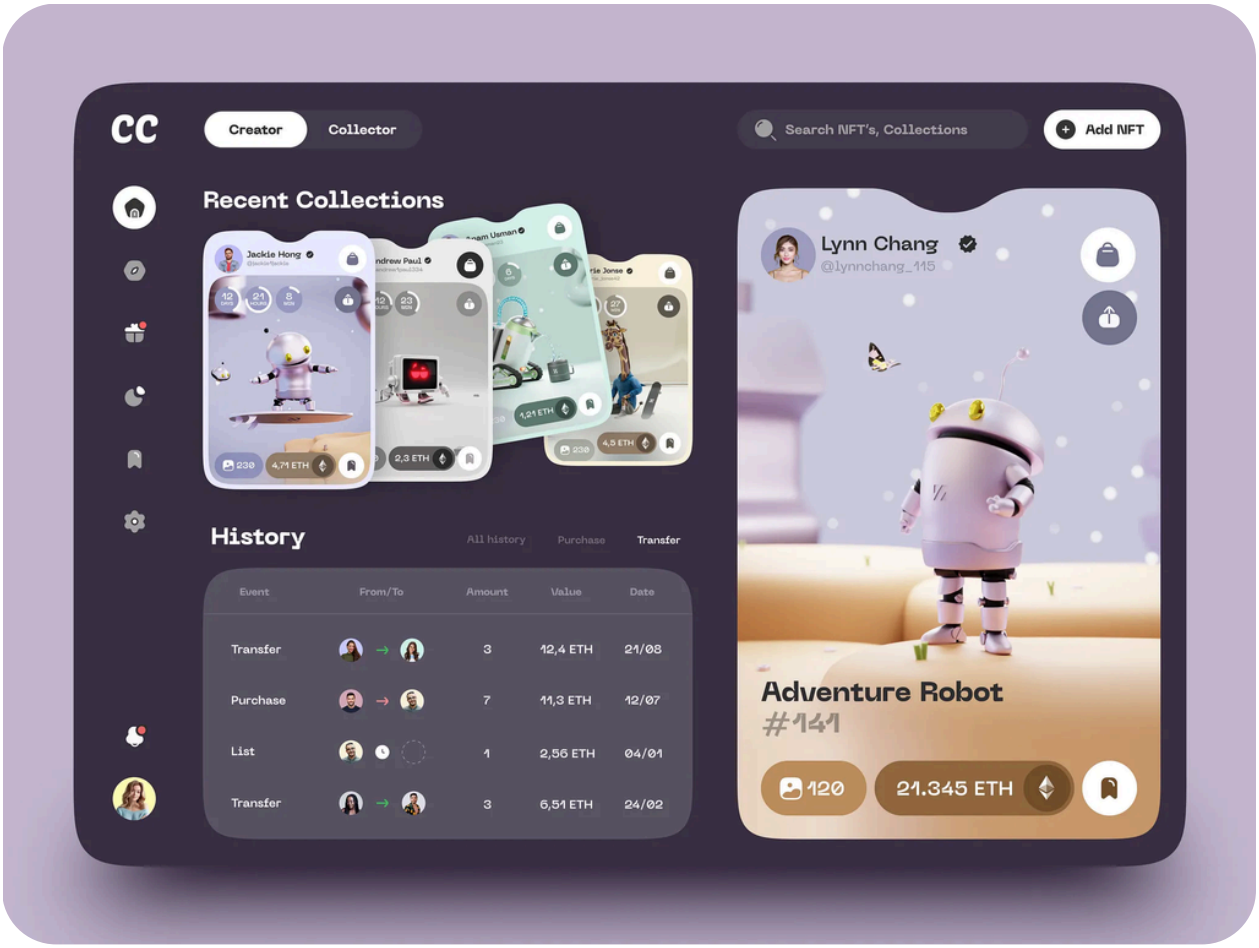
