ChatGPT

AI-Generated Short-Form Video Platform Specification

This document outlines a detailed design for a SaaS pipeline that transforms uploaded MP4 videos into polished short-form clips (e.g. YouTube Shorts, Instagram Reels) via an AI-driven, human-in-the-loop workflow. The system uses a Next.js frontend on Vercel, Remotion for video rendering, and n8n for backend orchestration. Claude (or a similar LLM) provides code and script generation, while specialized APIs (Whisper, Google Video Intelligence, ElevenLabs, HeyGen/D-ID, Runway, etc.) handle content analysis and asset creation. Each stage is modular and swap-friendly, and the platform supports logging, user feedback, and iterative improvement [1] [2].

Agents & Roles

- **Script Writer:** An AI agent (using Claude/GPT) that ingests scene transcripts, highlights, and any user prompt, then composes a concise narration script for the short video. This mirrors existing pipelines where an LLM generates voiceover text from content (e.g. using OpenAI to draft a voiceover script) [2]. The script writer can refine language style per user preferences (e.g. promotional tone, educational, humorous).

- **Audio Engineer (TTS Specialist):** Takes the approved script and calls a text-to-speech API (ElevenLabs) to synthesize a natural-sounding voiceover. ElevenLabs' API offers ultra-realistic, expressive voices in 30+ languages [3]. The agent selects a voice profile (or clones a voice) and configures parameters (emotion tags, pacing). For example, as in the n8n template, "ElevenLabs [is used] to generate voiceover audio from the script" [2].

- **Avatar Generator (Digital Presenter):** If an on-screen avatar/narrator is desired, this agent uses an AI avatar service. For instance, HeyGen or D-ID can turn a text script and optional portrait into a talking-head video. HeyGen's API "instantly create[s] professional AI video avatars from text, scripts, or uploaded images" [4]. Similarly, D-ID's API accepts a face image plus the TTS audio and generates a realistic talking-head video of that avatar speaking the script [5] [6].

- **Video Editor (Assembler):** Using Remotion (a React-based video framework), this agent programmatically constructs the video timeline. It ingests segments (avatar video, stock B-roll, graphics, etc.), overlays captions/typography, and applies transitions. As noted in tutorials, "Remotion for video rendering [and] Next.js... build a flexible video editing tool" [7], and tools like the Short Video Maker use Remotion to "assemble all the elements — audio, captions, background video — into a coherent timeline and render the final MP4 file" [8].

- **Subtitle Patcher:** Uses Whisper (or another ASR) to generate time-aligned captions from the audio, then reviews/fixes them. For example, Whisper.cpp (OpenAI's speech-to-text) is leveraged for "perfectly timed, engaging captions" in similar pipelines [9]. The agent checks alignment and correctness, ensuring caption text matches the spoken audio.

- **Typography & Style Curator:** Ensures visual elements (fonts, colors, animation styles) are consistent with branding or content type. This agent may apply stylized title cards, highlight quotes/tweets, or design motion graphics (e.g. “mini-lists” or infographic overlays). These design choices can be templated in Remotion or generated via graphic APIs. The role ensures the final video has professional polish (e.g. applying animated lower-thirds, slide-in text, etc.).

Each agent corresponds to one or more steps in the workflow and can be implemented as separate modules or n8n sub-flows. The system’s modular design allows swapping tools per role (e.g. replace HeyGen with D-ID for avatars) with minimal reconfiguration [1].

Workflow Stages

1. **Upload & Ingestion:** A Next.js frontend lets users upload an MP4 file. On upload completion (or via a webhook), an n8n workflow is triggered to store the video (e.g. in S3 or a database) and begin processing. Metadata (filename, user ID, timestamps) is logged.

2. **Scene & Audio Extraction:** The workflow calls a video analysis API to find key scenes. For example, Google Cloud Video Intelligence can detect shot changes and label frames (“extract rich metadata at the video, shot, or frame level” [10]). Simultaneously, the audio track is extracted. The audio is sent to Whisper (or Google Speech-to-Text) for transcription. Whisper.cpp, a CPU-efficient port of OpenAI’s Whisper, is commonly used to generate time-aligned captions [11] [12]. The result is a transcript with timecodes and highlighted keywords or detected topics from each scene.

3. **Script Generation:** Using the scene highlights and any user prompt (e.g. “summarize main points”), Claude generates a short video script. This step is akin to existing workflows where an LLM creates narration from content; for instance, the n8n template “Uses OpenAI to generate a voiceover script based on the captions” [2]. The agent’s output is a structured script (a few concise paragraphs or bullet points) describing the video’s flow.

4. **Voiceover Synthesis:** The approved script is sent to the ElevenLabs TTS API. ElevenLabs converts text to speech with expressive nuance, supporting dozens of languages and accents [3]. The workflow retrieves the resulting audio file (e.g. MP3 or WAV). This voiceover audio is what the final video will feature. (At this point, an avatar video could optionally be generated in parallel.)

