Automated Pipeline for Daily Financial News Podcast Videos

Research Report
by Shayan Doroudiani
For Next Inc

Canva Link

Core Objective

The production of a daily video podcast on financial news involves combining several media and AI technologies in a single production process. Our aim is to develop a system that will automatically generate a daily five minute podcast which includes a collection of key financial news stories. The tone of the news will be standardised and the quality will be uniform. However, human intervention will be involved in the selection of the stories to be covered. Each day's news needs to be collected, articles have to be transformed into about 150 word factual summaries devoid of opinions or advice, a script has to be produced using current articles and including introductory elements and transitional elements, this script must then be converted to spoken word through voice replication technology, lastly a corresponding news broadcast must be produced with images and closed captions. Here is the rephrased text:

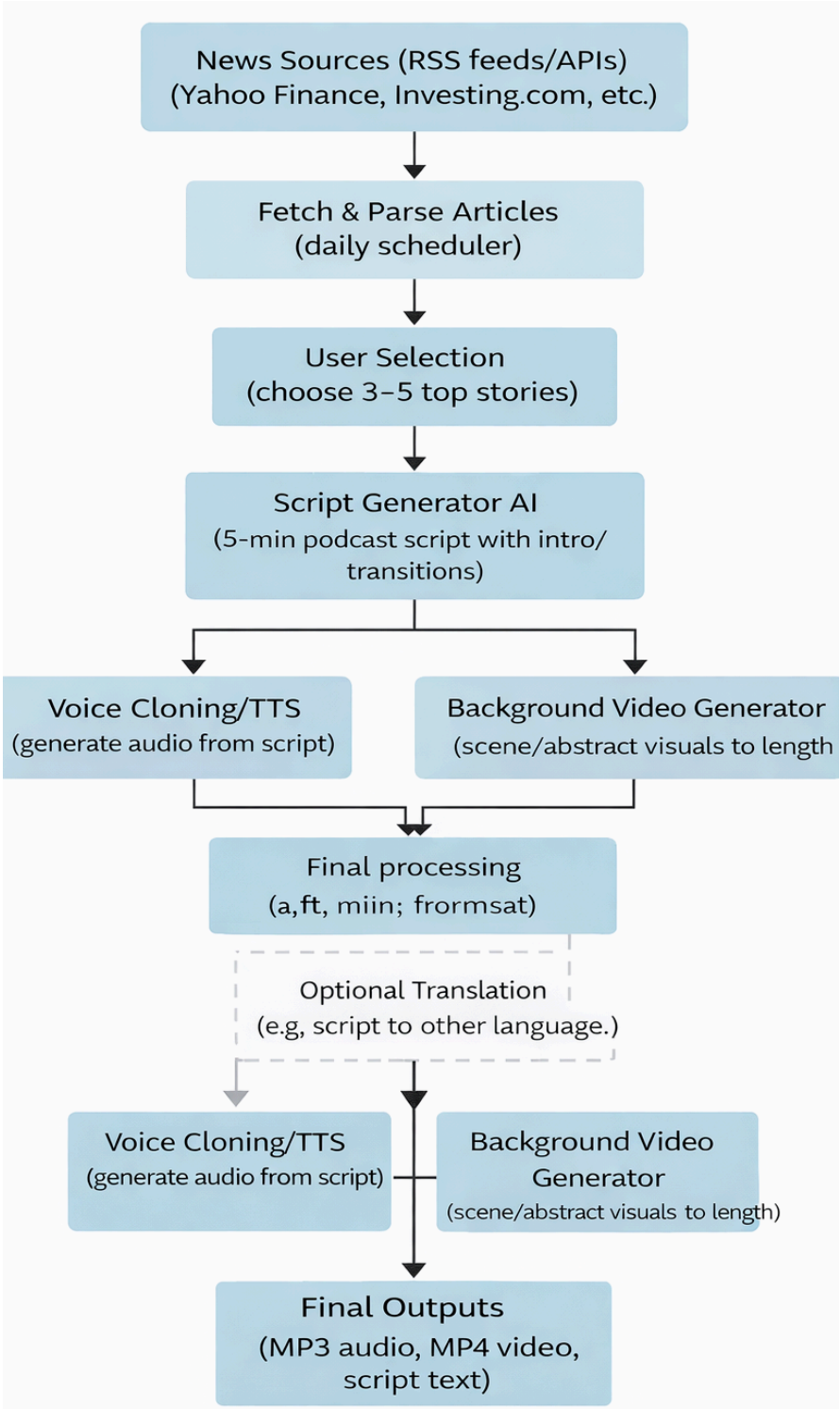
The output should be in a suitable YouTube format for all three types of output: audio (mp3), video (mp4) and the transcript (txt). This can be achieved by adding a translation function to the application, which could be linked to a database of translations or a translation service.

The system detailed below has been designed following a thorough examination of possible software tools which could be used in each component. The viability of the suggested system is discussed, along with proposals for any modifications which could be made at a later date. The system's architecture focuses on the use of various established AI techniques and third-party APIs to ensure that all processes run smoothly and efficiently. Many widely used services and open-source tools are identified and the trade-offs in terms of quality, complexity and cost are discussed where applicable. This document aims to provide a clear and comprehensive blueprint for the implementation of the automated pipeline.