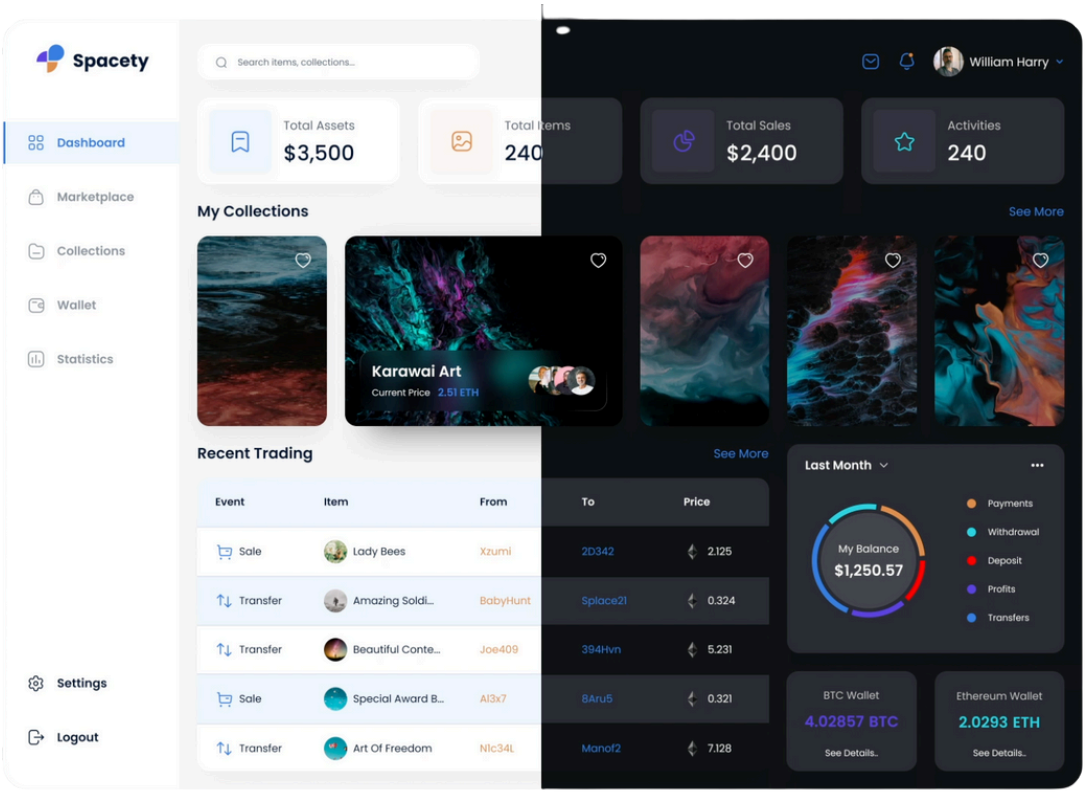
Gorgeous UI Dashboards