5. **Avatar/Narrator Generation (Optional):** If a talking avatar is desired, the workflow calls an AI avatar API. For example, HeyGen can “instantly create lifelike AI video avatars from text, scripts, or uploaded images” [4]. Alternatively, D-ID’s Generative AI API takes a single face image and the TTS audio to produce a realistic talking-head video [5] [6]. The agent specifies the script audio and chosen avatar, and retrieves the rendered avatar video segment.

6. **Visual Asset Generation:** For creative visuals or B-roll, the system may use Runway or stock footage. Runway’s API provides generative models (e.g. Gen-3/4) accessible via REST [13]. For instance, one could use a text-to-video model or image-to-video model to create dynamic backgrounds. The workflow might also fetch images (via Unsplash/Pexels) or query Runway’s text-to-image/video endpoint for abstract motion visuals tied to the script content.

7. **Video Assembly (Remotion):** The Video Editor agent now composes the final video. Remotion (a React framework) is used to programmatically combine segments: it imports the avatar video (or static host image), overlaid audio, and any generated B-roll. Captions from Whisper are added as timed subtitles. Additional graphics (e.g. highlighted tweets/documents) are placed as picture-in-picture overlays. Animated transitions and typographic effects (e.g. bullet list animations, text wipes) are coded in the Remotion scene logic. This mirrors how the Short Video Maker project uses Remotion to render "the final MP4 file" from scripted content [8]. Once all scene components are defined, Remotion renders the video (headlessly on Vercel or a serverless environment) into the final MP4.

8. **Post-Processing & Compilation:** If the video was generated in segments (e.g. intro, body, outro), they are concatenated (via Remotion or FFmpeg). Metadata (title, caption text) is extracted. The system might also auto-generate social copy: e.g. transcribing the final audio with Whisper and running that text through GPT/Claude to craft a caption or description.

9. **Output Delivery:** The final short video (vertical format) and any accompanying text (subtitles file, social caption) are saved. The user is notified (email or in-app) with a link to review the output. The platform might optionally push the video to the user's connected social accounts via APIs (similar to how upload-post.com can distribute videos to TikTok/YouTube [14]), or provide download links.

10. **User Rating & Feedback:** The user watches the video and provides feedback (e.g. thumbs-up/down, quality rating, or edits). This rating feeds into the evaluation system. A low rating can flag the workflow to run an improvement iteration (e.g. tweak the script or swap an avatar), while a high rating may finalize the content.

Input/Output Structure

- **Inputs:** A single MP4 file (up to a set size) and an optional text prompt (user instructions). The video is treated as the "source content." It can be a lecture, interview, tutorial, etc. The workflow expects the video to contain clear audio for transcription.
- **Intermediate Data:**
- Audio file extracted from the video.
- Whisper transcript with timecodes and key phrase highlights.
- Script text (LLM-generated).
- TTS audio file (ElevenLabs).
- Avatar video clip (if any).
- Generated visuals (images/videos from Runway or stock APIs).
- JSON logs of prompts, API responses, and metadata (for auditing).
- **Outputs:** A short video MP4 (vertical orientation), plus any generated assets: subtitle file (VTT/SRT), text captions (social post), and versions of the script. All output versions are versioned and stored (e.g. in S3 or a database) for review.

API & Tool Integrations

- **Storage/API Gateway:** Use a secure storage service (e.g. AWS S3) for inputs/outputs and an HTTP endpoint (or n8n webhook) to trigger the workflow when a video is uploaded.