System Architecture Overview

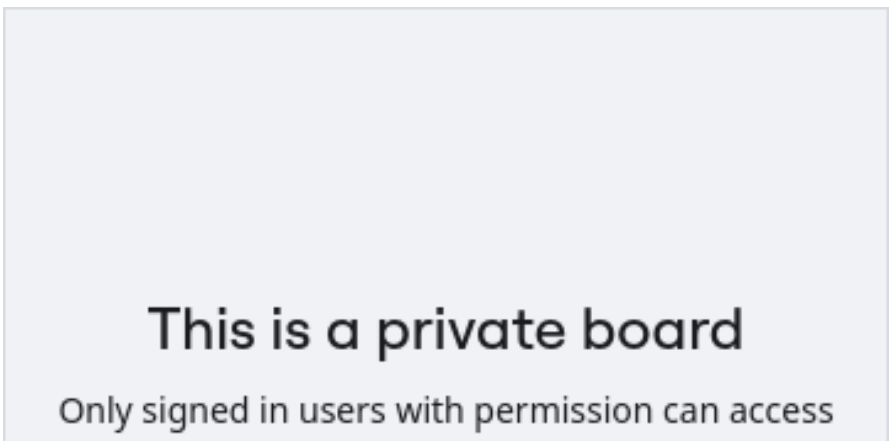
The system architecture consists of several key stages, each automating a critical part of the video creation process. Below is a simplified flowchart representing the pipeline.

1. **News Aggregation:** Collect financial news from various sources.
2. **AI Summarization:** Condense articles into concise summaries.
3. **User curation of top summaries**
4. **Script Generation:** Structure the summary into a podcast script.
5. **Voice Cloning:** Convert script to audio using AI.
6. **Video Generation:** Create video visuals using stock footage and animations.
7. **Output Packaging:** Combine audio and video into a final product.



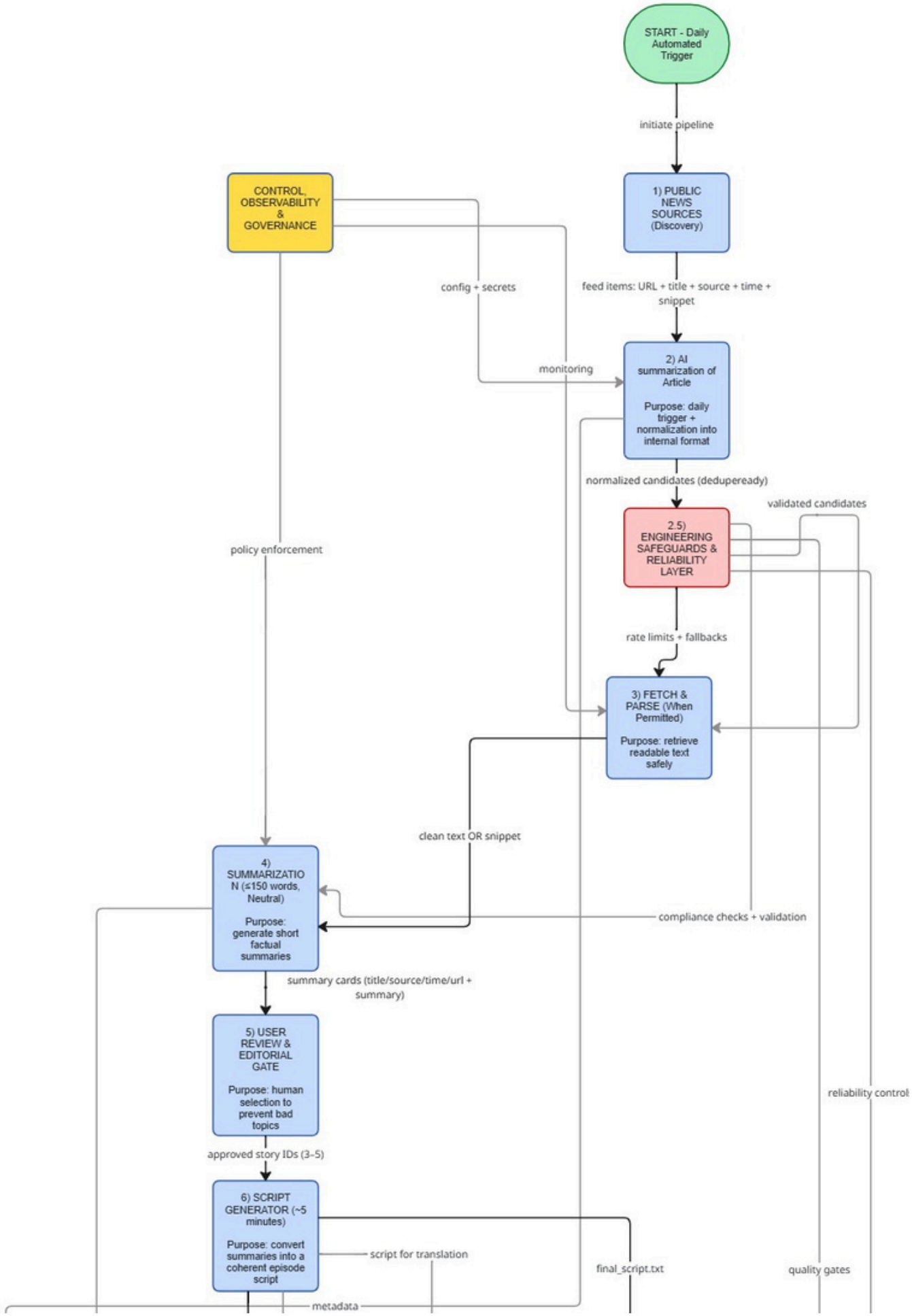
[Diagram Link]