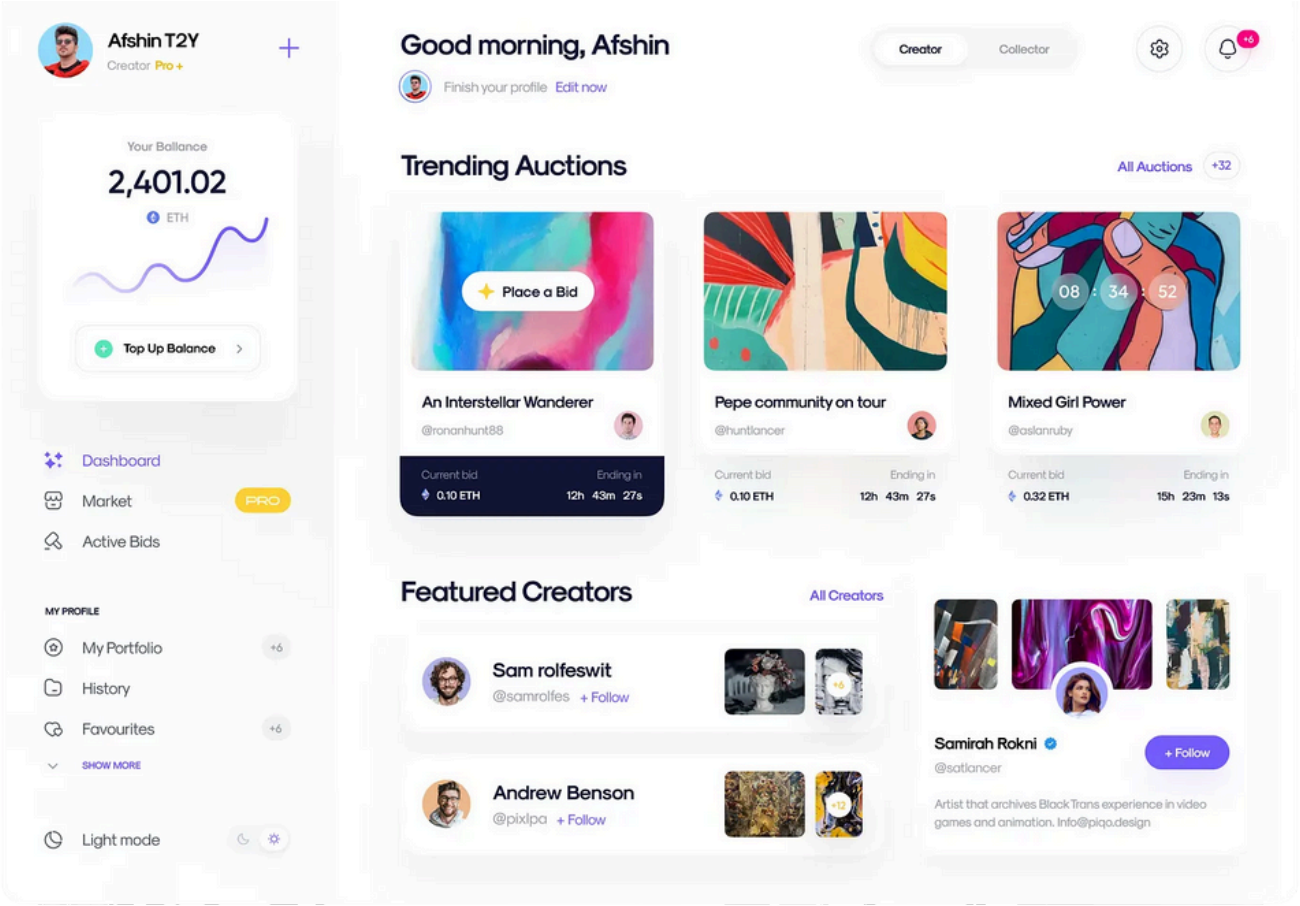
💡Inspiration



- https://dribbble.com/shots/23178378-Video-Sharing-Platform
- https://dribbble.com/shots/23200911-Mota-UX-UI-web-application-design-for-remote-work
- https://dribbble.com/shots/23188844-Sence-Point-HR-UX-UI-design
- https://dribbble.com/shots/23081011-Fitplan-Planner-Dashboard
- https://dribbble.com/shots/21198290-ValNFT-NFT-Dashboard-Concept
- https://dribbble.com/shots/18388554-Luval-NFT-Dashboard
- https://dribbble.com/shots/19801976-NFT-Dashboard-Manage-your-NFT-Collection
- https://dribbble.com/shots/17042125-NFT-Dashboard
- https://dribbble.com/shots/22419706-NFT-Dashboard
- https://dribbble.com/shots/18115126-Spacety-NFT-Dashboard
- https://dribbble.com/shots/20422049-Sportia-Sport-Soccer-Dashboard
- https://dribbble.com/shots/21235669-Merchant-dashboard-Overview-page-UI
- https://dribbble.com/shots/14413386-Business-analysis-dashboard
- https://dribbble.com/shots/16729003-Task-Management-Dashboard-Design
- https://dribbble.com/shots/17211535-Smartfarm-Dashboard-Design
- https://dribbble.com/shots/21567265-Parcel-Delivery-Admin-with-Custom-Illustrations
- https://dribbble.com/shots/22887468-E-learning-Dashboard
- https://dribbble.com/shots/21656734-Orelypay-Finance-Management-Dashboard
- https://dribbble.com/shots/14775845--Hoxye
- https://dribbble.com/shots/17138694-Vektora-Academy-Dashboard
- https://dribbble.com/shots/18468528-Cource-Productivity-Dashboard
- https://dribbble.com/shots/22615214-Productips-AI-Productive-Tracker
- https://dribbble.com/shots/22903820-Smart-Home-Dashboard
- https://dribbble.com/shots/17342291-Fintech-Dashboard
- https://dribbble.com/shots/20723362-Car-Dashboard-UI-SaaS
- https://dribbble.com/shots/22191383-Healthcare-Management-Dashboard
- https://dribbble.com/shots/22899045-Egghead-Shipping-tracking-order
- https://dribbble.com/shots/23123967-Bubble-POS-Point-Of-Sales-Saas-Admin-Dashboard
- https://dribbble.com/shots/22664473-SaaS-Project-Timeline
- https://dribbble.com/shots/15707372-Mac-Cleaning-app-dashboard

