Stack Implementation

The 'Snoopiest' Engine: A Hybrid Monorepo Architecture

Stack Implementation

This section outlines best-in-class, open-source technologies used for each layer of the Snoopiest architecture. Each selection is optimized for performance, scalability, and compatibility with modern frameworks like Next.js 15 and React 19.

MONOREPO TOOLING

NX (https://nx.dev/)

Best for complex, polyglot projects. Offers a rich plugin ecosystem (including Python), advanced dependency graphing, and robust caching.

ALTERNATIVES

Turborepo (https://turbo.build/)

WEB FRAMEWORK

The industry standard for building full-stack React applications. Providing optimized performance with SSR, SSG, and React Server Components.

ALTERNATIVES

Remix (https://remix.run/) Astro (https://astro.build/) TanStack Start (https://tanstack.com/start)

UNIVERSAL FRAMEWORK

Expo (https://expo.dev/)

Build for Web, iOS, and Android from a single TypeScript codebase. Features a powerful CLI and OTA updates.

ALTERNATIVES

Tamagui (https://tamagui.dev/)

API LAYER

tRPC (https://trpc.io/)

Enables end-to-end typesafe APIs with zero code generation. Unbeatable DX in a full-stack TS monorepo.

ALTERNATIVES

GraphQL (https://graphql.org/) REST (OpenAPI) (https://www.openapis.org/)

DATABASE

PostgreSQL (https://www.postgresql.org/)

Powerful, open-source relational database known for reliability and performance at scale.

ALTERNATIVES

MySQL (https://www.mysql.com/) SQLite (https://www.sqlite.org/)

DATABASE ORM

Drizzle ORM (https://orm.drizzle.team/)

Lightweight, performant, and type-safe SQL query builder with SQL-like syntax.

ALTERNATIVES

Prisma (https://www.prisma.io/)

AUTHENTICATION

better-auth (https://www.better-auth.com/)

Comprehensive, framework-agnostic auth for TypeScript. Self-hostable and avoids vendor lock-in.

ALTERNATIVES

- Supabase Auth (https://supabase.com/auth)
- Clerk (https://clerk.com/)
- WorkOS (https://workos.com/)
- Firebase Auth (https://firebase.google.com/products/auth)

AI/ML SERVICES

FastAPI (Python) (https://fastapi.tiangolo.com/)

High-performance Python web framework ideal for building AI/ML APIs and leveraging Python's ML ecosystem.

ALTERNATIVES

- Flask (https://flask.palletsprojects.com/)
- Django Ninja (https://django-ninja.rest-framework.com/)

HEADLESS CMS

PayloadCMS (https://payloadcms.com/)

Developer-first, open-source headless CMS built with TS and React. Deep Next.js integration.

ALTERNATIVES

- Strapi (https://strapi.io/)
- Directus (https://directus.io/)

CLIENT DATA FETCHING

TanStack Query (https://tanstack.com/query/latest)

De-facto standard for managing server state in React. Provides caching and background refetching.

ALTERNATIVES

SWR (https://swr.vercel.app/) Apollo Client (https://www.apollographql.com/docs/react/)

UI DATA GRIDS

TanStack Table (https://tanstack.com/table/latest)

Headless UI library for building powerful and fully customizable data tables and grids.

ALTERNATIVES

AG Grid (https://www.ag-grid.com/)

E2E TESTING

Playwright (https://playwright.dev/)

Modern, reliable E2E testing framework with true cross-browser support and auto-waits.

ALTERNATIVES

Cypress (https://www.cypress.io/)

COMPONENT TESTING

Storybook (https://storybook.js.org/)

Essential tool for developing UI components in isolation. Serves as a living documentation.

ALTERNATIVES

Ladle (https://ladle.dev/)

The AI Model Zoo (Execution Layer)

We utilize a Best-in-Class Modular Approach rather than a single provider. This prevents vendor lock-in and allows upgrading specific components (e.g., swapping the Image Generator without breaking the Text Analyzer).