- **Scene Detection:** Google Cloud Video Intelligence API can analyze content and detect scene changes, objects, and labels [10]. Alternatively, a simple ffmpeg keyframe detection can pick representative frames. The chosen scenes guide highlight selection.
- **Transcription:** OpenAI's Whisper API (or Whisper.cpp locally) transcribes audio to text [11]. This provides the spoken-word content and timecodes. (Whisper.cpp is highly efficient for CPU servers.)
- **Script Generation:** Claude/GPT via their API (HTTP request) is used to generate the video script. The prompts include extracted highlights and any user guidance. This is analogous to using OpenAI in other video pipelines to create voiceover text [2]. If available, an MCP-compatible interface (Model Context Protocol) can be used: for instance, Gyori's Short Video Maker service exposes a `create-short-video` tool via MCP [15], which Claude could call directly.
- **TTS (Text-to-Speech):** ElevenLabs API converts the script to speech [3]. Its SDK (or n8n's HTTP node) uploads the script and retrieves an MP3 of the voiceover. The system supports choosing different ElevenLabs models (e.g. ultra-low-latency vs expressive) based on use case.
- **Avatar Generation:** HeyGen's API or D-ID's API is called with the script audio (and optionally an image/avatar). For example, D-ID's API "lets you turn a still photo … and script (text or audio) into a realistic video of a digital presenter speaking" [16]. The agent chooses or clones an avatar, sends it the audio, and gets back a talking-head video clip.
- **Visual Generation:** RunwayML's Gen-3/4 models (via REST) can create or stylize video/image assets [13]. For example, one might generate a background video from text cues. Stock APIs (Unsplash/Pexels) can also be integrated via n8n nodes for generic B-roll.
- **Video Rendering (Remotion):** Remotion's rendering engine is invoked (possibly via a serverless call or Docker). Remotion is integrated into the Next.js app or run on a cloud function. It consumes the assembled React composition of scenes, using a headless Chromium to produce MP4 output [8] [7]. The Vercel deployment includes the Remotion endpoint for server-side rendering.
- **Fonts/Graphics:** Any custom fonts or style guides are stored in the code repo (Deployed with the app). Remotion's components reference these assets when generating text and overlays.
- **Social APIs (optional):** To automate posting, use platform APIs or an aggregator (like upload-post.com) to send videos to YouTube, TikTok, Instagram, etc. (Not required initially, but easily pluggable via n8n's HTTP nodes.)

At each stage, n8n's HTTP Request or dedicated nodes handle the API calls. Using standardized interfaces (REST/MCP) ensures each tool can be swapped. For instance, if a new TTS API is preferred, only the TTS node's endpoint changes. Notably, Anthropic's MCP standard allows LLMs to call tools generically [15] – the pipeline could be exposed as an MCP service so agents like Claude can invoke it directly.

n8n Workflow & Automation

We orchestrate the entire pipeline with n8n workflows:

- **Main Workflow (Video Pipeline):** Triggered by the upload event (HTTP/Webhook or S3 trigger). It proceeds step-by-step:
- *Start Node:* Receives the video file metadata.
- *Scene Extraction:* Uses an "HTTP Request" node to call Google Video Intelligence, and a "Function" node to parse shot boundaries.
- *Transcription:* Uses a "HTTP Request" to Whisper or Speech-to-Text API. Stores the transcript in n8n data.

- *Script LLM:* A “Chat” or “HTTP Request” node sends the transcript/highlights to Claude/GPT, returning a draft script.
- *TTS (ElevenLabs):* Another HTTP node sends the script to ElevenLabs and retrieves the MP3.
- *Avatar Generation:* If enabled, a conditional branch calls HeyGen/D-ID.
- *Remotion Assembly:* A **Code** or **Function** node generates the Remotion composition config (JSON) for the scene sequence. Then an execution node (e.g. via n8n’s “Execute Command” or a custom function) runs `remotion render` with that config.
- *Compilation:* If multiple video segments exist, a merge step uses FFmpeg (or Remotion’s concat) to combine them.
- *Output:* The final MP4 is uploaded to storage. Metadata (links, timestamps) is recorded.
- *User Notification:* Optionally send an email or webhook to notify the user for review.

- **Error Handling & Retries:** n8n supports defining an **Error Workflow** for each main workflow [17]. If any node fails (e.g. API timeout), the Error Trigger fires. We configure retries on critical nodes (e.g. TTS, Remotion render). For example, the “HTTP Request” node for ElevenLabs can be set to retry on 5xx errors. If a segment fails repeatedly, the Error Workflow can log the failure, alert the dev team (email/Slack), and optionally retry the entire workflow or skip to an alternative step.

- **Chaining and Parallelization:** Nodes pass JSON data, so the output of one step feeds into the next. n8n’s **Set** and **Function** nodes can massage data (e.g. extract transcript snippets). Conditional branches allow optional stages (e.g. skip avatar generation if no face image provided). The **Merge** or **SplitInBatches** nodes manage segment processing if the video is split.

- **Logging & Versioning:** Throughout the workflow, we use n8n’s **Database** nodes (or external DB actions) to log intermediate results (script text, audio URLs, prompts used). This ensures every output is versioned: we store which LLM prompts and tool versions produced each video. For example, before sending prompts to Claude, we write them to a log table; after rendering, we tag the output with a version ID. This traceability enables later auditing or rollbacks.

- **Example n8n Template:** n8n.io provides an AI video generation template that chains OpenAI (for script) and ElevenLabs (for voice) and final rendering [2] [18]. We follow a similar multi-node structure: AI nodes generate content, HTTP nodes call APIs, and a final render node produces the MP4.

Evaluation & Feedback Loop

To ensure quality, we implement an evaluation system in n8n:

- **Human-in-the-Loop Ratings:** After each video is produced, the user can rate its quality (e.g. 1–5 stars, or “approve/reject”). These ratings are captured (via a simple web UI or integrated survey) and fed back into the system. Each rating is stored alongside the versioned output and prompts. Over time, this builds a dataset of example inputs and judged outputs.

