Usuals: Designing Repeatable Video Production Workflows

Overview

This covers the workflow and template systems built for Usuals, not prompt-level fixes. The previous piece was about getting one prompt right. This one is about how a request actually moves through a system end to end, and why it's structured the way it is. Diagrams below are pulled directly from the working planning board.

Research approach

Early pipelines treated every video as a one-off: one brief, one long prompt, one generation. That breaks the moment anything needs reuse, a character in a second scene, a brand voice in a second format, because the context lived inside that single prompt and died with it. Every system below answers the same question differently: what gets generated once and reused, versus generated fresh per use.

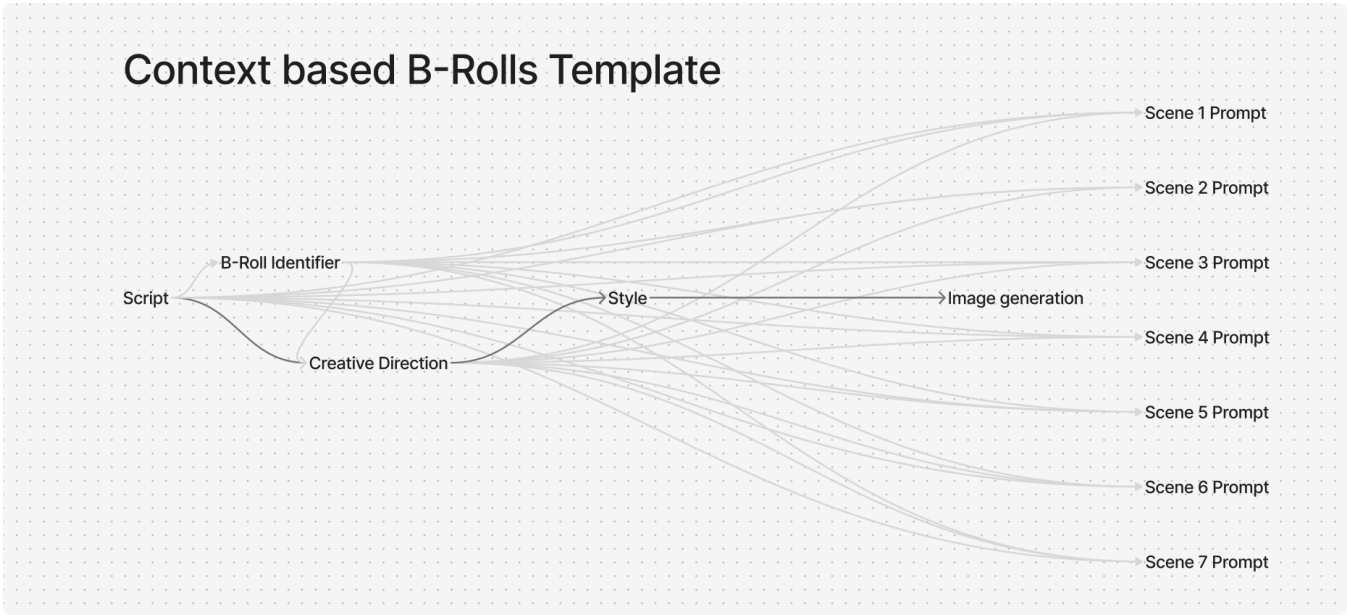
- Generated once, reused: character identity, brand identity, location, visual style
- Generated fresh, per use: scene prompts, shots, dialogue, scene-specific frames

System: context-based B-roll template

A script feeds two readers that split the work, then recombine before generation.

- Script goes to two nodes: a B-Roll Identifier (finds moments that need supplementary footage, not dialogue shots) and Creative Direction
- Both fan out directly into seven scene-prompt slots, one per scene
- Both also converge separately into a single Style node
- Style runs its own generation step before Image generation, on its own line, apart from the per-scene prompts

Style gets locked once, upstream of the per-scene fan-out. That's what keeps scene 1 and scene 7 looking like they came from the same shoot instead of seven independently written looks.