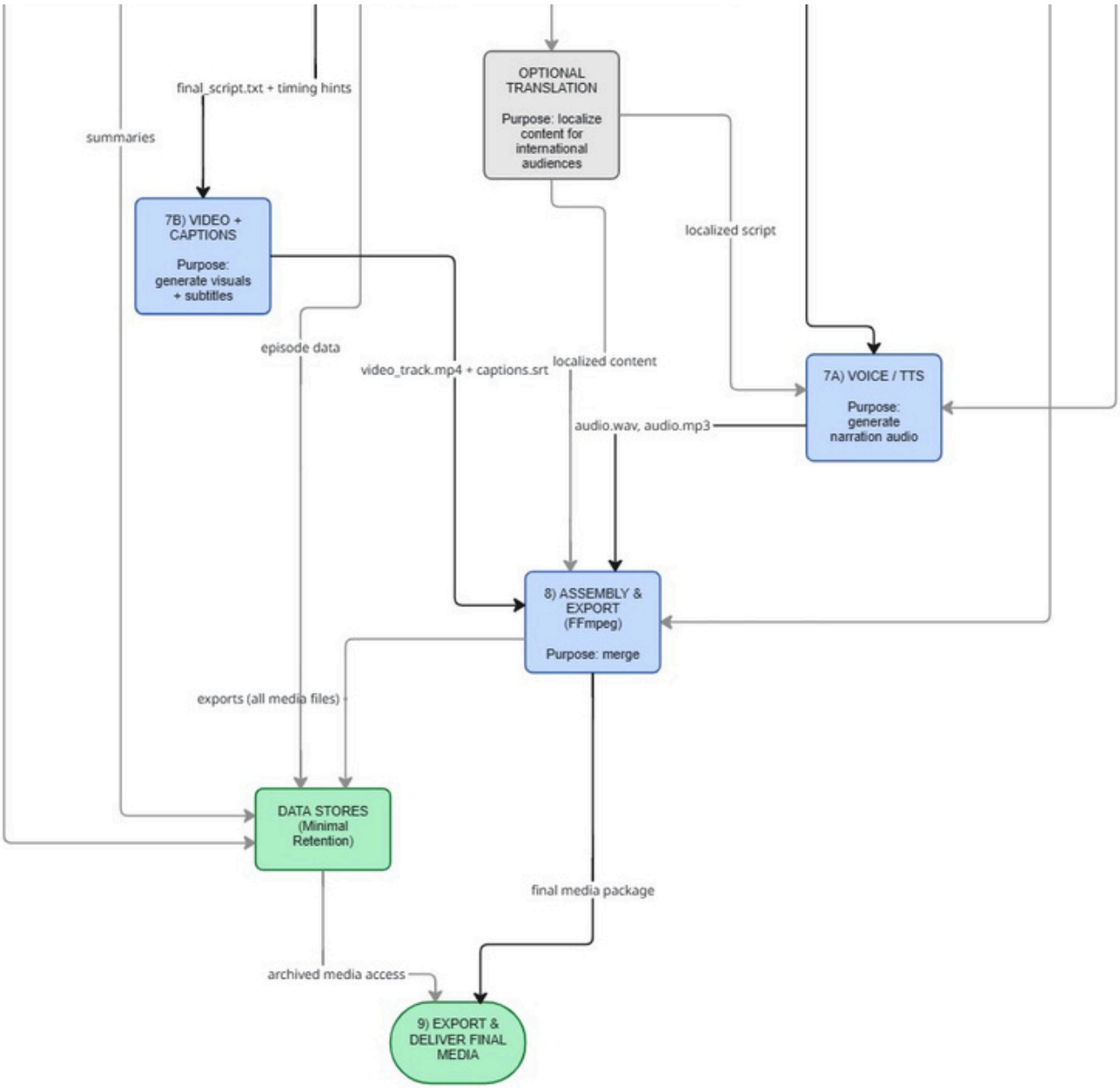
[Canva Link]



the board.

