**Data Jaw: PoC Implementation - Core Workflow Capture & Visualization**

**1. Context & Goal (The Why)**

**Product Vision**

Data Jaw eliminates the frustration of repeatedly rebuilding information workflows. Our core promise is "Show us once, never do it again." We automate recurring tasks by observing user demonstrations.

**Core Problem We Solve**

We target "workflow repeaters" (freelancers, small business owners, productivity enthusiasts) who lose 4-6 hours weekly to painful inefficiencies:

* The "setup tax": Starting similar tasks from scratch every time.
* The "change cascade": Manually updating multiple documents when source info changes.
* The "format fight": Struggling to maintain consistent appearance in outputs.
* The "technical barrier": Finding automation tools too complex or requiring coding skills.

**Data Jaw Solution**

We observe your workflow once (via screen recording + narration), create persistent connections between your data sources and outputs, and automatically update everything when sources change, maintaining your formatting – no coding needed.

**POC Details**

We're creating a proof-of-concept (POC) for Data Jaw's core value proposition: "Show us once, never do it again." This POC will demonstrate our ability to:

1. Capture a user's workflow through screen recording with audio narration
2. Process and understand this multimodal input
3. Transform it into a structured workflow representation
4. Present it back to the user as an interactive block-based diagram

**Why This Matters**

Our target users (freelancers, small business owners, and productivity enthusiasts) waste 4-6 hours weekly rebuilding similar workflows. This POC validates our ability to solve this pain point by demonstrating that we can:

* Accurately capture and understand user workflows from a single demonstration
* Convert unstructured demonstrations into structured workflows
* Present workflows in an intuitive, editable format

The success of this POC will validate our core technical hypothesis and set the foundation for the full product.

**2. Technical Architecture Overview**

The POC consists of five key components:

1. **Landing Page & Workflow Recorder UI**: Introduces the product and guides users through recording their workflow
2. **Video Processing Pipeline**: Handles uploaded recordings and prepares them for AI analysis
3. **Multimodal Analysis Pipeline (Gemini 2.0 Flash & Claude 3.5)**: Extracts raw workflow information from the recording
4. **Block Structure Generation (Claude 3.5)**: Converts workflow information into a block-based representation
5. **Interactive Workflow Viewer**: Displays the generated workflow to the user for validation and editing

**3. Implementation Details**

**3.1 Landing Page & User Flow**

**Landing Page Requirements**

* Create a visually appealing, simple landing page that clearly communicates our value proposition
* Headline: "Show us once, never do it again"
* Subheadline: Focus on the problem of repeated workflows for our target segment
* Three-step process visualization: Record → Process → Automate
* Prominent "Get Started" button

**Recording Instructions Page**

* Clear, step-by-step instructions for recording workflow with audio
* Focus on macOS/QuickTime for this POC
* Instructions should include:
  + Opening QuickTime Player and selecting "New Screen Recording"
  + Selecting the microphone from the dropdown menu
  + Recording only the relevant applications for the workflow
  + Speaking clearly to explain actions as they're performed
  + Keeping recordings under 5 minutes for optimal processing
  + Saving the recording when complete

**Video Upload Component**

* Drag-and-drop interface for file upload
* Support for .mov (QuickTime) format
* File validation checks:
  + Format validation (QuickTime .mov)
  + Size validation (max 300MB)
  + Duration validation (max 5 minutes)
* Feedback on validation issues
* Upload progress indicator

**3.2 Video Processing Pipeline**

**Backend Processing Service**

* Receive uploaded video files securely
* Store videos in a secure, temporary cloud storage location
* Perform basic preprocessing:
  + Convert to standard format if needed (WebM or MP4)
  + Ensure audio track is properly extracted
  + Validate video quality is sufficient for processing
* Generate unique workflow ID for tracking through system
* Create processing job and place in queue

**3.3 Multimodal Analysis Pipeline (**gemini-2.0-flash & claude 2.5 sonnet**)**

This pipeline uses gemini-2.0-flash & claude 2.5 sonnet to extract raw workflow information from the recording in two distinct steps:

**Step 1: Raw Workflow Extraction**

* **Input**: Processed video file with audio
* **Process**:
  + Frame extraction at regular intervals
  + Process frames to identify visible applications, UI elements, and user actions
  + Transcribe audio narration with timestamps
  + Pass frames, transcription, and timing information to gemini-2.0-flash
* gemini-2.0-flash **System Prompt for Raw Extraction**:

You are a specialized workflow analysis system designed to extract detailed information from screen recording videos with audio narration. Your task is to create a comprehensive transcript of everything occurring in this workflow recording.

For each significant moment in the workflow:

1. Document the timestamp

2. Describe what is visible on screen (applications, documents, interfaces)

3. Note user actions (clicks, typing, file operations)

4. Include the user's verbal explanation

5. Identify transitions between applications

Focus on creating a factual, chronological record of everything observed without interpretation. Document every detail systematically, as this will be used to reconstruct the workflow later.