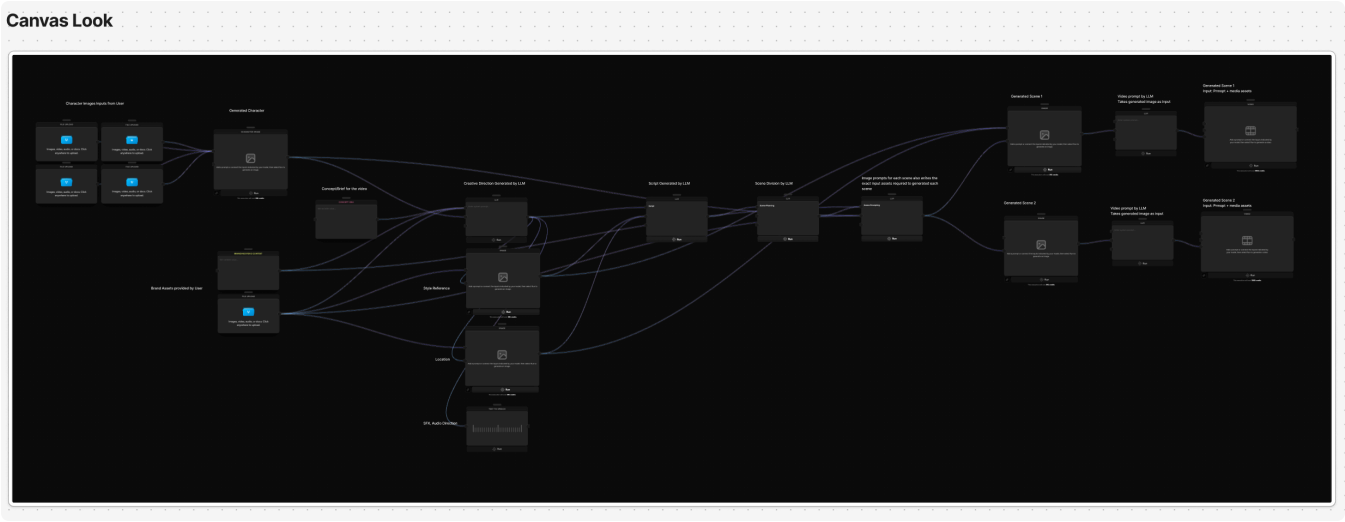


System: node-based production canvas

The interactive version, built as a canvas a person runs node by node.

- Character images come in through four upload slots per character (image, video, audio, or a link), feeding a Generated Character node
- Brand assets come in through a separate upload path
- Character, brand, and a Concept/Brief node all feed one Creative Direction node, generated by an LLM
- Creative Direction branches into three independently runnable nodes: Style Reference, Location, SFX/Audio Direction, each with its own run button
- All three converge into Script, then Scene Division
- Scene Division writes the prompt for each scene and decides exactly which upstream assets that specific scene needs
- Each Generated Scene gets its own video-prompt step that takes the already-generated image as input

Nodes run independently on purpose, so a wrong location image gets regenerated on its own without rerunning anything else in the chain.



System: a simple front end over a structured back end

What a user sees is four steps. What runs underneath isn't linear.