Sign in Sign up


START - Daily Automated Trigger
initiate pipeline
CONTROL, OBSERVABILITY & GOVERNANCE
1) PUBLIC NEWS SOURCES (Discovery)
config + secrets
feed items: URL + title + source + time + snippet
monitoring
2) AI summarization of Article
Purpose: daily trigger + normalization into internal format
normalized candidates (deduperready)
validated candidates
2.5) ENGINEERING SAFEGUARDS & RELIABILITY LAYER
policy enforcement
rate limits + fallbacks
3) FETCH & PARSE (When Permitted)
Purpose: retrieve readable text safely
clean text OR snippet
4) SUMMARIZATIO N (≤150 words, Neutral)
Purpose: generate short factual summaries
compliance checks + validation
summary cards (title/source/time/url + summary)
5) USER REVIEW & EDITORIAL GATE
Purpose: human selection to prevent bad topics
reliability control
approved story IDs (3-5)
6) SCRIPT GENERATOR (~5 minutes)
Purpose: convert summaries into a coherent episode script
script for translation
final_script.txt
quality gates
metadata



1. News Fetching from Public Sources

The system starts by pulling financial news data which it retrieves from public sources that run continuously throughout each day. The reliable sources for information include Yahoo Finance and Investing.com together with Reuters and Bloomberg and market news aggregators. The system can retrieve new articles through RSS feeds or public news APIs which operate as automatic article retrieval systems:

- **RSS Feeds:** A lot of financial websites provide their most recent news via RSS feeds. For instance, offers RSS feeds for every news category, providing "immediate access to breaking news, in-depth analysis, investing strategies... coverage of the macroeconomy, bonds, commodities, and foreign exchange. Although support has varied, Yahoo's finance news can also be retrieved via Yahoo's RSS (e.g., https://news.yahoo.com/rss/finance). The pipeline can poll these feeds every morning to collect headlines, article URLs, and succinct descriptions using an RSS reader library (such as Python's feedparser). RSS is easy to use and real-time, but keep an eye out for any feed availability changes or outages.
- **Web APIs & Aggregators:** Think about using news aggregator APIs if RSS is restricted or not available. For example, JSON of the day's top finance headlines from multiple sources can be returned by NewsAPI.org or similar services (though free tiers may have limits). Certain sources have their own APIs, such as Google News RSS via custom queries or Yahoo Finance's unofficial stock news APIs. An example from a related project obtained a list of pertinent news links every day by converting a Google News query into an RSS/JSON feed. Aggregator APIs make multi-source coverage easier, but they frequently have usage fees or require API keys.
- **Web Scraping (Fallback):** A scraping method can obtain content from news webpages in the absence of feeds or APIs. Articles can be parsed and HTML can be retrieved using programs like BeautifulSoup or Playwright. Scraping, on the other hand, is more vulnerable to site modifications and must adhere to Terms of Service. It serves as a backup in case official feeds are insufficient.

Selecting Sources: It's wise to use **multiple sources** for diversity and redundancy. Yahoo Finance and Investing.com are good starting points as they cover broad market news and have public feeds. Reuters or AP News provide straight factual reporting (often via RSS) which suits our factual tone. Be sure to target sources that are **free to access** (paywall-free) and update early in the day. The system can pull, say, 10–20 headlines across sources and filter or deduplicate them.

Data Retrieved: For each news item, we ideally want the title, a short description, and a link to the full article. RSS feeds typically provide these. Some feeds might include a short summary or the first paragraph, which can aid our summarizer. If only titles are available, the pipeline may fetch the article page to get full text for summarization. Keep track of publication time to ensure freshness.

Scheduling: A simple cron job or scheduled cloud function can trigger the news fetch every weekday morning (or whatever frequency desired). The fetched items then pass to the summarization stage.

2. AI Summarization of Articles

The subsequent module creates concise, factual summaries (~150 words) of each article after it has been identified. Condensing in-depth news into succinct reports without adding any opinion or analysis is a crucial step. Two primary methods of summarization are available in modern AI:

Abstractive summarization (LLM rewrite): A large language model takes the main ideas of a piece and puts them into fresh wording. While the output often flows better, almost like spoken audio, there is a chance of subtle inaccuracies or slant creeping in without tight controls. To reduce risk, set clear rules - stick to facts only, keep it short, avoid personal views - and check results afterward. You might pick cloud-based models through an API for stronger performance, though they come with usage fees and speed caps, or run smaller tools such as BART or PEGASUS locally instead.

Some tools pick full sentences straight from a piece of text - like how Sumy uses TextRank. These choices stay close to the original, so they do not twist meaning. Yet the result might seem uneven or run on too long. When key lines include heavy terms or quoted chunks, the summary could sound stiff. How it reads depends on what those top sentences actually say.

A solid move? Mix the two methods. Pull key facts first - names, dates, steps, amounts - then feed just those pieces to a language model. Let it shape them into one tight summary. Facts stay sharp. Words flow better. The mix works.

