **TanStack React Query** (React Query v4)로 관리합니다. Query를 통해 API 호출하고, 응답 데이터를 캐시하여 동일 쿼리 재호출 시 로딩 지연을 최소화합니다. React Query의 **stale-while-revalidate** 정책을 따른다: 이미 가져온 데이터는 일단 즉시 표시하고, 백그라운드에서 최신 데이터를 가져와 갱신합니다 [89]. 이를 통해 사용자에게는 빠른 응답성을 주면서도 데이터 일관성을 유지합니다. - 각 Query에는 `staleTime`과 `cacheTime`을 적절히 설정해 (예: 일반 목록 5초 staleTime) 너무 잦은 refetch를 피합니다. - **에러 처리:** Query 옵션의 `onError`에서 에러 메시지를 Toast로 알리거나, Error Boundary를 사용하여 오류 상태 컴포넌트를 표시합니다. 전역 ErrorBoundary를 `<App>` 상단에 두어 치명적 렌더 오류 발생 시 사용자에게 “문제가 발생했습니다” 화면을 표시하고, Sentry 등으로 로그합니다. - **스켈레톤 로딩:** Query 로딩 상태일 때, 기존 데이터가 있으면 유지하고 `status==='loading'` + `fetchStatus==='fetching'`을 구분하여 필요하면 Skeleton UI를 보여줍니다 [90]. (React Query의 `isFetching`를 이용). - **중복 요청 방지:** 동일 키로 다수 컴포넌트에서 호출 시 React

Query가 하나만 실행하고 나머지는 캐시를 활용하게 구성합니다. 또한 화면 숨김시엔 `enabled: false` 옵션을 활용합니다.

- **폼 상태 관리:** React Hook Form 라이브러리를 사용하여 폼 상태를 관리합니다. 이를 통해 비제어 컴포넌트로 DOM 조작을 최소화하고, 검증 룰(yup/zod 등 연계)도 쉽게 적용합니다. RHF의 `useForm`으로 form 레퍼런스를 만들고 `<form onSubmit={handleSubmit(onSubmit)}>`. 각 `<input {...register('fieldName', { required: true })}>` 식으로 작성합니다. 이로써 수백 개 필드도 성능 저하 없이 핸들링 가능합니다. 또한 RHF의 form state (`isDirty`, `errors`)를 활용해 UI에 실시간 반영 (저장 버튼 활성화 등)을 구현합니다.

- **테이블 구현:** 테이블은 **TanStack Table (React Table)** 또는 **AG Grid** 중 하나를 채택합니다.

- TanStack Table v8의 headless 구조를 이용하면 UI를 커스텀하기 용이합니다. 가상 스크롤은 TanStack Virtual과 결합합니다 66 .
- AG Grid는 full-featured 솔루션으로, 라이센스 고려 후 채택 가능합니다. AG Grid의 Enterprise 기능 (컬럼 그룹핑, Excel 내보내기 등)이 필요하면 포함하고, Tree Data 등 시나리오도 활용 가능합니다.
- 어떤 라이브러리든 **열 정의 패턴**을 정해두어 개발 편의를 높입니다. 예: 열 설정을 배열로 선언:

```
const columns = [
 { key: 'date', header: '날짜', width: 120, cell: formatDateCell },
 { key: 'amount', header: '금액', align: 'right', cell: formatCurrency }
];
```

이 정의를 기반으로 Table 헤더/셀을 map 렌더링하거나, Grid 콤포넌트에 넘깁니다.

- pagination, sorting 등은 라이브러리 제공 함수 활용. (TanStack Table의 useSortBy, usePagination 플러그인 등)

- **코드 분할 (Code Splitting):** 번들 크기를 줄이기 위해 **라우트 단위**로 코드 스플리팅을 적용합니다. React.lazy/Suspense를 사용하여, 각 주요 페이지 컴포넌트를 분리 로드합니다. 예:

```
const SettlementPage = React.lazy(() => import('./SettlementPage'));
```

라우터에서 해당 경로 접근시 suspense fallback (스켈레톤 화면) 보여주고 비동기로 로드합니다.

