Inventory App — Codebase Overview Report

An approachable, in-depth explanation for early-stage software engineers.

Art Inventory App — Engineering Overview

A modern inventory app built with Next.js 15, React 18, TypeScript, Tailwind v4, and Supabase. It supports multi-project organization, secure authentication, photo uploads via Vercel Blob, and collaborative access management. The UI is component-driven using shadcn/ui and Radix primitives, with a project context managing cross-page state.

Tech Stack

- Frontend: Next.js 15 (App Router), React 18, TypeScript 5
- Styling: Tailwind CSS v4, shadcn/ui, Radix UI, lucide icons
- Backend: Supabase (PostgreSQL, Auth, Row Level Security)
- Storage: Vercel Blob for image uploads
- Email: Mailgun (with mock fallback)
- Deploy: Vercel, with custom CORS for /api routes

Project Structure (Folders)

High-level layout of the repository:

- app/: Routes (pages, layouts, API routes) using Next.js App Router
- components/: Reusable UI and feature components (inventory, projects, engagement)
- contexts/: Global React contexts (Project, Achievements)
- lib/: Supabase clients, utilities, engagement helpers
- hooks/: Custom hooks (e.g., useIsMobile, toast)
- scripts/: SQL migrations and small utilities
- public/: Static assets

Directory Tree (selected)

```
Ø=ÜÁ inventory-app
Ø=ÜÄ .DS_Store
Ø=ÜÁ .claude
  Ø=ÜÁ agents
    Ø=ÜÄ inventory-product-manager.md
    Ø=ÜÄ ui-product-optimizer.md
  Ø=ÜÄ settings.local.json
Ø=ÜÄ .env.example
Ø=ÜÄ .env.local
Ø=ÜÄ .env.vercel
Ø=ÜÄ .gitignore
Ø=ÜÄ CLAUDE.md
Ø=ÜÄ IMPROVEMENT_RECOMMENDATIONS_REPORT.md
Ø=ÜÄ README.md
```

```
Ø=ÜÄ TECHNICAL_DOCUMENTATION_FOR_JUNIORS.md
Ø=ÜÄ TECHNICAL_DOCUMENTATION_FOR_JUNIORS.pdf
Ø=ÜÁ app
  Ø=ÜÄ 1756815312471.jpg
  Ø=ÜÁ api
    Ø=ÜÁ delete-item
      Ø=ÜÄ route.ts
    Ø=ÜÁ delete-photo
      Ø=ÜÄ route.ts
    Ø=ÜÁ project-access
      Ø=ÜÁ notify
        Ø=ÜÄ route.ts
      Ø=ÜÄ route.ts
    Ø=ÜÁ projects
      Ø=ÜÁ [id]
        Ø=ÜÁ analytics
          Ø=ÜÄ route.ts
        Ø=ÜÁ areas
          Ø=ÜÄ route.ts
        Ø=ÜÁ export
          Ø=ÜÄ route.ts
        Ø=ÜÁ house-zones
          Ø=ÜÄ route.ts
        Ø=ÜÁ inventory-types
          Ø=ÜÄ route.ts
        Ø=ÜÁ member-count
          Ø=ÜÄ route.ts
        Ø=ÜÁ members
          Ø=ÜÁ me
            Ø=ÜÄ route.ts
          Ø=ÜÄ route.ts
        Ø=ÜÁ stats
          Ø=ÜÄ route.ts
      Ø=ÜÄ route.ts
    Ø=ÜÁ upload
      Ø=ÜÄ route.ts
  Ø=ÜÁ auth
    Ø=ÜÁ auth-code-error
      Ø=ÜÄ page.tsx
    Ø=ÜÁ callback
      Ø=ÜÄ route.ts
    Ø=ÜÁ login
      Ø=ÜÄ page.tsx
    Ø=ÜÁ sign-up
      Ø=ÜÄ page.tsx
    Ø=ÜÁ sign-up-invitation
    Ø=ÜÁ sign-up-success
    Ø=ÜÁ signout
      Ø=ÜÄ route.ts
  Ø=ÜÁ dashboard
    Ø=ÜÄ loading.tsx
    Ø=ÜÄ page.tsx
  Ø=ÜÄ globals.css
  Ø=ÜÁ inventory
    Ø=ÜÁ add
      Ø=ÜÄ page.tsx
```