Summarization Tools Comparison

Tool	Pros	Cons
LLM API (GPT-4/GPT-3.5) Abstractive – e.g. OpenAI, Azure OpenAI	Very fluent, can incorporate instructions (tone, length). Can summarize even long texts via chunking.	Potential for subtle added phrasing or minor errors if not carefully prompted. Requires API cost; must handle long inputs (may need to truncate or summarize in parts).
Summarizer Model e.g. BART-large-CNN, PEGASUS on HuggingFace pipelines	Runs locally (no API fees), and models trained on news have some factuality checks. No risk of external API outage.	Requires decent compute (a GPU for speed). Quality might be slightly lower than GPT-4 for coherence. Still might omit context if article is very long (may need to feed only relevant part).
Extractive Algorithm e.g. TextRank (Python Sumy), or services like TLDR This (which extract key sentences)	Guaranteed no hallucination – uses original sentences, preserving exact facts. Fast and can be done with minimal CPU.	Summaries may be less readable (list of disjointed points). Might exceed 150 words unless pruned. Requires post-editing to ensure smooth transitions.
Hybrid (AI + extraction) e.g. Use an extractive summary as input to an LLM: “Rephrase these key points into a 150-word summary.	Combines factual grounding with fluent narration. LLM stays constrained by provided points.	More complex pipeline. Still need to trust LLM not to stray beyond provided points. Double processing adds latency.

When viewed up close, using a single-step method with large language models can work efficiently if the instruction is clear. Consider providing guidance like this: "Act as a helper creating a straightforward news recap based solely on the provided piece; avoid personal views, guesses, or suggestions; stay within set boundaries." Often, what follows is a condensed form of the source material, tightly focused. A properly shaped request helps shape the output. Should accuracy matter more, consider having the model pull out three to five factual highlights before building any account around them. Tools exist that specialize in generating summaries stripped of interpretation, focusing solely on what can be confirmed. One suggestion these systems offer involves crafting responses using nothing but evidence found directly in original material. Analysis, personal views, or guesses stay absent by design. Such methods aim to reflect content without shaping it

Factual Consistency: It is paramount that the summary not introduce any commentary. If using abstractive methods, one mitigation is to compare the named entities and figures in the summary back to the article to catch errors. Another is to prefer models known for factual consistency – research on improving factual consistency in news summarization has led to better pre-training strategies. If available, one could use an **AI fact-checker** or simply double-check any numbers/dates in the summary against the source.

Output format: We aim for each summary to be a short paragraph (~3-5 sentences, <= 150 words). It should read like a wire report: e.g. *"BigBank Inc. reported a 10% rise in Q4 profits, beating analyst expectations. The CEO attributed the growth to increased loan demand. Shares of BigBank rose 2% in premarket trading on the news. In other developments, the Federal Reserve signaled a possible rate pause, which bolstered financial stocks across the board."* No sentence should contain an opinion—by sticking to who, what, where, when facts, we comply with journalistic neutrality. (Reuters' handbook explicitly instructs that reporters must not give personal opinions and should stick to reported facts.)

After summarization, the system will have a list of summary candidates (perhaps 5–10, assuming that many articles were fetched). We then move to selecting which ones will go into the day's podcast.