- 또한 큰 라이브러리(예: chart.js, rich text editor 등)는 필요한 시점에만 동적 import합니다.
- 아이콘 라이브러리는 개별 아이콘만 import (tree shaking 지원)하는 것을 엄격 준수합니다. (예: `import { IconName } from '@mui/icons-material/IconName';` 대신 필요한 것만).

- 번들 분석 툴(Webpack Bundle Analyzer 등)로 주기적으로 확인하여, 중복 포함된 모듈이나 필요없는 폴리필 제거 등 최적화합니다.

- **이미지 최적화:** 모든 SVG 아이콘은 inline SVG 또는 SVG sprite로 사용하여 HTTP 요청 수를 줄입니다. 사진 등 비주얼 에셋은 Build 시 이미지 압축 처리하고, `<img srcset>`이나 `<picture>`를 사용해 고해상도(2x) 화면 대응 및 느린 네트워크 대응합니다. 필요 시 Lazy Loading (`loading="lazy"`)을 적용합니다.

- 차트 같은 canvas는 toBlob() 활용시에도 품질 설정을 조정해 용량을 줄이고, 다운로드시 파일명을 의미있게 지정합니다.

- **Web Vitals 및 성능 목표:**

- 초기 페이지 LCP(Largest Contentful Paint)를 데스크톱 기준 **2.5초 이하**로 유지합니다 [91] . 이를 위해 critical CSS 최소화, 폰트 preload, API 병렬처리 등을 합니다.
- FID(첫 입력 지연) 및 TTI(대화형 시간)도 짧게 (<100ms) 하도록 Main thread 블로킹을 피합니다. (대용량 데이터 가공은 Web Worker 위임 검토)

- Scroll 성능: 특히 표 가상스크롤 영역 60fps 이상이 목표. 개발 단계 Lighthouse와 profiler로 확인합니다.

- **메모리 관리:** SPA 환경에서 메모리 릭이 없도록, unmount 시 이벤트리스너 제거, 타이머 clear 등을 확실히 합니다. 대량 데이터는 필요 시 WebWorker로 이동해 메인 쓰레드 부담을 낮춥니다.

- **모니터링:** 사용자 환경 성능 모니터링 도구 (예: NewRelic, Google Analytics Web Vitals) 설정하여, 릴리스 후에도 Core Web Vitals 지표를 수집합니다. 목표: LCP p75 < 2.5s (Good), CLS < 0.1, INP < 200ms 등. 이를 분기별로 리뷰합니다.

기술 스택: React 18+, CRA 또는 Vite 기반, 상태관리 TanStack Query + Context API (필요시 Redux Toolkit은 최소화). UI 프레임워크는 Ant Design 5 또는 MUI 5 중 선정 (현재 Ant Design 언급 있으므로 AntD 가정). Table/chart 등은 상기 specialized library 사용.

8. 접근성 체크리스트 (WCAG 2.2 AA 기준)

본 콘솔은 **WCAG 2.2 AA** 준수를 목표로 합니다. 다음 체크리스트를 통해 개발/디자인을 검수합니다:

- **포커스 표시 및 순서:**
- 모든 인터랙티브 요소는 키보드 Tab으로 접근 가능해야 하고, 포커스가 갔을 때 뚜렷이 보이는 indicator가 있다 [34] . 기본 브라우저 아웃라인을 제거했다면 커스텀 outline 적용 필수.
- 포커스 이동 순서는 논리적이어야 함 (DOM 순서가 시각적 읽는 순서와 일치) [62] . 모달 열리면 내부 첫 요소로, 닫히면 이전 트리거로 돌아가는지 확인 [58] .
- **키보드 트랩 금지:** 특정 컨트롤 내에서 ESC나 특정 조작 없이 갇히지 않도록. 예: 드롭다운 메뉴는 Esc 누르면 닫히고 포커스 원래 버튼으로 복귀.

- 스킵 링크 제공 여부: 필요한 경우 메인 콘텐츠로 바로 가는 숨김 링크를 페이지 최상단에 두어 SR/키보드 사용자가 네비게이션 건너뛰게 합니다.