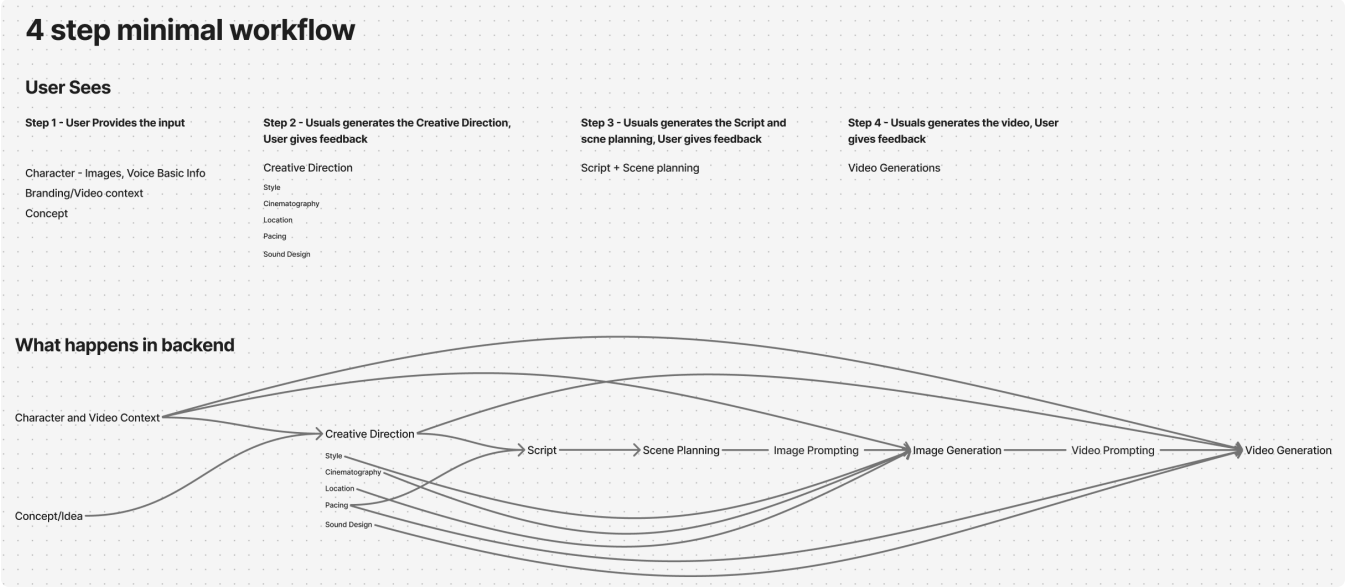
User-facing steps

- Step 1: character (images, voice basics), branding/video context, concept
- Step 2: Creative Direction, broken into five parameters: style, cinematography, location, pacing, sound design
- Step 3: script and scene planning
- Step 4: video generation

What runs underneath

- Character/video context and concept both feed Creative Direction
- Both also run direct lines straight into Image Generation and Video Generation, skipping past Script and Scene Planning

The direct lines matter because later generation steps don't only inherit from whatever the script stage produced a few hops earlier. They're re-grounded against the original brief directly, which keeps a five-hop pipeline from drifting away from what was actually asked for.

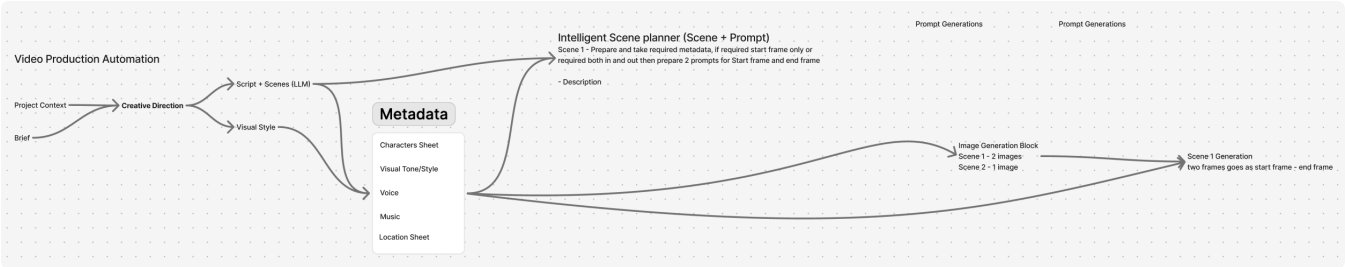


System: intelligent scene planner

One shared metadata object, and a planner that decides frame counts per scene instead of assuming them.

