ThirdEye — Select.dev–Inspired UI (Greenfield) + Cursor Tasks

> **Branding:** Company = **ThirdEye**. All UI copy, code, env vars, and docs use *ThirdEye* (no OpenMetadata terms).
> **Copyright:** © ThirdEye. Footer snippet provided below.

Tech & Conventions

- **Framework:** Next.js (App Router) + TypeScript
- **UI:** Tailwind CSS + shadcn/ui + lucide-react (select.dev–like neutral/compact theme)
- **State/Data:** React Query
- **Charts:** Recharts
- **Tables:** TanStack Table
- **Forms:** React Hook Form + Zod
- **Graph:** React Flow
- **Testing:** Vitest + Testing Library + Playwright (smoke e2e)
- **Auth:** ThirdEye API token (httpOnly cookie via action route)
- **Env:** `THIRDEYE_BASE_URL`, token saved via UI (cookie)

Folders

```
app/
  layout.tsx
  page.tsx                 # Home
  (app)/explore/page.tsx
  (app)/insights/page.tsx
  (app)/rule-agent/page.tsx
  (app)/settings/page.tsx
  (app)/settings/connectors/page.tsx
  (app)/settings/users/page.tsx
  api/connection/route.ts  # save baseURL+token to cookie
components/
  chrome/Sidebar.tsx
  chrome/Topbar.tsx
  chrome/AppShell.tsx
  chrome/Footer.tsx        # © ThirdEye
  ui/KpiCard.tsx
  data/StackedBar.tsx
  data/BasicTable.tsx
lib/
  theme.ts
  cookies.ts
  teClient.ts              # ThirdEye API client
styles/globals.css
```

Key Snippets

1) Cookie + Client

```
// lib/cookies.ts
'use server';
import { cookies } from 'next/headers';
const KEY = 'te_conn';
export function getConn(){
  const raw = cookies().get(KEY)?.value;
  return raw ? JSON.parse(raw) as { baseUrl:string; token:string } : null;
}
export function setConn(v:{baseUrl:string; token:string}){
  cookies().set(KEY, JSON.stringify(v), { httpOnly:true, secure:true,
sameSite:'lax', path:'/' });
}
```

```
// lib/teClient.ts
export async function teFetch<T>(path:string, init:RequestInit = {}){
  const { getConn } = await import('./cookies');
  const conn = getConn();
  if(!conn) throw new Error('Not connected');
  const res = await fetch(`${conn.baseUrl}${path}`, {
    ...init,
    headers: { 'Authorization': `Bearer ${conn.token}`, 'Content-
Type':'application/json', ...(init.headers||{}) },
    cache: 'no-store',
  });
  if(!res.ok) throw new Error(`${res.status}`);
  return res.json() as Promise<T>;
}
// examples
export const listUsers = () => teFetch<any>('/api/v1/users');
export const listTeams = () => teFetch<any>('/api/v1/teams');
```

```
// app/api/connection/route.ts
import { NextRequest, NextResponse } from 'next/server';
import { setConn } from '@/lib/cookies';
export async function POST(req:NextRequest){
  const body = await req.json(); // { baseUrl, token }
  setConn(body);
  return NextResponse.json({ ok:true });
}
```

2) Chrome + Footer

```
// components/chrome/Sidebar.tsx (excerpt)
import Footer from './Footer';
...
  <div className="p-3 border-t flex items-center gap-3"> ... user block ... </
div>
  <Footer/>
```

```
// components/chrome/Footer.tsx
export default function Footer(){
  return (
    <div className="text-[11px] text-neutral-500 px-3 py-2 border-t">
      © {new Date().getFullYear()} ThirdEye. All rights reserved.
    </div>
  );
}
```

```
// components/chrome/Topbar.tsx (excerpt label)
<header className="h-14 border-b bg-white/70 backdrop-blur flex items-center
px-4 gap-4">
  {/* search etc. */}
</header>
```

```
// components/chrome/Sidebar.tsx (brand title excerpt)
<div className="p-4 text-lg font-semibold tracking-tight">ThirdEye</div>
```

3) Env

```
THIRDEYE_BASE_URL=https://api.thirdeye.example.com
NEXT_PUBLIC_APP_NAME=ThirdEye Observability UI
```

Cursor Task List — ThirdEye (Select.dev–Inspired)

One PR per task unless noted. Copy/paste into Cursor.