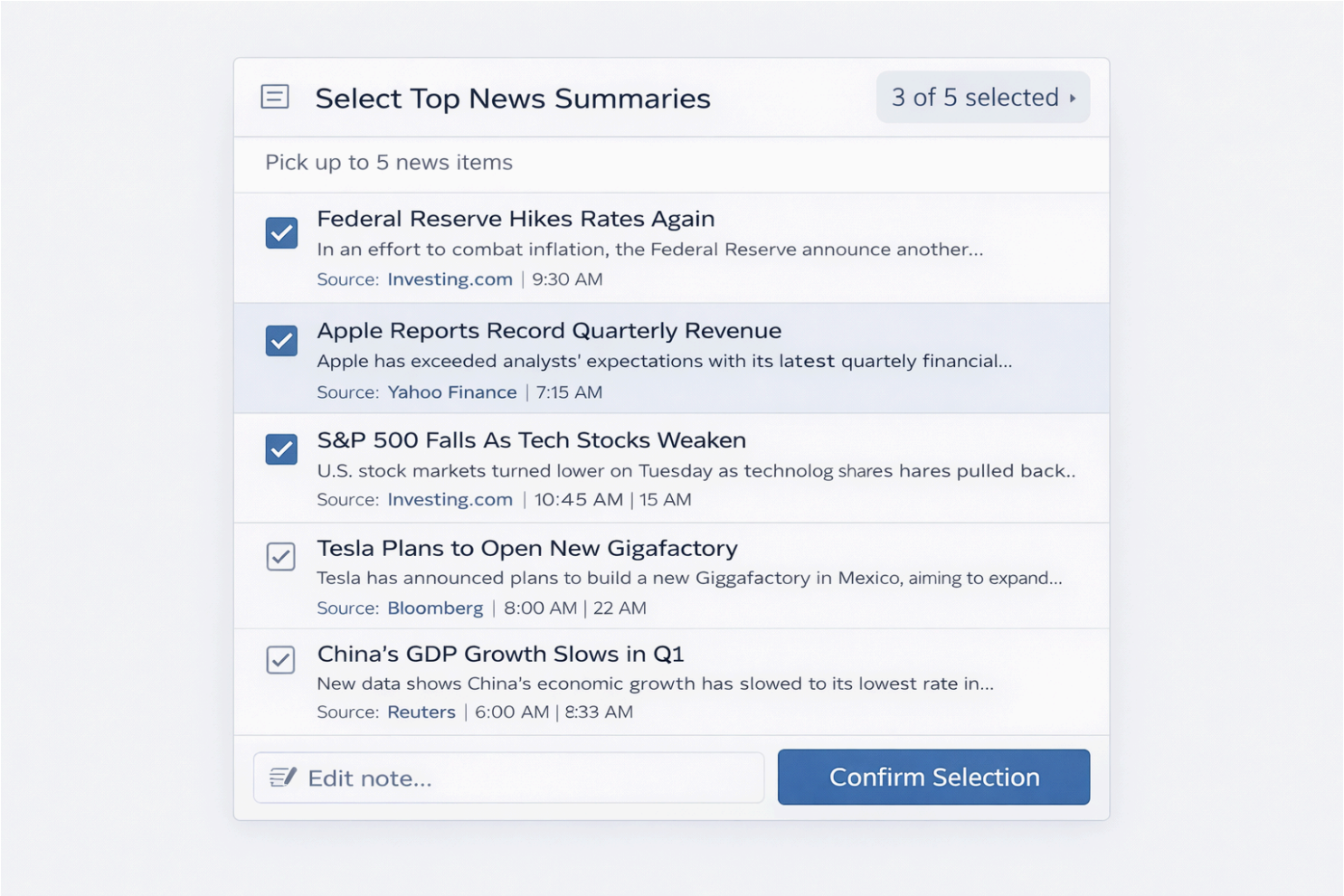
3. User Selection of Top Summaries

Despite automated systems, human judgment helps identify standout news each day. A streamlined tool allows podcast creators to examine machine-made summaries. From these, roughly three to five pieces may be marked for inclusion. Each morning, compiled results appear via email or online page, showing headlines and sources. Interaction involves selecting preferred entries through simple marks or clicks. Chosen articles then move forward into drafting scripts.

Functional Requirements

The module must:

- Display a list of AI-generated summaries (typically 5–15 items)
- Show key metadata for each summary:
  - Headline
  - Source name
  - Publication time/date
- Allow the user to select exactly or up to 3–5 items
- Persist the selected summaries for downstream processing
- Require minimal interaction time (≈ 30–60 seconds)



Implementation Options (Feasibility)

Option 1: Lightweight Web UI (Most Practical)

- **Tools:** Streamlit, Flask + HTML, or FastAPI + simple frontend
- **Why feasible:**
  - Minimal UI logic
  - Can be built in a few hours
  - Runs locally or on a small server
- **Best for:** Daily producer workflow

Option 2: Spreadsheet-Based Selection

- **Tools:** Google Sheets or CSV
- **Workflow:**
  - Summaries auto-filled into rows
  - User marks “Include = TRUE”
- **Pros:** No custom UI needed

- **Cons:** Less polished, slower feedback loop

Option 3: Command-Line Interface (CLI)

- **Workflow:**
  - Script prints numbered summaries
  - User inputs numbers (e.g., 1, 3, 5)
- **Pros:** Very fast to implement
- **Cons:** Less intuitive, not visual

Option 4: Messaging / Chat-Based Approval

- **Tools:** Slack bot, Telegram bot
- **Workflow:**
  - Summaries sent as messages
  - User replies with selection
- **Pros:** Extremely low friction

Possible automation: If desired, this step could be semi-automated by having an AI rank the news by significance (perhaps based on article prominence or stock impact). However, given the user's domain

knowledge and responsibility for content, keeping a human in the loop is wise. It ensures editorial control crucial for not missing a big story or including something off-topic.

From a design perspective, this module is straightforward. It's more about UI/UX than AI: no fancy tools needed beyond presenting text and capturing selection. For example, one could use a simple GUI library or even send the summaries to the user's phone via a messaging bot for quick approval.

Data passed on: The output of this step is the subset of summaries (3–5) that will form the episode. These will be concatenated or further processed into the script. We also might keep their original source info if we plan to mention source names or dates in the script.

4. AI Generation of Podcast Script (5 minutes)

The Podcast Script Generation Module transforms a small, curated set of AI-generated news summaries (typically 3–5 items) into a cohesive, 5-minute spoken news script suitable for direct narration via text-to-speech (TTS) or a cloned voice.

This module is responsible for:

- **Structuring content into a radio-style narrative**
- **Maintaining consistent tone, pacing, and format**
- **Ensuring editorial neutrality and factual integrity**
- **Producing output that sounds natural when spoken aloud**

Unlike summarization, which compresses information, this stage focuses on flow, clarity, and listenability.