- Project context and a brief feed Creative Direction, which splits into Script and Scenes, and Visual Style
- Both branches read and write a single Metadata object: Characters Sheet, Visual Tone/Style, Voice, Music, Location Sheet
- For each scene, the planner checks whether it needs a start frame only, or both a start and end frame, and prepares one or two prompts accordingly
- Two frames for a scene become the literal start and end frame for that scene's video generation

The planner's logic is conditional, not templated. In the working example, Scene 1 got two prompts and two images, Scene 2 got one, decided per scene instead of generating the same shape every time.

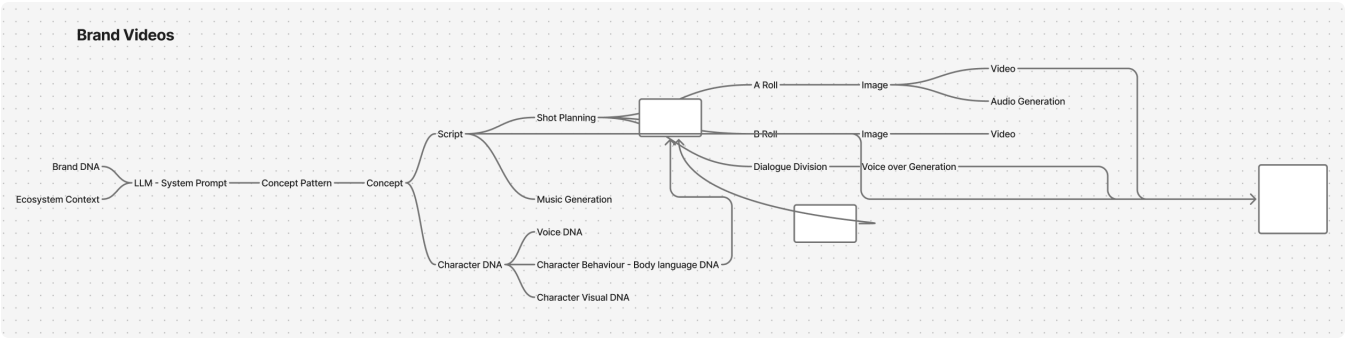


System: brand video pipeline

Built around one idea: a character's identity should be generated once and checked against, not redescribed every time it's used.

- Brand DNA and ecosystem context run through an LLM system prompt into a Concept Pattern, then Concept
- Concept splits into Script and Character DNA
- Character DNA splits into three: Voice DNA, Character Behaviour/body-language DNA, Character Visual DNA
- Script splits into Shot Planning, Music Generation, Dialogue Division
- Shot Planning drives A-roll and B-roll, each generating its own image then video
- Dialogue Division feeds Voice-over Generation, which also produces video
- A-roll, B-roll, and voice-over all converge into one final compile step

Character DNA loops back into the shot-planning block directly instead of staying parked at the top. Voice, behaviour, and visual identity get checked against at generation time, not just described once and hoped for.



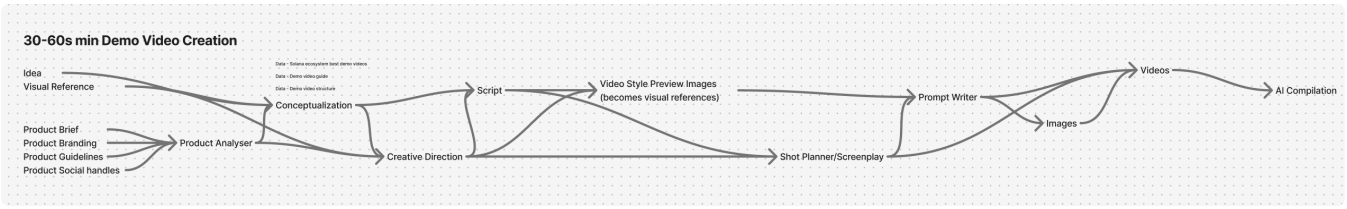
System: demo video pipeline

Scoped tightly to a 30 to 60 second demo video, with its own research grounding built in.

