Budget Tracker – Project Flow Guide

Overview

This Next.js (App Router) project implements a mobile-first budget tracker. It uses client-side React components, a shared dashboard data provider for caching, and Supabase (via API helpers) for persistent data and file storage (avatars). The architecture favors quick navigation with minimal re-fetching and supports exporting reports to PDF.

Structure & Routing

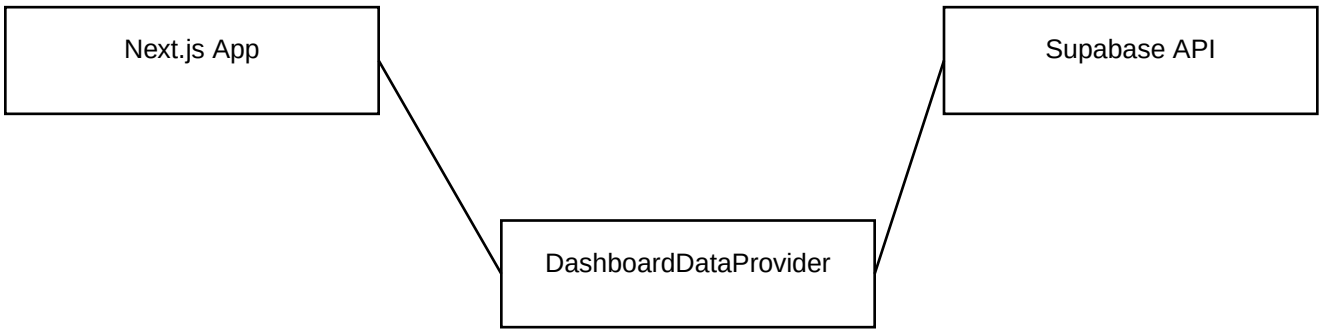
App directory structure under src/app uses the App Router. Private routes live under src/app/(private)/dashboard/* (dashboard, category pages). Shared UI and forms are in src/app/components/budget/*. Global layout is src/app/layout.js; private layouts gate auth.

Authentication

Auth helpers are provided in src/hooks/useAuth.js (session, user, signOut). Private pages gate on auth and show a loading overlay while session resolves. The dashboard reads effectiveUser and profile to render header state including avatar and initials.

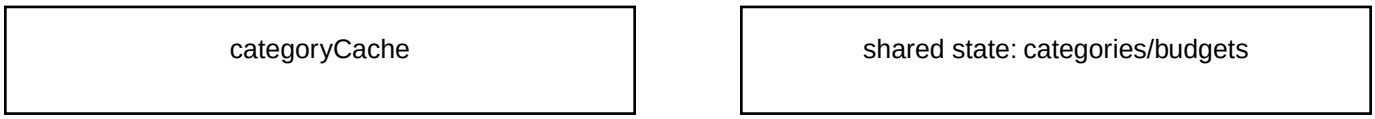
Data & API Layer

API helpers in src/api/db.js include categories (getUserCategories, addCategory), budgets (getBudgetForMonth, upsertBudget, getBudgetsForMonthBulk), expenses (listExpenses, addExpense, updateExpense, listRecentExpenses), notifications (listNotifications), and avatars (uploadAvatarDataUrl, getProfileForUser, getPublicAvatarUrl). Next.js API routes under src/app/api/* proxy to Supabase. Avatars are stored in the Supabase storage bucket “avatars”, with public URLs built by getPublicAvatarUrl.



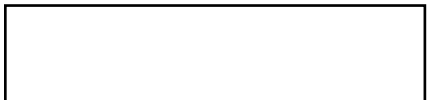
State & Caching

src/hooks/useDashboardData.js provides a co
transactions, notifications) and a per-category
pages. Pages hydrate from cache instantly an
avoid full re-renders.

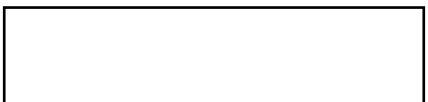


Rendering Flow

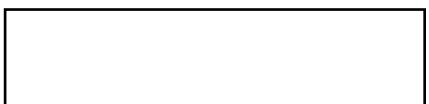
Dashboard shows a header with avatar; recent list uses Interse
implement incremental “load more” without visible loaders. Cat
infinite scroll in category lists also uses sentinel elements kept



List items



Invisible sentinel



Load more (on intersect)

Performance & Avoiding Re-ren

Avatar URLs avoid cache-busting, allowing b
re-renders by keeping shared lists and updat
pages reuse cached data via getCategoryDa

Security & Sanitization

Text inputs in ExpenseForm and numeric inp
sanitizeTextStrict strips HTML tags, normaliz
parses positive numbers and rejects invalid v
the database.

PDF Exports

src/lib/pdf.js uses jsPDF and jspdf-autotable
PNG data URLs (SVG capability included). F
correctly.

Key Files & API Examples

Key Files

• src/app/(private)/dashboard/page.jsx: Dashboard UI, header, avatar logic, recent
• src/app/(private)/dashboard/category/[slug]/page.jsx: Category UI, buying/labour forms, filters, infinite scroll
• src/app/components/budget/ExpenseForm.jsx: Add/Edit expenses, sanitized
• src/app/components/budget/BudgetForm.jsx: Set budgets, sanitized
• src/hooks/useDashboardData.js: Shared cache/state for data and category revisits
• src/api/db.js: API helpers and storage utilities
• src/lib/pdf.js: PDF generation utilities

API Examples

Examples of calling helpers from components:

```
import { listExpenses, addExpense } from "src/api/db"

async function loadExpenses(categorySlug) {
  const expenses = await listExpenses(categorySlug)
  return expenses
}

async function createExpense(payload) {
  const res = await addExpense(payload)
  // update cache and UI...
}
```

How Data Flows (Step-by-Step)

• User authenticates via useAuth; private layouts gate routes.
• Dashboard loads profile via getProfileForUser; avatar resolves via stable public URL.
• Provider initializes categories, budgets, recent transactions, notifications and caches them.
• Category page reads cache via getCategoryData(slug) to render immediately; background fetch merges updates via setCategoryData.
• Adding/editing an expense updates local UI and writes to API; provider updates recent list via addRecentExpense/updateRecentExpense.
• Infinite scroll increases visible count when sentinel intersects, avoiding visible loaders.

Screenshots

If you want real UI screenshots (Dashboard header, Category lists), I can capture them and embed them into this PDF. This version includes diagrams and the budgzyx.svg logo at top.

Notes

This guide summarizes the implementation and design choices to help students understand the flow, data management, and security posture of the project.