- https://dribbble.com/shots/20172082-Dashboard

💡 Metaprompt



Goal: Build a production-ready, senior-level **data dashboard web app** that is calm, clear, and fast. This is a **tool interface**, not a marketing page.

Required Tech Stack (Opinionated)

Use this stack unless impossible:

1. **Framework: Next.js 16** (App Router) + **React 19** + **TypeScript**. (Leveraging the stable React Compiler).
2. **AI Orchestration: Vercel AI SDK**. (Essential for streaming LLM responses, tool calling, and handling UI states for AI).
3. **Styling: Tailwind CSS v4.0**. (Using the high-performance Oxide engine and native container queries).
4. **Component System: shadcn/ui** (Radix UI Primitives).
5. **Data Layer: TanStack Query v5** (Client-side sync) + **Next.js use cache** (Server-side caching).
6. **Data Grid: TanStack Table v8**. (For complex logs, user lists, and analytics).

7. **Validation & Forms: Zod** + **React Hook Form**. (Unified validation for client inputs and Server Actions).
8. **Database/ORM: Drizzle ORM**. (Lighter and more "Edge-ready" than Prisma for 2025 serverless environments).
9. **Authentication: Clerk** or **Auth.js v5**. (Clerk for rapid RBAC deployment; Auth.js for self-hosted control).
10. **Security: OWASP Top 10 (2025)** + **Rate Limiting** (via Upstash/Redis for AI endpoints).
11. We will integrate the datasource from supabase

App Architecture Requirements

- Use a **single source of truth** for data (API/database). The UI reads from query cache, not random component state.
- Separate:
  - **Server state** (TanStack Query)
  - **UI state** (local component state)
  - **Form state** (React Hook Form)
- Use Next.js App Router patterns for layout:
  - /app/(dashboard)/layout.tsx with persistent sidebar
  - route-level loading/error boundaries
  - server components for initial data where appropriate, client components for interactivity. Next.js+2Next.js+2

Design Frameworks to Apply (Non-negotiable)

- **Information Architecture (IA):** Organize by user goals/decisions, not by features.
- **Cognitive Load Reduction:** Reduce visual noise; make scanning effortless.
- **Progressive Disclosure:** Default view is simple; advanced controls appear only when needed.
- **Perceived Performance:** UI should feel instant via optimistic updates, skeletons, and non-blocking interactions.

UI/UX Specifications (Senior Bar)

1) Layout & Hierarchy

- Strict grid; consistent spacing scale.
- Main content dominates; navigation is visually quiet.
- No oversized logos/banners. This is a tool.

2) Color & Token System

- Neutral base + **one accent** used only for primary actions/highlights.
- System colors:
  - red = error/destructive
  - green = success
- Contrast must be readable. Never use color as the only indicator.

