

Software Requirements Specification (SRS): Teach by Doing (TbD) MVP - V1.0

Status: Approved for Build **Author:** Greg Burns **Date:** November 21, 2025 **Related Documents:** [Link to TDD]

1. Introduction

1.1 Purpose

The purpose of this document is to define the functional and non-functional requirements for the **Teach by Doing (TbD) MVP V1**. This system serves as the "Data Ingestion Engine" for the **Pathways as Data (PAD)** initiative.

1.2 Scope

The V1 MVP is a **local, standalone prototype**. Its primary function is to ingest a screen recording of a human expert performing a digital task and automatically convert that video into a structured, machine-readable **Pathway .json** file.

Out of Scope for V1:

- Real-time streaming (batch processing only).
- Complex semantic understanding (LLM integration is V2).
- Cloud deployment (Local Docker execution for V1).

1.3 Strategic Context (The "Bridge")

To de-risk the project against delays in the "TimeSuite" VCU dependency, this V1 utilizes a **"Bridge Stack"** of open-source Computer Vision tools (**OpenCV**, **Tesseract**, **PySceneDetect**) to achieve immediate functionality for internal data labeling.

2. Functional Requirements

2.1 Video Ingestion (The Input)

- **FR-01:** The system **shall** accept video files in `.mp4` format via a defined API endpoint or file path.
- **FR-02:** The system **shall** validate that the input file contains a video stream readable by `OpenCV`.
- **FR-03:** The system **shall** support standard screen resolutions (1920x1080 preferred) and frame rates (30fps minimum).

2.2 Event Segmentation (The "When")

- **FR-04:** The system **shall** analyze the video stream to detect distinct "Action Events" (e.g., moving from one screen state to another).
- **FR-05:** The system **shall** use visual differencing algorithms (via `PySceneDetect`) to identify scene changes with a configurable sensitivity threshold (default: 30% pixel change).
- **FR-06:** The system **shall** generate a timestamped list of segments, defined by `start_time` and `end_time` for each stable state.

2.3 Visual Data Extraction (The "What")

- **FR-07:** For each detected segment, the system **shall** extract a representative "Key Frame" (defined as the middle frame of the stable state).
- **FR-08:** The system **shall** perform Optical Character Recognition (OCR) on the Key Frame using `Tesseract 5` to extract visible UI text.
- **FR-09:** The system **shall** identify the "Active Region" of interaction by comparing the Key Frame to the previous segment's frame to isolate the specific UI element that changed (e.g., a highlighted button or opened menu).

2.4 Pathway Generation (The Output)

- **FR-10:** The system **shall** map each detected segment to a standard `ActionNode` object.
- **FR-11:** Each `ActionNode` **shall** contain:
 - `timestamp_start` / `timestamp_end`
 - `ui_element_text` (derived from OCR of the Active Region)
 - `confidence_score` (derived from OCR confidence)
- **FR-12:** The system **shall** compile these nodes into a sequential `Pathway` object.
- **FR-13:** The system **shall** export the final `Pathway` object as a JSON file conforming to the **PAD Schema v0.1** (defined in TDD Section 2).

3. Non-Functional Requirements (NFRs)

3.1 Performance & Latency

- **NFR-01:** The parsing pipeline **shall** process a video in no more than **2x** the video's duration (e.g., a 1-minute video processes in <2 minutes) on a standard CPU-only developer machine.
- **NFR-02:** The system **shall** utilize frame-skipping (processing 1fps) to optimize compute resources for the MVP.

3.2 Deployment & Portability

- **NFR-03:** The entire system **shall** be containerized using **Docker**.
- **NFR-04:** The Docker container **shall** include all OS-level dependencies (**libglib11**, **tesseract-ocr**) to ensure it can run on any machine (Windows/Mac/Linux) without environment configuration.

3.3 Data Privacy

- **NFR-05:** All video processing **shall** occur locally within the container. No video data **shall** be transmitted to external third-party APIs for V1.
-

4. Interface Requirements

4.1 API Interface

- **IR-01:** The system **shall** expose a RESTful API via **FastAPI**.
- **IR-02:** The API **shall** provide a **POST /process** endpoint that accepts a video file and returns the **Pathway.json** payload.

4.2 User Interface (MVP)

- **IR-03:** For demonstration purposes, the system **shall** provide a simple command-line interface (CLI) or basic HTML upload form to trigger the processing.