Target Length and Timing Constraints

Spoken Audio Constraints

- Typical professional news narration rate: 140–160 words per minute
- Target duration: ~5 minutes
- Resulting script length: ~700–800 words

Allocation Example

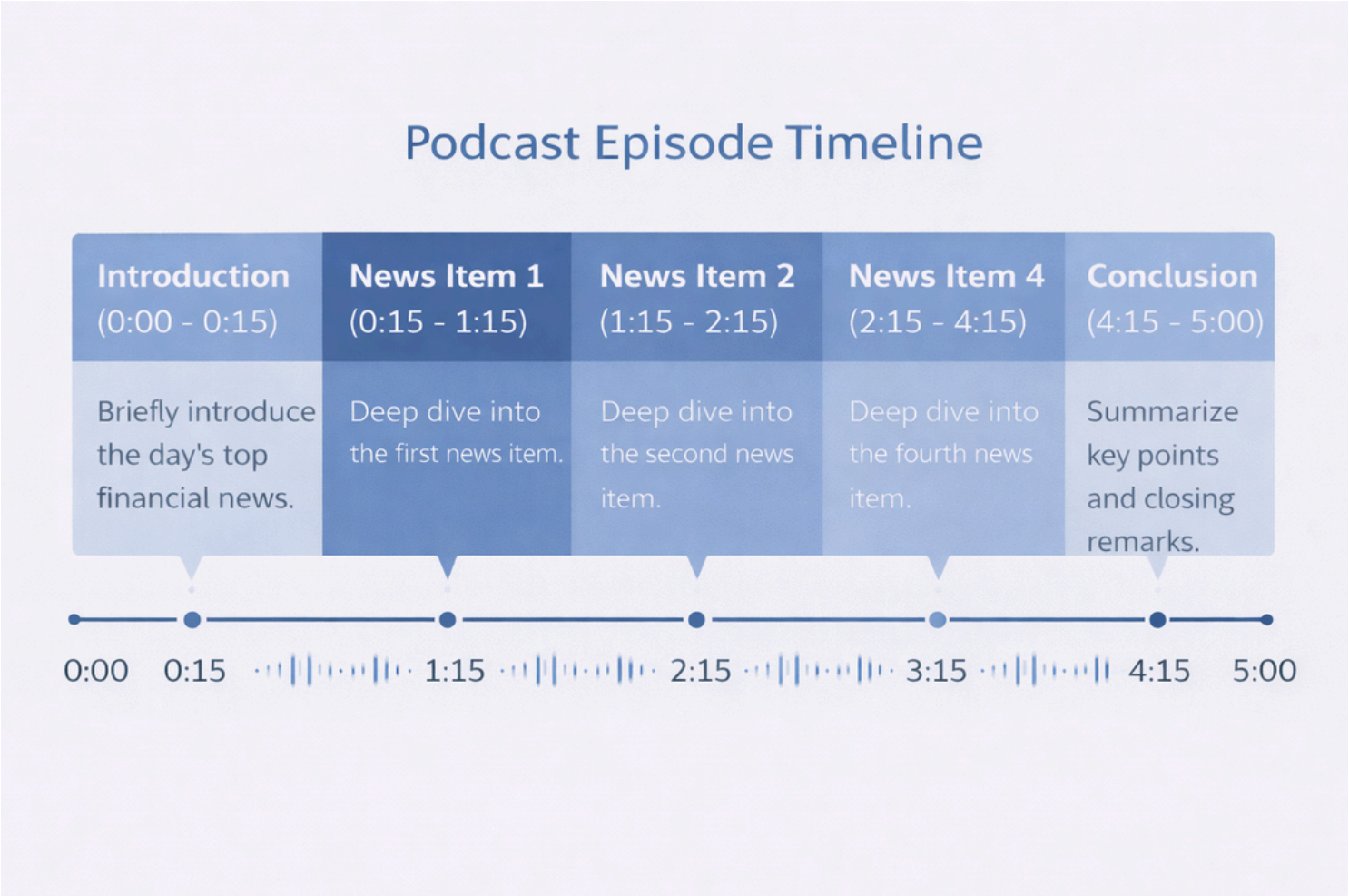
For a 4-story episode:

- Introduction: ~80–100 words
- Each story: ~140–160 words
- Transitions between stories: ~20–30 words each
- Closing outro: ~40–60 words

This allocation ensures:

- Balanced coverage
- Predictable episode length
- Clean TTS alignment (important for subtitles and video sync)

Script Structure (Enforced Template)



The script follows a fixed structural outline to ensure consistency across episodes:

1. Introduction

Purpose:

- Greet the listener
- Identify the podcast and date
- Briefly preview the stories

Example structure:

“Hello, and welcome to your daily financial news update for March 5th. Here are today’s top market stories…”

2. Story Blocks (Repeated 3–5 Times)

Each story block contains:

- Headline restatement (spoken, not read verbatim)
- Key facts and figures
- Attribution to source where appropriate
- Smooth transition to the next story

Transition phrases are intentionally simple:

- “In other news…”
- “Meanwhile…”
- “Turning to the markets…”

3. Closing

Purpose:

- Signal completion
- Thank the listener
- Maintain brand consistency

Example:

“That’s today’s financial news roundup. Thanks for listening, and we’ll be back tomorrow with the latest updates.”

Tone and Style Control

Required Tone Characteristics

- Professional
- Neutral
- Informative
- Calm and measured

The script explicitly avoids:

- Opinions
- Predictions
- Investment advice
- Emotional language

This mirrors the style of financial radio news (e.g., Bloomberg Radio, NPR Marketplace).