Ø=ÜÁ edit
Ø=ÜÁ [id]
Ø=ÜÄ page.tsx
Ø=ÜÄ page.tsx
Ø=ÜÄ layout.tsx
Ø=ÜÄ not-found.tsx
Ø=ÜÄ page.tsx
Ø=ÜÁ profile
Ø=ÜÄ page.tsx
Ø=ÜÁ projects
Ø=ÜÁ [id]
Ø=ÜÁ access
Ø=ÜÄ page.tsx
Ø=ÜÁ areas
Ø=ÜÄ page.tsx
Ø=ÜÄ page.tsx
Ø=ÜÁ select-project
Ø=ÜÄ page.tsx
Ø=ÜÁ components
Ø=ÜÄ auth-guard.tsx
Ø=ÜÁ engagement
Ø=ÜÄ StreakCounter.tsx
Ø=ÜÄ TrophyShelf.tsx
Ø=ÜÄ WeeklyGoal.tsx
Ø=ÜÄ inventory-form.tsx
Ø=ÜÄ inventory-grid.tsx
Ø=ÜÄ inventory-search.tsx
Ø=ÜÄ inventory-stats.tsx
Ø=ÜÄ invitation-form.tsx
Ø=ÜÄ invitations-list.tsx
Ø=ÜÄ project-analytics.tsx
Ø=ÜÄ project-categories-manager.tsx
Ø=ÜÄ project-form.tsx
Ø=ÜÄ project-header.tsx
Ø=ÜÄ project-settings.tsx
Ø=ÜÄ project-switcher.tsx
Ø=ÜÄ projects-list.tsx
Ø=ÜÄ recent-items.tsx
Ø=ÜÄ theme-provider.tsx
Ø=ÜÁ ui
Ø=ÜÄ accordion.tsx
Ø=ÜÄ alert-dialog.tsx
Ø=ÜÄ alert.tsx
Ø=ÜÄ aspect-ratio.tsx
Ø=ÜÄ avatar.tsx
Ø=ÜÄ badge.tsx
Ø=ÜÄ breadcrumb.tsx
Ø=ÜÄ button.tsx
Ø=ÜÄ calendar.tsx
Ø=ÜÄ card.tsx
Ø=ÜÄ carousel.tsx
Ø=ÜÄ chart.tsx
Ø=ÜÄ checkbox.tsx
Ø=ÜÄ collapsible.tsx
Ø=ÜÄ command.tsx
Ø=ÜÄ context-menu.tsx