- Idea and a visual reference feed Conceptualization, while product brief, branding, guidelines, and social handles feed a separate Product Analyser, in parallel
- Conceptualization is grounded against three reference sources: strong demo videos, a demo video guide, a documented demo video structure
- Both feed Creative Direction, which produces a Script and also feeds Shot Planner/Screenplay directly
- Script generates Video Style Preview Images, explicitly labeled as becoming visual references

- Those preview images feed back into Prompt Writer alongside the shot plan
- Prompt Writer produces images and videos, which go to AI Compilation

Nothing here gets thrown away once the script is approved. The preview images loop back in as grounding for the actual generation prompts.



System: a template library, not one-off pipelines

The recurring pipelines got turned into templates, each with a narrow, specific input list.

- Product Demo Script Builder: product description, features, target audience, optional rough notes
- Demo Video Shot Planner: demo script, creative direction, optional product visuals
- Founder-Led Demo Video Builder: founder character DNA and tone, product info, launch message, optional selfie or founder footage style
- Screen Recording to Polished Demo: raw screen recording, optional notes, optional desired tone
- Use-Case Demo Template
- Pre-production templates

None of these templates ask for character DNA, metadata, or style from scratch, since that already exists upstream and gets pulled in. A template's input list only holds what's actually unique to that format.

System: cinematography parameter library

Cinematography is broken into eleven independently settable parameters instead of one blended description.

- Eleven parameters, each tracked and settable on its own: Camera, Focal Length, Lens Style, Aperture/Depth of Field, Lighting Style, Shutter, Aspect Ratio, Color Grade Style, Contrast Style, Texture, Sensor Format
- Any one of them can be set per project or per scene without touching the others

This came from the same problem as everything else here. The word "cinematic" means something different every time a model interprets it, so breaking it into eleven named parameters makes a look specifiable and repeatable instead of hoping the model lands on the same read twice.

System: location DNA

Locations get the same generate-once, reuse-everywhere treatment as characters.

- A text brief for the location, pulled from a script or concept, goes through an analysis LLM
- That LLM outputs two separate image-model prompts: one for a 4-grid location image, one for a wide master shot
- Both generations run, producing a 4-grid image and a master wide shot
- Those two images, plus an image address, become a Location DNA asset

A location gets generated once, from two angles, then referenced by its DNA asset everywhere it's needed after that, the same pattern used for character identity.

System: character DNA from a questionnaire

Character identity doesn't only come from uploaded footage. There's a second path for when the input is a visual reference plus structured answers instead.

- A visual reference and a questionnaire go in
- A first LLM stage analyzes the visual reference against the answers
- A second LLM stage takes that analysis and builds the Character DNA object, the same one every other pipeline reads from

Analysis and construction run as two separate LLM passes instead of one. Asking a single pass to both interpret a reference image and author a structured identity object produced weaker DNA than splitting the reasoning in two.

System: character restoration

Not every character image comes in clean. There's a dedicated pipeline just for fixing the ones that don't.

- Character images and separate restoration images go in

- An LLM prompt analyzes what's wrong with the source and what needs correcting
- Output is a set of restored images that feed back into the main character pipeline

It runs as its own pipeline instead of a retry step hidden inside character generation, because restoring a broken asset is a different job from generating a new one.

System: multi-shot campaign pipeline

Built for producing a full set of shots off one character cast without regenerating identity for each one.

- A user prompt runs through a Prompt Extractor, then Shot Pre-planning
- Five character images get read once and reused across the entire set
- Each shot gets its own prompt, generated individually, then run through the image model
- Each shot's resulting image gets its own video prompt and its own separate video generation pass

Shot pre-planning happens once, upfront, before any image generates. That's what keeps a ten-shot campaign from turning into ten separate improvisations off the same five reference images.

System: demo video creative direction generator

The demo pipeline's Creative Direction stage, mentioned above as one box, isn't one LLM call. It's nine chained stages, each doing one narrow job.