- **ARIA 및 역할 적절성:**

- ARIA role 사용 시 실제 기능과 맞는 role인지 검토. (예: 팝업 메뉴는 `role="menu"` + `role="menuitem"`, 데이터테이블은 `<table>` 그대로 또는 ARIA grid 등).
- ARIA 속성 (`aria-expanded` , `aria-selected` , `aria-controls` 등) 값 토글이 상태변경에 따라 제대로 업데이트 되는지. (예: 사이드바 토글 버튼 `aria-expanded` true/false)

- 라이브 영역(`aria-live`) 사용 부분이 필요한 경우에만 있고, 중복으로 여러곳에 assertive 없도록. 중요한 알림은 하나의 `role="alert"` 로 집중.

- **색 대비:**

- 텍스트와 배경의 **명도 대비**가 작은 텍스트 4.5:1 이상, 큰 텍스트(18pt 이상 또는 Bold 14pt 이상) 3:1 이상인지 검사 [13] . 아이콘 등 UI 필수 요소도 3:1 이상이어야.
- 다크모드에서도 동일 대비 기준 충족하는지 (특히 경고색 위 흰 텍스트 등 inverse 토큰 활용 확인) [15] .

- 색상만으로 정보를 전달하지 않았는지. (예: “빨간색은 실패, 초록은 성공”이 아닌, 아이콘이나 레이블도 함께 사용) [92] .

- **텍스트 대체:**

- 모든 의미있는 아이콘과 이미지에는 적절한 대체텍스트(alt 또는 aria-label)가 있다 [31] . 장식용 이미지는 `aria-hidden="true"` or `role="presentation"` 처리.

- 차트 등 캔버스 그래픽에는 데이터 요약을 별도 제공 (table이나 aria-describedby).

- **양식 레이블 및 오류:**

- 모든 입력엔 `<label>`이 연결돼 있으며, placeholder는 레이블 대체로 쓰이지 않았다 [44] . (이름없는 아이콘 버튼 등도 ARIA Label 있음)
- 에러 발생 시 해당 입력에 자동 포커스 이동하고, 오류 메시지를 `aria-live`로 읽어준다 [42] . 또한 오류난 입력의 `aria-invalid="true"` 속성이 추가된다.

- 필수 입력은 `aria-required="true"`나 `required`로 표기.

- **도움말 및 지시:**

- 입력 도움말 (예: 포맷 안내)은 `aria-describedby`로 연결되어 SR이 필요 시 듣게 함.

- 복잡한 위젯 사용법(예: 그리드 셀 이동 키)은 문서나 툴팁으로 제공, SR에도 접근 가능.

- **테이블 구조:**

- `<table>` 사용 시 `<th>`에 `scope`나 `headers` 속성으로 셀과 헤더 연결 확인 [70] .
- ARIA grid 사용 시, 초점 이동 및 `aria-colindex/rowindex`가 올바르게 적용됐는지 (라이브러리에 따라 자동 관리).

- 표에 캡션(`<caption>`)이나 대체설명 마련.

- **초점 관리 (모달 등):**

- 모달/다이얼로그 열리면 첫 포커스 이동 & 뒷배경 inert 처리 [57] , 닫으면 이전 요소 복귀 [58] .
- 모달 안 Esc키로 닫힘 지원여부 확인 [56] .

- Drawer 등 모달과 유사한 overlay도 동일 검토.

- **다크모드 대비:**

- 다크모드에서 특정 색상 조합이 여전히 읽기 쉬운지 확인 (특히 경고 노란 배경에 흰색글자 대비 등, Atlassian처럼 inverse 토큰 활용) [15] .

- 다크모드 전환 시 깜빡임/과도한 애니메이션 없이 매끄러운지 (CSS 변수로 즉시 변경, 트랜지션 0.3s 이내).

- **키보드 순환:**

- 페이지 내 헤더-메인-푸터 등 영역 간 순환이 논리적.

- 사이드바 메뉴는 Arrow로 이동 가능, Enter로 활성화 (혹은 단순 Tab 순서라도 계층적 메뉴는 Arrow Nav 지원 검토).
- ESC로 열린 메뉴/툴팁 닫히는지.
- 팝오버, 툴팁 등은 키보드로도 트리거 가능 (버튼으로 제공)하고 `Escape`로 닫힘.

