🎬 SOP: Automated Video Generation Engine via Google AI Studio

Objective: Turn Google AI Studio into a video creation engine by generating a self-playing, fully animated web interface with perfectly synced AI voiceovers and music, which is then screen-recorded.

Prerequisites:

- A Google account (logged into Google AI Studio).
- Screen recording software (OBS Studio, QuickTime, or Loom) capable of recording **System Audio**.
- A concept or script for your video.

STEP 1: Workspace Setup

The goal is to prepare the environment where the AI will write and render your code.

1. **Navigate to the Platform:** Open your browser and go to Google AI Studio.
2. **Open the Developer Environment:** Locate and click on the **Build** tab (or the specific coding/artifact interface used for creating applications).
3. **Prepare for Iteration:** Do not try to build the entire video in one prompt. You will use a sequenced prompting strategy, starting with visuals and ending with audio.

STEP 2: Build the Visual Foundation (Prompt 1)

The goal is to make the AI act as your motion graphics designer and build the visual shell.

Action: Write your first prompt focusing *strictly* on visuals and user interface.
Ensure your prompt includes these 4 instructions:

- ☐ **The Flow:** Describe the exact sequence of events, scene-by-scene (e.g., "Scene 1 shows the title, Scene 2 slides in bullet points...").
- ☐ **The Aesthetics:** Specify the design language. Use keywords like *smooth spring animations*, *glassmorphism UI*, or *dark modern theme* to simulate After Effects-style quality.
- ☐ **Playback Controls:** Explicitly state: *"Include a prominent 'Play Video' button at the top of the screen to trigger the animation sequence."*

- ☐ **Progress Tracker:** Explicitly state: *"Include a top progress bar that fills up as the sequence plays, mimicking a real media player."*

✅ ***Validation:*** *Do not move on until the AI generates a visually pleasing webpage where clicking "Play" triggers your requested animations in order.*

STEP 3: Integrate AI Voiceover & Syncing (Prompt 2)

The goal is to add a narrator and ensure the animations match the speaking time perfectly.

Action: Submit a follow-up prompt to add the voiceover.
Ensure your prompt includes these 3 technical instructions:

- ☐ **The Script:** Provide the exact script the AI voice should read for each scene. Tell it to sound like a professional product explainer.
- ☐ **The Model:** Instruct the AI to explicitly use the `gemini-2.5-flash-preview-tts` model for audio generation.
- ☐ **Perfect Synchronization (CRITICAL):** Paste this exact instruction to the AI:

> *"Do not use hardcoded durations for the scenes. You must dynamically calculate the duration of each scene based on the exact length of the generated AI voiceover audio file for that scene. The visual animations must wait for the audio to finish before moving to the next scene."*

STEP 4: Apply the Audio Playback Fix (Prompt 3)

The goal is to fix the "Element has no supported sources" browser error caused by raw AI audio.

Action: If the audio fails to play, submit this highly technical patch prompt to fix the audio formatting.
Ensure your prompt includes these instructions:

- ☐ **The Header Fix:** Tell the AI: *"The Gemini TTS model returns raw PCM audio data which HTML5 `<audio>` tags cannot play. Manually construct a proper WAV file header (24kHz, 1 channel, 16-bit) around the raw PCM data."*
- ☐ **The Delivery Method:** Tell the AI: *"Create a Blob and a playable Object URL from that constructed WAV data so the browser can play it."*
- ☐ **Add Download Utility:** Tell the AI: *"Add a 'Download Audio' button to the UI so I can download the generated .wav files."*

✅ ***Validation:*** *Click the "Play Video" button. You should now hear the AI speaking, and the animations should transition exactly when the voice finishes a sentence.*

STEP 5: Add Background Music (Prompt 4)

The goal is to mix an ambient track behind the narrator to make it feel like a polished video.

Action: Submit a final prompt to layer in external music.
Ensure your prompt includes these 3 instructions:

- ☐ **The Source:** Instruct the AI to embed a direct URL to a royalty-free MP3 using an HTML5 `<audio>` tag.
- ☐ **Audio Mixing:** Explicitly state: *"Set the volume of the background music to 0.1 (10%) so it does not overpower the AI voiceover."*
- ☐ **Trigger Sync:** State: *"Tie the background music playback to the main 'Play Video' button. It must start and stop exactly when the animation and voiceover start and stop."*

STEP 6: Record the Final Output

The goal is to capture the fully automated web experience as a standard video file.

Pre-Flight Recording Checklist:

1. ☐ Check the UI: Has the "Generating AI Voiceover" loading state finished?
2. ☐ Open Screen Recorder (OBS/QuickTime).
3. ☐ Set screen recorder to capture **System Audio / Browser Audio** (Do *not* record via your microphone).
4. ☐ Frame the recording window cleanly around the generated UI.
5. ☐ **Hit Record** on your screen recording software.
6. ☐ **Click "Play Video"** on the AI-generated website.
7. ☐ Let the sequence play from start to finish hands-free.
8. ☐ Stop recording and trim the beginning/end of the video file as needed.

💡 Bonus: The Master "Copy & Paste" Prompt Template

If you want to try bypassing steps 2-5, use this master prompt to build the engine in one go:

> "I want you to build a self-playing, fully animated web interface that functions exactly like a video.
>
> **Visuals:** Use a dark modern theme with glassmorphism UI and smooth spring animations. Include a prominent 'Play Video' button and a progress bar at the top.
> **Scenes:** [Insert Scene 1, Scene 2, Scene 3 descriptions here].

> **Voiceover:** Use the `gemini-2.5-flash-preview-tts` model to read [Insert Script].
>
> **Synchronization (CRITICAL):** Do not use hardcoded durations. Dynamically calculate scene lengths based on the exact duration of the generated TTS audio.
>
> **Audio Fix (CRITICAL):** Because the TTS returns raw PCM data, you must manually construct a WAV file header (24kHz, 1 channel, 16-bit), turn it into a Blob, and create an Object URL so the browser `<audio>` tag can play it without errors. Add a button to download these .wav files.
>
> **Music:** Embed a direct link to a royalty-free MP3 using an `<audio>` tag, set the volume to 0.1, and tie its playback to the main 'Play Video' button."