Ø=ÜÄ dialog.tsx
Ø=ÜÄ drawer.tsx
Ø=ÜÄ dropdown-menu.tsx
Ø=ÜÄ form.tsx
Ø=ÜÄ hover-card.tsx
Ø=ÜÄ input-otp.tsx
Ø=ÜÄ input.tsx
Ø=ÜÄ label.tsx
Ø=ÜÄ menubar.tsx
Ø=ÜÄ navigation-menu.tsx
Ø=ÜÄ pagination.tsx
Ø=ÜÄ popover.tsx
Ø=ÜÄ progress.tsx
Ø=ÜÄ radio-group.tsx
Ø=ÜÄ resizable.tsx
Ø=ÜÄ scroll-area.tsx
Ø=ÜÄ select.tsx
Ø=ÜÄ separator.tsx
Ø=ÜÄ sheet.tsx
Ø=ÜÄ sidebar.tsx
Ø=ÜÄ skeleton.tsx
Ø=ÜÄ slider.tsx
Ø=ÜÄ sonner.tsx
Ø=ÜÄ switch.tsx
Ø=ÜÄ table.tsx
Ø=ÜÄ tabs.tsx
Ø=ÜÄ textarea.tsx
Ø=ÜÄ toast.tsx
Ø=ÜÄ toaster.tsx
Ø=ÜÄ toggle-group.tsx
Ø=ÜÄ toggle.tsx
Ø=ÜÄ tooltip.tsx
Ø=ÜÄ components.json
Ø=ÜÁ contexts
Ø=ÜÄ AchievementsContext.tsx
Ø=ÜÄ ProjectContext.tsx
Ø=ÜÄ execute_migration.js
Ø=ÜÁ hooks
Ø=ÜÄ use-mobile.ts
Ø=ÜÄ use-toast.ts
Ø=ÜÁ lib
Ø=ÜÁ email-templates
Ø=ÜÄ invitation.tsx
Ø=ÜÁ engagement
Ø=ÜÄ confetti.ts
Ø=ÜÁ services
Ø=ÜÄ email.ts
Ø=ÜÁ supabase
Ø=ÜÄ api-client.ts
Ø=ÜÄ client.ts
Ø=ÜÄ server.ts
Ø=ÜÁ types
Ø=ÜÄ invitations.ts
Ø=ÜÄ projects.ts
Ø=ÜÁ utils
Ø=ÜÄ invitations.ts

```
  Ø=ÜÄ utils.ts
Ø=ÜÄ middleware.ts
Ø=ÜÄ next-env.d.ts
Ø=ÜÄ next.config.mjs
Ø=ÜÄ package-lock.json
Ø=ÜÄ package.json
Ø=ÜÄ postcss.config.mjs
Ø=ÜÁ public
  Ø=ÜÄ placeholder-logo.png
  Ø=ÜÄ placeholder-logo.svg
  Ø=ÜÄ placeholder-user.jpg
  Ø=ÜÄ placeholder.jpg
  Ø=ÜÄ placeholder.svg
Ø=ÜÁ scripts
  Ø=ÜÄ 001_create_inventory_schema.sql
  Ø=ÜÄ 002_create_profiles.sql
  Ø=ÜÄ 003_seed_sample_data.sql
  Ø=ÜÄ 004_create_projects_schema.sql
  Ø=ÜÄ 005_migrate_existing_data.sql
  Ø=ÜÄ 006_create_project_areas.sql
  Ø=ÜÄ 007_create_invitations_schema.sql
  Ø=ÜÄ 008_create_project_categories.sql
  Ø=ÜÄ 009_fix_missing_rls_policies.sql
  Ø=ÜÄ 010_remove_invitation_system.sql
  Ø=ÜÄ 011_fix_rls_circular_dependency.sql
  Ø=ÜÄ 012_create_pending_access_table.sql
  Ø=ÜÄ 013_fix_missing_profiles.sql
  Ø=ÜÄ 014_database_security_audit.sql
  Ø=ÜÄ check-db-structure.js
  Ø=ÜÄ fix-foreign-keys.sql
  Ø=ÜÄ generate-codebase-report.js
  Ø=ÜÄ monitor-logs.js
Ø=ÜÁ supabase
  Ø=ÜÁ .temp
    Ø=ÜÄ gotrue-version
    Ø=ÜÄ pooler-url
    Ø=ÜÄ postgres-version
    Ø=ÜÄ rest-version
Ø=ÜÄ tsconfig.json
Ø=ÜÄ vercel.json
```

Authentication, Authorization, and RLS

Authentication is handled by Supabase. The Next.js middleware creates a per-request Supabase SSR client, checks that the user is signed in, and enforces profile completion. For protected routes, it also verifies that the user belongs to at least one project and redirects accordingly.

- Database: Row Level Security policies safeguard tables, limiting access to a user's data or project membership.
- Project membership: Roles include owner, manager, member, viewer.
- API routes: Use a service-role client to perform server-side operations after validating the caller's membership by token.

Projects and State Management

The app is multi-tenant. A user may belong to several projects via project_members. ProjectContext loads the latest project, stores it as active, and offers a safe switch operation with upload-in-progress guards to avoid disrupting photo workflows.

- Active project drives queries (inventory items, categories, stats).
- Upload guard prevents reloading context during photo uploads to avoid losing state.
- Member counts and analytics are fetched through dedicated API routes.

Inventory Feature

The inventory feature covers listing, filtering, stats, add/edit flows, and deletion. Photos are uploaded to Vercel Blob. The UI is optimized for mobile, including Android/iOS quirks, and prevents accidental loss during uploads.

- List: InventoryGrid displays items with status, photos, location, and actions.
- Filter: InventorySearch supports free text, product type, location, status, and price range.
- Stats: InventoryStats summarizes counts and total values.
- Form: InventoryForm handles validation, upload, and persistence (insert/update) with robust logging.
- Delete: /api/delete-item validates auth + membership and deletes the item.

Photos and Uploads

Uploads are handled by /api/upload, which authenticates users by bearer token, validates file type and size, and stores images in Vercel Blob as public assets. The form implements device-aware workarounds and guards to ensure a smooth mobile experience.