* **Output**: Structured chronological transcript of the workflow with timestamps, screen states, actions, and narration

**Step 2: Workflow Organization**

* **Input**: Raw extraction output from Step 1
* **Process**: Pass the raw extraction to claude 2.5 sonnet with a system prompt focused on organization
* claude 2.5 sonnet **System Prompt for Organization**:

You are a workflow organization expert. Your task is to analyze the raw workflow transcript and organize it into a coherent, logical structure.

Specifically, you should:

1. Identify distinct workflow steps and number them sequentially

2. For each step, summarize:

- The primary action being performed

- The application(s) being used

- The input data and its source

- The output data and its destination

- Any key considerations mentioned in the audio narration

3. Identify patterns or repetitions in the workflow

4. Note conditional logic or decision points

5. Identify potential triggers for this workflow

6. Determine how often this workflow would likely be repeated

Structure your response as a clear, organized JSON object that represents the logical flow of the workflow.

* **Output**: Structured JSON representing the organized workflow steps, applications, data flows, and considerations

**3.4 Block Structure Generation (Claude 3.5)**

* **Input**: Organized workflow from claude 2.5 sonnet (Step 2 output)
* **Process**: Pass the organized workflow to Claude 3.5 sonnet to generate a block-based representation
* **Claude 3.5 sonnet System Prompt**:

You are a specialized system for Data Jaw that translates organized workflow information into a formal block-based structure. Your expertise is in understanding workflow patterns and representing them using Data Jaw's block architecture.

In Data Jaw's architecture:

- Blocks are the fundamental units (Document Blocks, Presentation Blocks, Interface Blocks, Data Blocks)

- Sources are external data attachments (Files, Web Content, API Data, Manual Input)

- Templates define structure and formatting

- Connections show information flow between blocks

Given the organized workflow information provided, create a complete block-based representation with:

1. Blocks for each logical step, including:

- id: unique identifier

- type: block type based on purpose

- title: clear, descriptive name

- description: what this block does

- properties: formatting, inputs, and outputs

2. Sources for each external data input, including:

- id: unique identifier

- type: source type

- location: where the data comes from

- updateRules: how often it changes

3. Connections between blocks, including:

- sourceBlockId: origin block

- targetBlockId: destination block

- dataType: what information flows

- updateRules: how changes propagate

Generate a complete, valid JSON structure that follows this architecture precisely. The result should be directly usable by the Data Jaw visualization engine.

* **Output**: Formal block-based workflow JSON structure following Data Jaw's architecture

**3.5 Interactive Workflow Viewer**

* **Input**: Block-based workflow structure from Claude 3.5 sonnet
* **Features**:
  + Visual rendering of blocks, sources, and connections
  + Color-coding by block type
  + Zoom and pan navigation for larger workflows
  + Click interaction to view block details
  + Simple editing capabilities:
    - Edit block titles and descriptions
    - Edit connection properties
    - Add/remove connections
    - Rearrange blocks
  + "Run Workflow" button (simulated for POC)
  + "Save Workflow" button
  + Export options (PDF, PNG)
* **UI Components**:
  + Main canvas for workflow visualization
  + Properties panel for selected elements
  + Toolbar with common actions
  + Minimap for navigation of complex workflows

**3.6 Processing Status UI**

* Create a clear, informative processing status page that shows:
  + Video processing status
  + Gemini 2.0 flash Raw Extraction progress
  + Claude 3.5 Organization progress
  + Claude 3.5 Block Structure Generation progress
  + Overall completion percentage
  + Estimated time remaining
  + Error handling with retry options

**4. Technical Implementation Guidelines**

**Frontend Technologies**

* **Framework**: React with TypeScript
* **State Management**: React Context API or Redux
* **Styling**: Tailwind CSS
* **Component Library**: Headless UI components
* **Diagram Rendering**: React Flow for interactive workflow diagrams
* **Video Processing**: Use browser APIs for initial validation

**Backend Technologies**

* **API**: Node.js with Express
* **Video Processing**: FFmpeg for video manipulation
* **Storage**: AWS S3 for temporary video storage
* **AI Integration**: Google Gemini API (gemini 2.0 flash) and Anthropic API (Claude 3.5 sonnet)
* **Authentication**: Simple JWT for the POC
* **Deployment**: Docker containers on AWS ECS

**API Endpoints**

1. /api/upload - Handles video upload and initial processing
2. /api/workflow/extract - Triggers gemini 2.0 flash raw extraction
3. /api/workflow/organize - Triggers claude 3.5 sonnet workflow organization
4. /api/workflow/generate-blocks - Triggers Claude 3.75 sonnetblock structure generation
5. /api/workflow/{id} - Returns current workflow state
6. /api/workflow/{id}/status - Returns processing status
7. /api/workflow/{id}/update - Handles workflow edits