- Product/Feature Analyzer: reads what the product actually does
- Audience and Use-Case Extractor: who it's for and what they'd use it for
- Core Message Distiller: the one thing the video needs to land
- Demo Narrative Structurer: the shape the story takes
- Visual Language/Style Mapper: what it should look like
- Tone and Pacing Definer: how fast, in what mood
- Proof-Moment Planner: where the video actually proves the claim instead of just stating it
- Reference Interpretation Layer: reads in the three grounding sources (best demo videos, demo video guide, demo video structure)

- Output Formatter for Production Handoff: turns all eight prior outputs into the format the next pipeline stage expects

Each stage only has to get one thing right, instead of one prompt reasoning about product, audience, message, visuals, pacing, and proof all at once. The chaining is the content strategy, not something bolted on after generation.

System: faceless video templates

A separate template family for videos that never show a person on screen.

- Project context and a script, VO, or idea go in
- Creative Direction splits into a Visual Style Division, which offers two visual style options to choose between rather than locking one
- A Scene Division stage breaks the piece into individual scenes, four in the working example
- A Voice Over Generator runs in parallel
- A B-roll Planner assigns style to supporting footage separately from the main scene generation

Visual style is offered as a choice between two directions instead of generated once and locked. Without a character on screen, visual style carries more of the video's identity than usual, so it's worth a second option before committing.

System: talking head pipeline

Three entry points into the same underlying pipeline: from a concept, from a script, and from a script built specifically for a product video.

- All three share the same building blocks: Metadata, a Scene Description, a Prompt Generator (LLM), and a step that combines a fixed Character Persona with the per-scene prompt into one complete prompt
- Video generation for this format runs through specific named models, Seedance and Kling
- B-roll gets slotted in alongside the talking-head footage rather than added as an afterthought

Character Persona and Scene Prompt get combined at generation time instead of writing one long prompt per scene from scratch. The persona stays fixed and only the scene half changes, which is what keeps the same character consistent across scenes generated independently of each other.

System: poster and static asset template

The same generate-once-and-reuse context logic, applied to a single static image instead of a video.

- Six input categories: project context, user brief, text input, branding, visual references, other elements
- A Design Style (illustration, 3D, live-action image, or other) gets picked upfront
- An LLM analyser and prompt maker reads all of it and builds the image-model prompt
- The image model generates against that single prompt

Design Style gets chosen before anything generates, not inferred afterward. A poster only gets one shot, so it needs to be built for a style from the start rather than nudged toward one later.

System: animation pipeline

This pipeline ends in a program that renders a video, not a generated video file directly.

- Script/VO, branding, and creative direction feed a Design Direction stage, then an Animation Script
- The animation script splits into multiple animation sequences, four in the working example, each built from a visual style reference, UI and icon assets, and font files, not generated images
- Separate LLM stages handle script formatting, creative direction, animation scripting, sequence extraction, location, and visual style, each its own pass
- The render path: a scene becomes structured scene data, gets matched to a template, passed in as component input, rendered frame by frame, then encoded to video

Animation skips image and video generation models entirely and runs through a rendering engine driven by structured data instead, because frame-accurate motion and exact brand assets are things a generative model can approximate but a renderer can guarantee.

Templates

1. Product Demo Script Builder

Input:

- product description
- features
- target audience
- optional rough notes

2. Demo Video Shot Planner

Input:

- demo script
- creative direction
- optional product visuals

3. Founder-Led Demo Video Builder

Input:

- founder character DNA tone
- product info
- launch message
- optional selfie / founder footage style

4. Screen Recording to Polished Demo

Input:

- raw screen recording
- optional notes
- optional desired tone

5. Use-Case Demo Template



What this demonstrates

The same handful of decisions shows up across every system above:

- Separate what's reusable (identity, brand, style, location) from what's generated per use (a scene, a shot, a line)
- Keep one shared metadata object as the source of truth instead of letting each stage redescribe what came before it
- Re-ground later generation steps against the original brief instead of trusting a chain of LLM hops not to drift
- Make generation granular enough that fixing one wrong node doesn't mean rerunning the whole pipeline

This is the infrastructure that made repeatable, multi-scene AI video production possible at Usuals.

Utkarsh Bramhankar · Usuals AI