API Routes (Highlights)

- GET/POST /api/projects: list and create projects (service-role, token required).
- GET /api/projects/[id]/stats: counts items and total estimated value.
- GET/POST/PUT/DELETE /api/projects/[id]/inventory-types and /house-zones: category management.
- GET /api/projects/[id]/export: export project data as JSON or CSV.
- POST /api/upload: upload photos to Vercel Blob.
- DELETE /api/delete-item: delete an inventory item (membership checked).

Database and Migrations

SQL scripts under scripts/ define core tables (inventory_items, projects, project_members, categories, pending access, profiles) and enforce RLS policies. Triggers keep updated_at fresh; indexes support common queries.

Engagement Features

A lightweight achievements system tracks user streaks, weekly goals, and trophies. Confetti and toasts celebrate key milestones. The dashboard shows a streak counter, weekly goal progress, and a trophy shelf (with reset for a specific project).

Configuration and Environment

- NEXT_PUBLIC_INVAPPSUPABASE_URL / NEXT_PUBLIC_INVAPPSUPABASE_ANON_KEY: Supabase client config.
- SUPABASE_SERVICE_ROLE_KEY: Server-side key for API routes (validate before privileged DB actions).
- BLOB_READ_WRITE_TOKEN: Vercel Blob uploads.
- MAILGUN_API_KEY / MAILGUN_DOMAIN: Emails (mock fallback when missing).
- NEXT_PUBLIC_APP_URL: Used in links and emails.

Security Considerations

- RLS on all tables; API routes validate bearer tokens and project membership.
- Middleware guards access to authenticated routes and project-required pages.
- CORS configured in dev (Next config) and prod (vercel.json) for /api routes.

Extensibility and Next Steps

- Server-side filtering/pagination for very large inventories.
- Stronger auth on /api/delete-photo (add membership checks).
- AI-assisted tagging and OCR for photos.
- Offline/PWA support with queued uploads and sync.

Key Files and What They Do

- package.json: Project metadata and dependencies (Next.js 15, React 18, Tailwind v4, shadcn/ui, Supabase).
- next.config.mjs: Next.js configuration: images, headers for dev CORS, external packages.
- vercel.json: Vercel function limits and production CORS headers for /api routes.
- middleware.ts: Request-time auth: creates Supabase SSR client, enforces profile and project checks, redirects unauthenticated users.
- app/layout.tsx: App shell: fonts, global styles, Theme + Project + Achievements providers, toaster.
- app/page.tsx: Landing page: redirects authenticated users to dashboard or project selection.
- app/dashboard/page.tsx: Dashboard UI with stats, engagement widgets (streak, weekly goal, trophies).
- app/inventory/page.tsx: Inventory screen: filters, stats, grid of items; now with “Volver al Dashboard” button.
- app/inventory/add/page.tsx: Add new item flow with InventoryForm.
- app/inventory/edit/[id]/page.tsx: Edit existing item; guards to avoid losing uploads.
- components/inventory-form.tsx: Form for creating/editing items with robust photo upload (Vercel Blob) and mobile guards.
- components/inventory-search.tsx: Search bar and filters (tipo, ubicación, estado, precio) hooked into list filtering.
- components/inventory-grid.tsx: Cards view of items with status badges, actions (editar, borrar).
- components/inventory-stats.tsx: Summary stats and values for the inventory list.
- components/project-header.tsx: Top header: project switch, profile menu, sign-out.
- contexts/ProjectContext.tsx: Global state: active project, switching, and upload-in-progress protection.

- contexts/AchievementsContext.tsx: Engagement: streaks, weekly goal, trophies, confetti, and reset function.
- lib/supabase/client.ts: Browser Supabase client (uses NEXT_PUBLIC_ vars).
- lib/supabase/server.ts: Server Supabase client (SSR + cookies).
- lib/supabase/api-client.ts: Service-role Supabase client for API routes (bypass RLS with checks).
- lib/engagement/confetti.ts: Lightweight confetti utility respecting reduced motion.
- app/api/upload/route.ts: Photo upload API: validates auth and file, uploads to Vercel Blob.
- app/api/delete-item/route.ts: Deletes item after verifying project membership.
- app/api/projects/*.ts: APIs for listing/creating projects and per-project analytics, categories, export.
- scripts/*.sql: Database migrations (inventory, projects, RLS, categories, etc.).