C4 Architecture (Levels 1–4) — Agentic Finance Summary

This document describes the architecture of the **Agentic Finance Summary** application using the **C4 model**.

- **Tech stack**: Next.js 14 (App Router), React 18, TypeScript, Tailwind, Recharts, Anthropic SDK
- **Primary use case**: Upload a transactions CSV → compute analysis/insights/actions → visualize results → optionally chat (streaming) about the computed results.

Level 1 — System Context

System

Agentic Finance Summary

A web application that helps a user understand their finances by uploading a bank transactions CSV, generating computed summaries, and providing a chat experience grounded in the computed output.

Primary actors

- **End user**
  - Uploads CSV
  - Views charts/tables/insights/actions
  - Asks questions using the chat panel

External systems

- **Anthropic API**
  - Used to power the streaming chat assistant
  - Authentication via `ANTHROPIC_API_KEY`

Context diagram (text)

- **User** → uses → **Agentic Finance Summary (Web App)**
- **Agentic Finance Summary** → calls → **Anthropic API** (only for chat)

Level 2 — Containers

At runtime, the system is composed of the following containers.

Container: Web UI (Browser)

- **Type**: Single-page UI rendered by Next.js / React
- **Responsibilities**:
  - File picker upload
  - Trigger analysis request
  - Render results (tables + charts)
  - Maintain chat state and stream assistant output
- **Key file**:
  - `src/app/page.tsx`

Container: Next.js App Server

- **Type**: Node.js server running Next.js (App Router + Route Handlers)
- **Responsibilities**:
  - Serves UI assets/routes
  - Provides API endpoints for analysis and chat
  - Performs CSV parsing + computation on the server
  - Streams chat responses back to the client
- **Key locations**:
  - UI routes: `src/app/*`
  - API routes: `src/app/api/*/route.ts`
  - Domain logic: `src/lib/*`

Container: Anthropic API

- **Type**: External SaaS API
- **Responsibilities**:
  - Produces assistant responses for the chat endpoint
- **Integration**:
  - `src/app/api/chat/route.ts`

Container diagram (text)

- **Browser (Web UI)** ⇄ HTTP ⇄ **Next.js App Server**
- **Next.js App Server** ⇄ HTTPS ⇄ **Anthropic API**

Level 3 — Components (inside the Next.js App Server)

App Router route components

- **`src/app/page.tsx`**
  - Client component (`"use client"`)
  - Owns UI state:
    - uploaded file
    - analysis result
    - chart/table rendering
    - chat messages and streaming updates
  - Calls server endpoints:
    - `POST /api/analyze`
    - `POST /api/chat`
    - `GET /api/env-check`
- **`src/app/layout.tsx`**
  - Root layout wrapper

API route handlers (Route Handlers)

- **`POST /api/analyze`** — `src/app/api/analyze/route.ts`
  - Accepts `multipart/form-data` with `file`

    - Reads CSV text
    - Calls `parseTransactionsFromCsv()`
    - Runs the `Orchestrator` to produce:
        - `analysis`
        - `insights`
        - `actions`
    - Returns JSON plus `warnings` and `counts`
- **`POST /api/chat`** — `src/app/api/chat/route.ts`
    - Requires `ANTHROPIC_API_KEY`
    - Uses Anthropic SDK with `stream: true`
    - Sends a prompt containing:
        - user message
        - JSON context (analysis/insights/actions)
    - Streams assistant tokens back as `text/plain`
- **`GET /api/env-check`** — `src/app/api/env-check/route.ts`
    - Returns booleans about `ANTHROPIC_API_KEY` presence/shape
    - Does **not** expose secrets

Domain modules (`src/lib/*`)

- **CSV parsing** — `src/lib/csv.ts`
    - Validates required headers
    - Parses money/date
    - Produces normalized `ClassifiedTransaction[]`
- **Orchestration** — `src/lib/orchestrator.ts`
    - Coordinates "agents":
        - Worksheet analysis agent
        - Insight agent
        - Action agent
- **Agents** — `src/lib/agents/*`
    - `worksheetAnalysisAgent.ts`
        - Totals (income/expense/net)
        - Monthly series
        - Major spending
        - 2025 inflows/outflows breakdowns
        - Event and Event Details aggregations
    - `insightAgent.ts`
        - Converts computed analysis into human-readable insights
    - `actionAgent.ts`
        - Produces action recommendations based on analysis + insights

Component diagram (text)

- `page.tsx` → calls → `/api/analyze` → uses → `csv.ts` → uses → `orchestrator.ts` → uses → `worksheetAnalysisAgent.ts`, `insightAgent.ts`, `actionAgent.ts`
- `page.tsx` → calls → `/api/chat` → uses → Anthropic SDK → calls → Anthropic API

Level 4 — Code (key classes/functions)

CSV parsing (server-side)

- **`parseTransactionsFromCsv(csvText: string)`** (`src/lib/csv.ts`)
  - Output:
    - `transactions: ClassifiedTransaction[]`
    - `errors: string[]`
  - Key behaviors:
    - Header validation against required schema
    - Normalization of debit/credit values to absolute numbers
    - Computes `amount = credit - debit`
    - Computes `direction` via `classifyDirection()`

Orchestrator (server-side)

- **`Orchestrator.run(transactions)`** (`src/lib/orchestrator.ts`)
  - Calls:
    - `WorksheetAnalysisAgent.run()` → `AnalysisResult`
    - `InsightAgent.run(analysis)` → `Insight[]`
    - `ActionAgent.run(analysis, insights)` → `ActionItem[]`

Worksheet analysis (server-side)

- **`WorksheetAnalysisAgent.run(transactions)`** (`src/lib/agents/worksheetAnalysisAgent.ts`)
  - Produces:
    - `totals`
    - `series.monthly`
    - `majorSpending`
    - `year2025` (optional) including:
      - `byCategory`
      - `byEvent`
      - `byEventDetails` (Event + Event Details)
    - `anomalies`

Chat streaming (server-side)

- **`POST(req)`** (`src/app/api/chat/route.ts`)
  - Reads:
    - `process.env.ANTHROPIC_API_KEY`
    - optional `process.env.ANTHROPIC_MODEL`
  - Uses Anthropic SDK `messages.create({ stream: true, ... })`
  - Streams text deltas to the browser using `ReadableStream<Uint8Array>`

UI composition (client-side)

- **`Home` component** (`src/app/page.tsx`)
  - `analyze()` :
    - posts CSV to `/api/analyze`
    - stores returned `result`
    - calls `/api/env-check` to show key warnings
  - `sendChat()` :
    - posts `{ message, context }` to `/api/chat`
    - reads `res.body.getReader()` and appends streamed chunks

Deployment / runtime notes

- Local dev: `npm run dev` (Next.js dev server)
- Production: `npm run build` + `npm run start`
- Secrets:
  - Keep `.env.local` out of git (project `.gitignore` ignores `.env*`).
  - Required for chat: `ANTHROPIC_API_KEY`