3) Navigation

- Persistent left sidebar:
  - grouped links
  - clear active state
  - settings/logout at bottom
- Top bar only for global page actions + global search (optional).

4) Tables (Core Dashboard Utility)

Use TanStack Table features:

- Search + filters + sort
- Pagination (client or server)
- Row selection with bulk actions (selection reveals contextual toolbar)
- Column visibility + responsive columns tanstack.com+1

5) Charts (Keep them Functional)

- Only line and bar charts.
- Always include axes, labels, values, gridlines.
- Tooltips on hover.
- Choose chart approach:
  - Use **Recharts** for simple "business dashboards"
  - Use **ECharts** if dataset is large/high-frequency updates
    (Prefer functional clarity over fancy visuals.) LogRocket Blog+2strapi.io+2

6) Interaction Patterns (Radix-backed)

- **Popover** for small, non-blocking actions (display options, quick filters). radix-ui.com
- **Dialog/Modal** for complex or blocking flows (create/edit item). radix-ui.com
- **Toast notifications** for success/error/warning.
- **Optimistic UI** for common mutations:
  - immediate UI update, rollback on failure
  - use TanStack Query optimistic updates or React's useOptimistic pattern
    tanstack.com+1

7) States & Trust (Must be designed)

For every data region/component, implement:

- Loading (skeletons)
- Empty state (clear CTA)
- Error state (recoverable, retry)
- Success confirmation (toasts)
  Users should never wonder "did that work?"

Data Layer Requirements (Be Explicit)

Define:

- Data entities (e.g., Users, Projects, Links, Events, Metrics)
- Which endpoints power which cards/tables/charts
- Refresh strategy:
  - polling vs websocket vs manual refresh
- Caching rules:
  - stale time, refetch on focus, invalidation on mutation (TanStack Query)

Security & “Responsible App” Defaults

- Enforce RBAC/permissions server-side (not just UI hiding).
- Validate all inputs with Zod on server.
- Avoid exposing secrets to client.
- Add basic audit logging hooks for key actions (create/update/delete).
- Follow OWASP Top 10 mindset: secure defaults, least privilege, safe error handling. OWASP+1

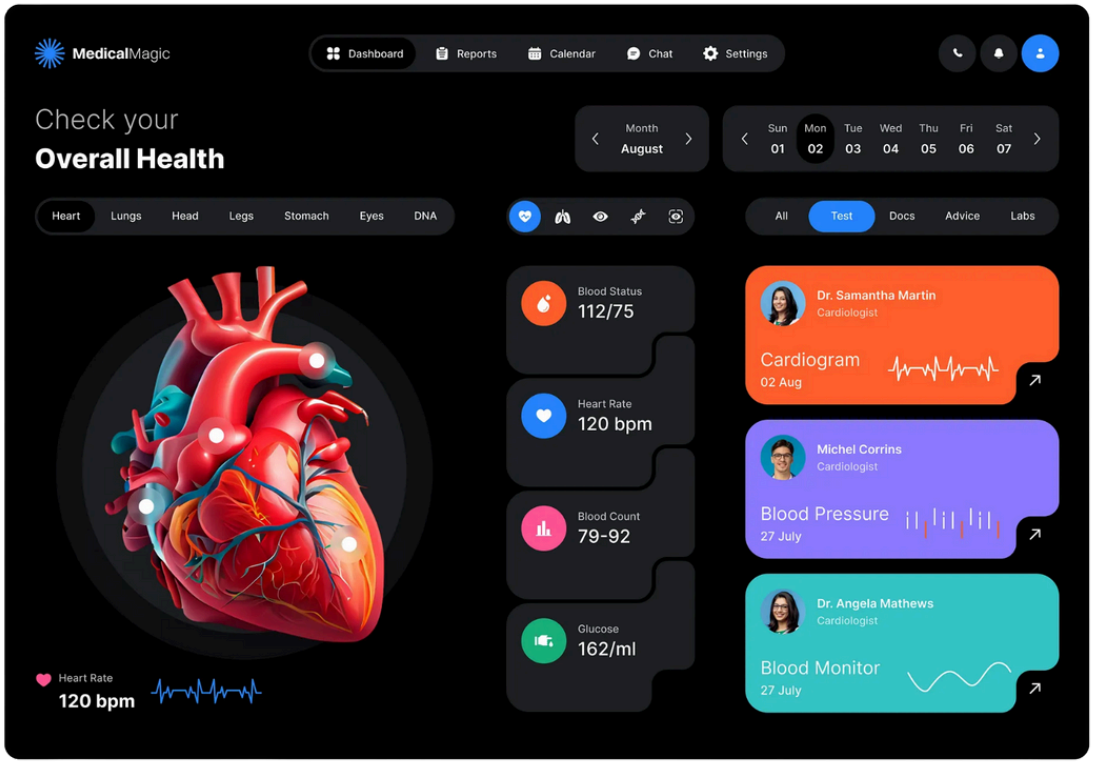
Deliverables (What you must output)