- **라이브 영역 활용:**

- 통상 토스트 알림은 `role="status"`로, 오류는 `role="alert"`로 구현 [56]. 적절히 적용됐는지.
- aria-live로 불필요한 내용을 매번 읽지 않도록 (동적인 숫자 업데이트 등 주의).

- **반응형 접근성:**

- 모바일 화면에서도 터치 영역 최소 24px 이상인지 확인 (WCAG 2.5.8), 실제 대부분 버튼 40px+로 설계됨 [26].
- 콘텐츠 reflow 시 의미 손실 없는지 (예: 카드 2열 -> 1열 돼도 순서 논리적).
- 스크린 방향 전환이나 축소 확대 시에도 UI가 깨지지 않고 사용 가능.

- **언어 속성:**

- 페이지 최상단 `<html lang="ko">` 등 올바른 언어 지정. 영문 전환 시 `lang` 바뀌는지.
- 텍스트 내 특수 어구(예: 영문 문장 포함 시)에는 `lang` 속성으로 해당 부분 언어 표기.

- **초점 가시성:**

- 초점시 요소가 가려지거나 화면 밖이면, JS로 `scrollIntoView` 처리한다. 특히 가상 리스트나 탭 전환 시 체크.
- Tab 순서가 모달 오픈 시 behind 요소로 빠져나가지 않도록 (Focus trap) [57].

위 항목들을 QA 단계에서 수동 및 자동 도구(예: Axe, Lighthouse)로 점검하여 발견된 문제를 수정합니다. 체크리스트를 개발/디자인 완료 검토에 포함해 접근성 준수를 지속적으로 관리합니다.

9. 레퍼런스 (근거 및 버전)

- **Material Design 3 (Google)** – Material Design Guidelines, Adaptive design, Dark theme, etc. (2023) [23] [91]
- **Ant Design 5 (Ant Group)** – Ant Design Components, Button/Input specs, Design Tokens (v5.0, 2023) [93] [27]
- **Atlassian Design System** – Foundations: Grid, Color, Spacing, Typography, Components: Modal, Flag, etc. (v2.1, 2025) [20] [13]
- **Shopify Polaris** – Polaris Design System, Card layout, Tokens, Color roles. (v12, 2025) [45] [50]
- **IBM Carbon Design** – Data Visualization Guidelines, accessibility in charts. (v10, 2024) – (참고: 색각 보정 패턴)
- **WCAG 2.2 (W3C)** – Web Content Accessibility Guidelines 2.2, Criteria 1.4.3 Contrast, 2.4.7 Focus Visible, 2.5.8 Target Size. (W3C Recommendation Draft, 2025) [13] [26]
- **WAI-ARIA Authoring Practices** – ARIA APG 1.2, Patterns: Dialog (Modal), Grid (interactive table). (W3C, 2022) [57] [75]
- **ICU MessageFormat / Unicode CLDR** – ICU User Guide (ver. 67) for plural & gender localization [86], CLDR v42 for locale number/currency formats [68].

- **TanStack React Query** – React Query v4 Docs, Caching & stale-while-revalidate patterns. (2023) [89]
- **AG Grid & TanStack Table** – AG Grid Documentation v28 (CSV Export, Column resize) [82], TanStack Table v8 Guide.
- **Stripe API Reference** – Stripe API Keys best practices. (2025) [83]
- **Dmitry Sergushkin,** "Best UX Practices for Designing a Sidebar", UXPlanet (Dec 2024) [94].
- **WebAIM** – Accessibility for Icon Buttons, Carie Fisher's A11Y Style Guide (2023) [81].
- **Core Web Vitals** – Largest Contentful Paint, web.dev (2023) [91].

(모든 링크된 레퍼런스는 해당 내용의 출처를 나타내며, 버전과 날짜는 각 괄호에 표기했습니다.)