CP-S1 — Bootstrap + Theme

Branch: feat/select-theme
Goal: Tailwind + shadcn + neutral token set (select.dev vibe).
Do: Install deps; init shadcn; add Button/Input/Select/Dialog/Sheet/Tabs/Tooltip/Toast; add theme tokens (radius, shadows, borders).
AC: `/ui` gallery renders with compact density and soft shadows.

CP-S2 — App Chrome (Sidebar/Topbar)

Branch: feat/chrome-shell
Goal: Sidebar **Home/Explore/Insights/RuleAgent/Settings**; user block pinned at bottom; Topbar search + period chip.
AC: Nav works; active highlight; keyboard focus rings.

CP-S3 — Connection Setup

Branch: feat/connection
Goal: Save ThirdEye baseURL+token to httpOnly cookie via `/api/connection`.
AC: Test Connection calls `/api/v1/health` (or `/versions`) and shows Connected/Failed.

CP-S4 — ThirdEye Client

Branch: feat/te-client
Goal: `teClient.ts` fetch helper + typed fns (`listUsers`, `listTeams`, `listServices`, `search`, etc.).
AC: Vitest mocks pass; errors normalized.

CP-S5 — Home Overview

Branch: feat/home-overview
Goal: KPI cards + stacked bar + workload table (mock first).
AC: Responsive; loading/empty/error handled.

CP-S6 — Explore (Search)

Branch: feat/explore
Goal: Federated search via `POST /api/v1/search/query` with filters (entityType/owner/tag).
AC: Debounced search; URL state; keyboard nav.

CP-S7 — Insights (Trends)

Branch: feat/insights
Goal: Runs by status, avg duration, cost (if available) with time-range chips.
AC: Range change updates charts smoothly.

CP-S8 — Workloads Page

Branch: feat/workloads
Goal: Stacked daily cost bars + sortable workload table with % change badges and cost bars (select.dev style).
AC: 60 FPS scroll; sort/persist via URL.

CP-S9 — Settings → User Management (Read-only)

Branch: feat/settings-users
Goal: Users/Teams lists with quick filters and role badges.
AC: Paginated; standardized empty/error.

CP-S10 — Settings → Connectors: Snowflake

Branch: feat/connector-snowflake
Goal: Create/update **Snowflake Database Service** + **Ingestion Pipeline** via ThirdEye APIs.
Fields: Account, Auth (user/pass or key), Warehouse, Role, DB/Schema include-exclude, Schedule (cron), Secrets (masked).
AC: Save → success toast; appears in Services list; Zod validation.

CP-S11 — Settings → Connectors: dbt

Branch: feat/connector-dbt
Goal: Link dbt artifacts/jobs via ThirdEye APIs (pipeline service + metadata ingestion).
Fields: Artifacts path, Project, Target, Schedule.
AC: New pipeline visible in list with basic status panel.

CP-S12 — RuleAgent (Rules & Actions)

Branch: feat/rule-agent
Goal: Rules (name, trigger, condition, action, status) + test-send to webhook/Slack URL.
AC: Create/edit/delete locally; validations enforced.

CP-S13 — RBAC (Read-only respect)

Branch: feat/rbac
Goal: Hide/disable actions based on ThirdEye roles/permissions.
AC: Snapshot tests show disabled actions for viewer.

CP-S14 — Theming & Dark Mode

Branch: feat/dark-mode
Goal: System-aware dark theme; toggle persists.
AC: WCAG AA; charts/tables readable.

CP-S15 — Error/Loading/Empty UX

Branch: chore/ux-states
Goal: Shared `<EmptyState/>`, `<ErrorState onRetry/>`, `<Skeleton/>`.
AC: All lists use shared states.

CP-S16 — E2E Smoke Tests

Branch: test/e2e
Goal: Playwright tests for connection, Home, Explore, Settings→Connectors.
AC: CI green.

CP-S17 — Packaging

Branch: chore/docker-ci
Goal: Dockerfile + compose + GitHub Actions (build/test).
AC: `docker compose up` serves app with `THIRDEYE_BASE_URL` env.

CP-S18 — Perf Pass

Branch: chore/perf
Goal: Virtualize big tables; memoize heavy cells; code-split.
AC: Lighthouse ≥ 85.

Notes

- All copy and code use **ThirdEye** naming; no references to any other OSS names.
- If your backend is wire-compatible with OM, keep that detail internal; this UI stays vendor-neutral under the **ThirdEye** brand.