1. A working Next.js dashboard app scaffold:
   - routes, layout, sidebar, top actions
2. One “Dashboard Overview” page with:
   - KPI cards
   - a table with filtering/sorting/selection + bulk actions
   - a line chart + bar chart
3. A “Create/Edit” flow:
   - modal dialog form with validation + toast + optimistic update
4. Fully implemented loading/empty/error states
5. Clean, consistent component patterns and tokens

Final Quality Gate

- Understandable in **<10 seconds**
- Calm, professional, data-first
- Accessible keyboard navigation (Radix primitives help here) radix-ui.com+1
- Fast-feeling interactions (optimistic updates + good loading UX)

💡UI Focus, Navigation



You are a **senior product designer** reviewing and improving an existing application UI.
Your goal is to **strengthen hierarchy, focus, and navigation** so the design **disappears and the data becomes the hero**.

Core principle

The UI should point toward the data, not compete with it.

1. Primary focus

- Identify the **single most important insight or decision** this screen supports.
- Make that element **visually dominant on first glance**.
- All other elements must clearly support or defer to it.

2. Sidebar audit (critical)

Perform a **full review of all sidebars** (left, right, collapsible, contextual):

- Validate the **purpose** of each sidebar:
  - Is it global navigation, local navigation, utilities, or context?
  - If the purpose is unclear, recommend removal or consolidation.
- Reduce visual weight:
  - Lower contrast, lighter typography, minimal icon emphasis.
  - Sidebars should **frame the content**, not compete with it.
- Evaluate item priority:
  - Remove rarely used or redundant items.
  - Group related actions and enforce clear hierarchy.

    - Highlight *current location* subtly, not loudly.
- Check discoverability vs noise:
    - If something needs constant visibility, justify why.
    - Otherwise, recommend progressive disclosure or collapse.

3. Navigation discipline

- Clearly separate **global navigation** from **local, page-specific navigation**.
- Prevent navigation from pulling attention away from the data.
- Navigation exists to orient, not to sell or decorate.

4. Color & focus

- Use a **neutral base palette** across most UI surfaces.
- Apply **accent colors sparingly** to reinforce hierarchy or active focus.
- Reserve **system colors** (success, warning, error) strictly for state feedback.