---

[1] [2] [3] [5] [6] [7] [8] [13] [15] [16] [67] Overview - Border - Atlassian Design System
https://atlassian.design/foundations/color-new/

[4] Overview - Spacing - Atlassian Design System
https://atlassian.design/foundations/spacing

[9] Overview - Typography - Atlassian Design System
https://atlassian.design/foundations/typography-beta

[10] What is the 8-Point Grid System? - LinkedIn
https://www.linkedin.com/pulse/what-8point-grid-system-stephen-paul-jbtkf

[11] Design Tokens: The Smarter Way to Style Your UI | by Ashish Garg
https://www.designsystemscollective.com/design-tokens-the-smarter-way-to-style-your-ui-1550346e282c

[12] [20] Overview - Iconography - Atlassian Design System
https://atlassian.design/foundations/grid-beta/

[14] [35] Tokens — Shopify Polaris React
https://polaris-react.shopify.com/design/colors/color-tokens

[17] [18] [21] [94] Best UX Practices for Designing a Sidebar | by Dmitry Sergushkin | UX Planet
https://uxplanet.org/best-ux-practices-for-designing-a-sidebar-9174ee0ecaa2?gi=603dcd73f445

[19] Navigation system - Navigation system - Components - Atlassian Design System
https://atlassian.design/components/navigation-system/

[22] [24] [25] [30] [88] Dark mode - Material UI
https://mui.com/material-ui/customization/dark-mode/

[23] Dark theme - Material Design
https://m2.material.io/design/color/dark-theme.html

[26] [33] Target Size - Complete Guide | Web Accessibility Glossary | WebAbility
https://www.webability.io/glossary/target-size

[27] Input - Ant Design
https://4x-ant-design.antgroup.com/components/input/

[28] [29] [32] [93] Button - Ant Design
https://ant.design/components/button/

[31] [34] [81] [92] A11Y Style Guide
https://a11y-style-guide.com/style-guide/section-general.html

36 Design tokens - Tokens - Components - Atlassian Design System
https://atlassian.design/components/tokens/

37 55 57 58 59 60 61 62 63 64 65 Dialog (Modal) Pattern | APG | WAI | W3C
https://www.w3.org/WAI/ARIA/apg/patterns/dialog-modal/

38 39 40 41 Make an accessible password reveal input | Make Things Accessible
https://www.makethingsaccessible.com/guides/make-an-accessible-password-reveal-input/

42 44 Errors - Patterns - Material Design
https://m1.material.io/patterns/errors.html

43 70 71 72 73 74 75 ARIA: grid role - ARIA | MDN
https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Reference/Roles/grid_role

45 46 47 48 49 50 51 52 53 Card layout — Shopify Polaris React
https://polaris-react.shopify.com/patterns/card-layout

54 Color — Shopify Polaris React
https://polaris-react.shopify.com/tokens/color

56 80 Modal — Shopify Polaris React
https://polaris-react.shopify.com/components/deprecated/modal

66 TanStack Virtual
https://tanstack.com/virtual

68 Formatting localized currency — Shopify Polaris React
https://polaris-react.shopify.com/foundations/formatting-localized-currency

69 90 Breakpoints — Shopify Polaris React
https://polaris-react.shopify.com/tokens/breakpoints

76 77 78 Grid (Interactive Tabular Data and Layout Containers) Pattern | APG | WAI | W3C
https://www.w3.org/WAI/ARIA/apg/patterns/grid/

79 Steppers - Components - Material Design
https://m1.material.io/components/steppers.html

82 JavaScript Grid: CSV Export
https://www.ag-grid.com/javascript-data-grid/csv-export/

83 docs.stripe.com
https://docs.stripe.com/keys

84 Manage Configurations and Secrets - Choreo Documentation - WSO2
https://wso2.com/choreo/docs/devops-and-ci-cd/manage-configurations-and-secrets/

85 Number and currency patterns - Unicode CLDR Project
https://cldr.unicode.org/translation/number-currency-formats/number-and-currency-patterns

86 ICU message format: Guide to plurals, dates & localization syntax
https://simplelocalize.io/blog/posts/what-is-icu/

87 Formatting Messages | ICU Documentation
https://unicode-org.github.io/icu/userguide/format_parse/messages/

89 React Query: Making Your Server-State Problems Disappear (Like ...
https://dev.to/sathish/react-query-making-your-server-state-problems-disappear-like-magic-2fjn

91 Largest Contentful Paint (LCP) | Articles - web.dev
https://web.dev/articles/lcp