- **N8n Evaluation Framework:** We leverage n8n’s **Evaluation** feature to test the workflow against known cases [19]. Initially, we create a small “light” evaluation set: manually selected sample videos with expected scripts and video outcomes (e.g. transcripts). As the system runs, we scale up to

metric-based evaluation: using the collected real user data as test cases. For each case, we compare the generated script/video against either a human-written target or quality metrics (e.g. transcript accuracy, voiceover clarity). Metrics like BLEU or ROUGE (for script vs. reference) or audio fidelity scores can be used.

- **Continuous Quality Monitoring:** Whenever users flag a video as poor, the inputs (original video, prompts) are added to the evaluation dataset. We periodically run these examples through the latest workflow (regression testing) to ensure fixes haven't broken older cases [20]. Evaluation nodes in n8n can be set to alert if scores drop below a threshold.

- **Usage of Evaluations:** By "running data through them and observing the output" [19], we quantify reliability. We track metrics like LLM token usage, API latency, and user rating trends. This informs improvements (e.g. adjusting prompts, swapping tool versions). Human feedback effectively becomes the "expected output" for training and refining the agents.

Modularity & Extensibility

The system is designed for easy swapping of components:

- **Tool-Agnostic Interfaces:** Each stage communicates via well-defined JSON or media formats. For example, the TTS stage only expects script text and returns an audio file; the avatar stage needs an image/audio and returns a video clip. This means that to switch from ElevenLabs to another TTS, only that HTTP node's URL and credentials change.

- **MCP-Ready Connections:** Where possible, we use or create MCP-compatible tools. Anthropic's MCP standard allows LLMs to use tools seamlessly [15]. For instance, the entire video-assembly stage could be offered as an MCP tool (`create-short-video`). Tools like the open-source Short Video Maker are natively MCP-enabled. We ensure our API calls follow consistent patterns so that, if an agent framework needs to call them, it's straightforward.

- **Configuration-Driven:** All APIs and models are specified in config (e.g. environment variables). Adding a new service is just a matter of adding its endpoint and credentials. For example, if HeyGen is swapped for D-ID, we update the avatar-generation node's endpoint and payload format. Because our design decouples agents, we can also parallelize or chain additional tools (e.g. insert a music-generation step before final render).

- **Reusable Components:** The frontend (Next.js) and backend (n8n) are loosely coupled. The Next.js app can upload to any compatible storage, and n8n reads from there. Future interfaces (CLI, API) could trigger the same n8n workflows. The Remotion compositions (React code) are modular: new scene templates or effects can be added as separate components without altering the core pipeline.

In summary, the platform decomposes the short-video creation process into discrete AI-driven tasks. Each task uses the best available AI service (MCP/REST API) for that function. Tools are connected via n8n flows with robust error handling [17], and outputs are continuously evaluated and versioned [19]. This composable approach – chaining specialized components as in existing open-source pipelines [1] – ensures high flexibility and quality for the resulting short-form videos.

Sources: The design leverages modern AI APIs and orchestration patterns documented in sources like n8n workflow templates and AI-video pipelines [2] [8] [10] [3] [5] [4] [19]. These references illustrate how each component (LLM, TTS, avatar, Remotion) integrates to fully automate short-form video production.

---

[1] [8] [9] [11] [15] Why David Gyori's Short Video Maker is the AI Engineer's New Favorite Tool
https://skywork.ai/skypage/en/david-gyori-short-video-maker-ai-tool/1977980935875268608

[2] [12] [14] [18] Fully automated AI video generation & multi-platform publishing | n8n workflow template
https://n8n.io/workflows/3442-fully-automated-ai-video-generation-and-multi-platform-publishing/

[3] Text to Speech API - Ultra-realistic and low latency speech generation
https://elevenlabs.io/text-to-speech-api

[4] Create Realistic AI Video Avatars in Minutes | HeyGen
https://www.heygen.com/avatars/ai-video-avatar

[5] [6] [16] Generative AI API for Talking Head Video Creation | D-ID
https://www.d-id.com/api/

[7] Building a Web-Based Video Editor with Remotion, Next.js, and Tailwind CSS - DEV Community
https://dev.to/sambowenhughes/building-a-web-based-video-editor-with-remotion-nextjs-and-tailwind-css-pfg

[10] Video AI and intelligence | Google Cloud
https://cloud.google.com/video-intelligence

[13] API Documentation | Runway API
https://docs.dev.runwayml.com/

[17] Error handling | n8n Docs
https://docs.n8n.io/flow-logic/error-handling/

[19] [20] Evaluations | n8n Docs
https://docs.n8n.io/advanced-ai/evaluations/overview/