Next.js Code Layout Summary (fin-analytics)

This project is a **Next.js (App Router)** application. It uses **React client components** for the UI and **Route Handlers** under `src/app/api/*` for backend endpoints.

High-level structure

- `src/app/`
  - `page.tsx`
    - Main UI page (file upload, analysis display, charts/tables, and streaming chat UI)
  - `layout.tsx`
    - Root layout for the App Router
  - `globals.css`
    - Global styles
  - `api/`
    - `analyze/route.ts`
      - API endpoint that accepts a CSV file upload and returns analysis/insights/actions
    - `chat/route.ts`
      - Streaming chat endpoint backed by Anthropic Claude
    - `env-check/route.ts`
      - Safe endpoint to confirm `ANTHROPIC_API_KEY` is present (does not expose the secret)

- `src/lib/`
  - `csv.ts`
    - CSV parsing + normalization
    - Converts the uploaded CSV into `ClassifiedTransaction[]`
  - `types.ts`
    - Shared TypeScript types used by analysis, API routes, and UI
  - `orchestrator.ts`
    - Orchestration layer that runs multiple “agents” in sequence
  - `agents/`
    - `worksheetAnalysisAgent.ts`
      - Core financial computations (totals, monthly series, major spending, 2025 inflows/outflows)
    - `insightAgent.ts`
      - Derives human-readable insights from the analysis result
    - `actionAgent.ts`
      - Produces recommended actions based on analysis + insights

Request flow (from UI to results)

1. **User uploads CSV in** `src/app/page.tsx`.

2. UI calls `POST /api/analyze` with `FormData` containing `file`.
3. `src/app/api/analyze/route.ts`:
    - reads the file text
    - calls `parseTransactionsFromCsv()` in `src/lib/csv.ts`
    - runs the `Orchestrator` (`src/lib/orchestrator.ts`)
    - returns JSON:
        - `analysis`
        - `insights`
        - `actions`
        - `warnings` + `counts`
4. UI renders:
    - totals (income/expense/net)
    - monthly charts
    - 2025 “Major inflows & outflows” breakdowns
    - major spending tables

Streaming chat flow (Anthropic)

1. After an upload is analyzed, the UI enables the chat panel and calls `POST /api/chat`.
2. `src/app/api/chat/route.ts`:
    - reads `ANTHROPIC_API_KEY` (and optional `ANTHROPIC_MODEL`) from environment
    - sends a prompt to Claude including the computed context as JSON
    - streams tokens back to the browser using a `ReadableStream`
3. The UI reads the response stream and appends chunks to the last assistant message.

Where to change things

- **CSV column mapping / parsing:** `src/lib/csv.ts`
- **Financial calculations / aggregations:** `src/lib/agents/worksheetAnalysisAgent.ts`
- **Insights text:** `src/lib/agents/insightAgent.ts`
- **Recommended actions:** `src/lib/agents/actionAgent.ts`
- **UI layout and tables/charts:** `src/app/page.tsx`
- **Backend endpoints:** `src/app/api/*/route.ts`

Runtime notes

- API routes set `export const runtime = "nodejs";` so they run in the Node.js runtime.
- Environment variables:
    - `ANTHROPIC_API_KEY` is required for chat.
    - `ANTHROPIC_MODEL` is optional.