> *“[!NOTE] **Cost Analysis:** A detailed breakdown of the costing layer is available in the Cost Estimator.”*

LOGIC / TEXT

Claude 3.5 Sonnet (https://www.anthropic.com/)

PROVIDER

`Anthropic API` (https://docs.anthropic.com/)

> *"Superior reasoning capabilities and larger context window (200k) for analyzing full chapters."*

IMAGE GEN

Flux.1 [Dev] (https://blackforestlabs.ai/)

PROVIDER

`Replicate / Fal.ai` (https://replicate.com/)

> *"Currently beats Midjourney in prompt adherence and text rendering."*

VIDEO GEN

Luma Dream Machine (https://lumalabs.ai/dream-machine)

PROVIDER

`Luma API (https://lumalabs.ai/)`

> *"High temporal coherence. Relies on "Keyframe" feature for control."*

AUDIO / TTS

ElevenLabs (Turbo v2) (https://elevenlabs.io/)

PROVIDER

`ElevenLabs API (https://elevenlabs.io/api)`

> *"Low latency and highest emotional range."*

LIP SYNC

SyncLabs / SadTalker (https://synclabs.so/)

PROVIDER

`API / Local`

> *"Decoupled lip-syncing ensures we can perfect audio performance before mapping to video."*

Advanced Document Management

We treat the screenplay not just as text, but as **executable documentation**.

The Quarto (QMD) Pipeline

1. **Source**: `` `Chapter_01.md` `` (Raw Text).
2. **Processing**: The Agent converts this into `` `Script_01.qmd` `` (Quarto Markdown).
3. **Metadata Injection**: The Agent embeds JSON metadata (Camera angles, Lighting) inside YAML headers or hidden code blocks within the QMD.
4. **Render**:

    - **For Humans**: Quarto renders a clean PDF looking like a Hollywood script (Courier font, proper indentation).
    - **For Robots**: The system parses the underlying JSON data blocks from the same file to drive the video generator.

> *"[!TIP] **Single Source of Truth**: The readable PDF script reviewed by humans is the exact same code that generates the video."*

Audio & Lip Sync Architecture

Professional production requires **Decoupling**. We generally avoid "all-in-one" generators to maintain granular control over performance.

- **Step 1: Audio Production (The Radio Play)**

    - Generate full audio track using ElevenLabs.
    - **Forced Alignment**: Use tools like Gentle or OpenAI Whisper to get exact timing of every word.

- **Step 2: Video Generation (The Silent Film)**

    - Generate the 8-second video visuals based on the visual prompt.

- **Step 3: The Sync Pass (Post-Process)**

    - **Lip-Sync**: Run Video + Audio through a dedicated Sync engine (Wav2Lip/SyncLabs).

---

5. Asset Management: “The Cloud-Local Mirror”

Team collaboration on 50GB+ video projects is challenging. We solve this with a “Split-Brain” storage strategy.

Storage Strategy

- **Code & Scripts:** `` `GitHub` `` (.md, .qmd, .json) - Version controlled, lightweight.
- **Heavy Assets:** `` `AWS S3 / Cloudflare R2` `` (.mp4, .png, .wav) - Cheap object storage.

The Sync Mechanism (`` `npm run asset:sync` ``)

1. Cloud Worker renders video → Uploads to S3 → Pushes Manifest to Database.
2. Local CLI detects new manifest.
3. **Node.js** `` `fs` `` generates folder structure locally matching the Chapter/Scene hierarchy.
4. Pulls only the new video files to your local folder.

---

6. Execution Environment

Writing / Logic

`Cloud (Anthropic)`

WHY: Requires massive GPU/TPU for LLM reasoning.

Folder Gen / Management

`Local (Node.js)`

WHY: Fast file system operations; zero latency UI updates.

Image/Video Rendering

`Cloud (Replicate)`

WHY: Requires A100 GPUs. Too slow/hot to run on local MacBook.

Final Assembly

`Hybrid`

WHY: FFmpeg WASM for quick previews; Cloud Lambda for 4K export.