DataWeave Implementation Plan

Goal Description

Build "DataWeave", a premium, no-code ETL and Visualization platform. Users upload CSV files, and the system automatically parses, understands, transforms, and visualizes the data. The user can refine the data and dashboard before exporting.

User Review Required

IMPORTANT

Architecture Decision: This will be a **Client-Side First** architecture using **Next.js**.

- **Why?** Handling data (up to ~100MB) in the browser provides instant feedback and ensures data privacy (files aren't stored on a server).
- **Trade-off**: Very large files (>500MB) might lag on slower computers. *Is this acceptable?*

Proposed Changes

1. Technology Stack

- **Framework**: Next.js 14+ (App Router, TypeScript).
- **Styling**: Tailwind CSS + Shadcn/UI (for that "Premium" glassmorphism/dark mode look).
- **State Management**: Zustand (Global store for the active dataset and dashboard config).
- **Data Processing**:
    - PapaParse: Fast CSV parsing.
    - Danfo.js (Pandas equivalent for JS): DataFrame manipulation, statistics, transformations.
- **Visualization**: Recharts (Polished, animated charts) + Lucide React (Icons).

2. Core Architecture Modules

[NEW] Data Engine (/src/lib/data-engine)

The "Brain" of the application.

- **Ingestion**:

        - Drag-and-drop file zone.
        - Web Worker offloading (to keep UI responsive during parsing).
    - **Analyzer (analyzeDataset)**:
        - **Type Inference**: Detects if a column is Date, Categorical, Numeric, or Text.
        - **Quality Check**: Counts missing values, detects outliers.
        - **Stats**: Min, Max, Mean, Median, StdDev for numeric columns.
    - **Transformer (transformData)**:
        - **Auto-Feature Engineering**:
            - *Dates*: Extract Year, Month, Day, DayOfWeek.
            - *Numeric*: Auto-binning (e.g., Age -> Age Groups).
        - **User Actions**: Filter, Rename, Drop, Sort.

[NEW] Visualization Engine (/src/lib/vis-engine)

The "artist" making the dashboard.

- **Heuristic Mapper**: Decides the best chart based on selected columns.
    - *1 Cat Variable* -> Bar Count or Pie.
    - *1 Num Variable* -> Histogram/Box Plot.
    - *1 Cat + 1 Num* -> Bar Chart (Aggregated).
    - *2 Num* -> Scatter Plot.
    - *Time + Num* -> Line Chart.
- **Chart Factory**: A dynamic component that renders the appropriate Recharts component based on config.

[NEW] Dashboard Interface (/src/components/dashboard)

- **Layout**: CSS Grid / Masonry layout for charts.
- **Interactivity**:
    - "Edit Graph" modal (change type, colors, axes).
    - "Data Table" view (spreadsheet-like editable view).

- **Export**:
    - html2canvas for taking screenshots of the dashboard.
    - CSV serializer for downloading the transformed dataset.

3. Step-by-Step Implementation Flow

1. **Project Initialization**: Setup Next.js, Tailwind, Font (Inter/Outfit), and Theme Provider.
2. **Core UI**: Build the Shell (Navbar, Sidebar) and Landing Page (Upload Zone).
3. **Data Logic**: Implement Parsing & Basic Analysis (view raw data in a table).
4. **Transformation Layer**: Add "Smart Columns" logic.
5. **Vis Layer**: Implement the Chart Factory and Auto-Mapper.
6. **Dashboard Assembly**: Put it all together in a draggable grid.
7. **Export & Polish**: Add download buttons and refine animations.

Verification Plan

Automated Tests

- **Unit Tests**:
    - analyzeDataset: Verify it correctly identifies Date vs String.
    - transformData: Verify "add column" logic works accurately.
- **Component Tests**:
    - Verify Chart component does not crash with empty data.

Manual Verification

- **Test Case 1 (Sales Data)**: Upload a standard sales CSV. Check if "Date" is recognized and a Line Chart of "Sales over Time" is suggested.
- **Test Case 2 (Messy Data)**: Upload CSV with missing values. Ensure app doesn't crash and suggests handling them.
- **Test Case 3 (Export)**: Download the modified CSV and verify it opens correctly in Excel.