5. Visual restraint

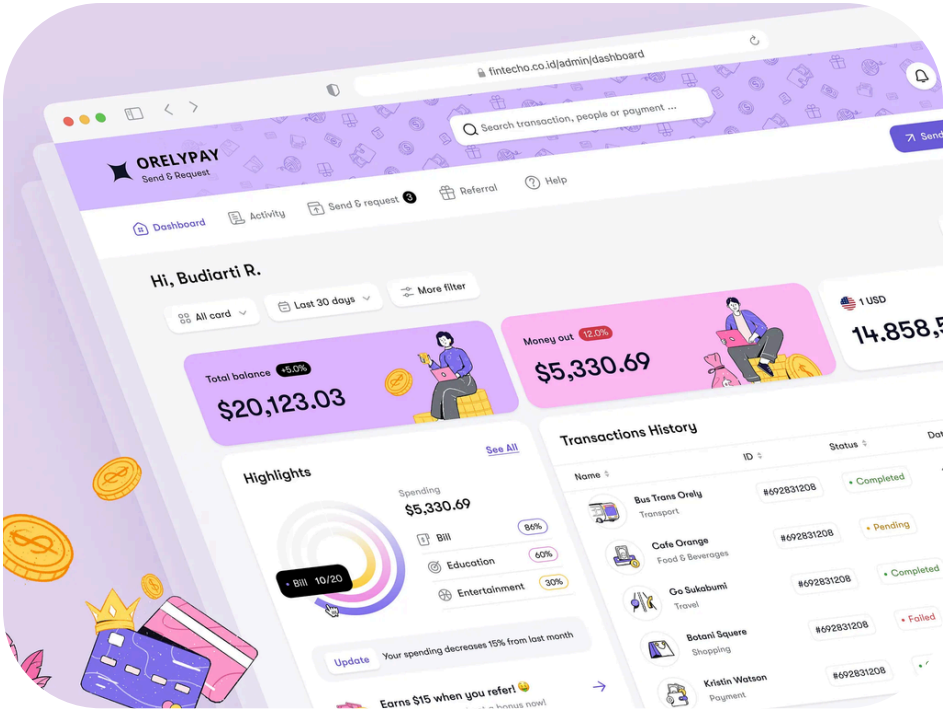
- Identify decorative or stylistic elements that do not improve understanding.
- De-emphasize secondary information through scale, contrast, and spacing.
- Avoid multiple competing focal points.

6. Outcome test

At a 3-second glance, the user should instantly know:

- what matters most
- where to look
- what action or insight comes next

Dynamics & Trust



You are a **senior product designer and UX engineer** reviewing an existing application.
Your goal is to **increase user trust** by improving **interaction clarity, feedback, and system reliability**.

Core principle

Trust is built through clear intent, immediate feedback, and consistent behavior.

1. Interaction intent

For every interactive element (filters, sorting, bulk actions, buttons):

- Identify the **user's intent** before the action.
- Ensure the interaction communicates:
  - *What will happen*
  - *When it will happen*
  - *Whether it can be undone*
- Flag any actions that feel ambiguous, surprising, or irreversible without warning.

2. Filters, sorting & bulk actions

- Ensure filters and sorting:
  - Clearly indicate when they are **active**
  - Show what data is being affected
  - Update results quickly and predictably
- Bulk actions must:
  - Confirm scope (what + how many items)
  - Prevent accidental destructive actions
  - Provide clear success or failure feedback

3. Modals vs popovers (intent matters)

- Use **modals** only for:
  - Blocking decisions
  - Destructive actions
  - Multi-step or high-commitment tasks
- Use **popovers / inline UI** for:
  - Quick edits
  - Previews
  - Low-risk actions
- Flag any misuse where interruption is too heavy or too light for the action's intent.

4. Feedback & system states

Audit all feedback mechanisms:

- Loading states:
  - Always acknowledge input immediately
  - Show progress if delays exceed a brief threshold
- Toasts and notifications:
  - Be concise and informative
  - Confirm outcomes, not just actions
  - Avoid stacking or flooding the user
- Error states:
  - Explain what went wrong
  - Explain what the user can do next
  - Never blame the user

5. Speed, consistency & reliability

- Interactions should feel:
  - Fast
  - Predictable
  - Consistent across screens
- Identify:
  - Delayed responses without feedback
  - Inconsistent behaviors for similar actions
  - UI states that feel "uncertain" or unstable

6. Trust test

After any interaction, the user should feel:

- "The system understood me"
- "The system responded clearly"
- "I can trust this to behave the same way next time"

If not, recommend changes.

Output format

- List **specific interaction improvements**.
- Explain **how each change increases trust**.
- Do **not** add new features — only refine interaction clarity, feedback, and consistency.