Linguistic Design for Audio

Because the output is spoken, the script:

- Uses shorter sentences
- Avoids dense numeric clusters
- Rephrases statistics conversationally

Example:
“The S&P 500 rose 1.23 percent following earnings.” **FALSE**
“The S&P 500 rose just over one percent following strong earnings reports.” **TRUE**

Using AI to Generate the Script

Model Role

A large language model (LLM) acts as a script compositor, not a researcher. It is constrained to:

- Only use facts from the provided summaries
- Focus on phrasing, ordering, and transitions

Input to the Model

The model receives:

- The selected summaries (structured text)
- Episode metadata (date, episode type)
- Strict behavioral instructions

Example Prompt (Engineered)

You are a professional script writer for a daily financial news podcast.
Using only the information provided below, write a 5-minute podcast script (~750 words).

Requirements:

- Include a short introduction, transitions between stories, and a closing
- Maintain a neutral, professional news tone
- Do not add opinions, advice, predictions, or new facts
- Use spoken-friendly language suitable for audio narration

Here are the selected news summaries:
[Summary 1]
[Summary 2]
[Summary 3]
[Summary 4]

This prompt design ensures deterministic structure with creative phrasing, which is ideal for audio content.

Consistency Across Episodes

Structural Consistency

- Same intro style
- Same transition patterns
- Same outro format

This consistency:

- Builds listener familiarity
- Simplifies TTS and video alignment
- Makes the podcast feel “produced,” not generated

Style Enforcement Techniques

- Fixed intro/outro templates
- Prompt instructions forbidding stylistic drift
- Optional post-generation validation checks (length, tone)

Length and Quality Control

Word Count Enforcement

- Automatically count generated words
- If outside range (e.g., >850 words):
  - Re-prompt AI to compress
  - Or programmatically trim lower-priority sentences

Factual Integrity Safeguards

- The script generator cannot introduce new data
- All content traces back to validated summaries
- Minor connective phrases are allowed, but not speculation

Tooling and Feasibility

Primary Options

- OpenAI GPT-4 / GPT-4-class models: highest quality
- GPT-3.5-class models: lower cost, acceptable quality
- Anthropic Claude / Cohere / AI21: viable alternatives

Why Prompt-Based Generation Is Sufficient

- News scripts follow predictable structure
- No need for fine-tuning initially
- Easy to iterate and control via prompts

Alternative (Non-AI-Heavy) Approach

A template-based method could insert summaries into pre-written blocks.
However:

- It sounds robotic
- Lacks natural transitions
- Reduces perceived quality

A hybrid AI approach provides polish without sacrificing control.

Optional Extensions (Not Core but Compatible)

Human Review

- User may optionally edit script text before finalization
- Especially useful for brand voice tweaks

Multilingual Output

- Translate final script after generation
- Re-use same structure
- Pair with multilingual TTS voices

Output of the Module

The final output is:

- **A fully formatted script (plain text)**
- **Approximately 750 words**
- **Ready for direct narration**
- **Suitable for subtitle generation and video synchronization**

5.Voice Cloning & Narration Synthesis

This module turns the completed podcast script into spoken audio, matching the user's voice through a pre-trained neural **TTS/voice-cloning system**. The voice is created once (enrollment/training), then reused daily for fast generation.

If the user chooses not to clone their voice, the same pipeline can fall back to a **high-quality stock neural voice** (news-anchor style) for reliability and speed.

1) Reliable voice cloning

A voice clone is “reliable” in production when it consistently delivers:

- **Identity similarity:** the output clearly resembles the user (timbre/accent/voiceprint)
- **Stability:** no sudden voice drift, warble, metallic artifacts, or random tone changes
- **Intelligibility:** accurate pronunciation (especially tickers, company names, finance terms)
- **Consistency across scripts:** similar pacing, energy, and clarity across episodes
- **Operational reliability:** predictable latency, repeatable results, no frequent failures

2) Third-party services that provide voice cloning

Voice Cloning Tools Comparison

Voice Cloning Tools	Sample Data Required / Duration	Notes on Quality & Usage
ElevenLabs (Instant)	~1–3 minutes of clear speech audio Immediate clone (minutes)	Very easy setup via web; surprisingly high-quality for minimal data (captures voice timbre well). Limited emotional range with small sample. Good for most needs. Multi-lingual speaking supported. Make sure sample has no background noise (studio quality helps).
ElevenLabs (Professional)	≥ 30 minutes (up to a few hours optimal) ~2–4 hours for model training	Yields a near-perfect replica, with finer intonation and pacing. Suitable if you have a lot of voice recordings (e.g., previous podcasts) to feed in. Requires signing usage agreement and completing a voice captcha (ensuring you are the voice owner). Best for long-term professional content.
Resemble AI (Standard)	≥ 5–10 minutes to start (50 sentences min), but 45+ minutes recommended A few hours for training	High-quality custom voice with ability to adjust style. More data improves naturalness significantly. Provides an API for integration. Offers features like emotion tags, but cost can be significant for heavy use (pricing per generated second).
Azure Custom Neural Voice	≥ ~30 minutes (300+ utterances) required; 2–3 hours of audio ideal ~20–40 compute hours training	Produces one of the most natural synthetic voices. However, access is gated (must apply as a vetted use case due to potential misuse). Once model is made, inference is real-time via Azure cloud. Good option if enterprise-level quality and support are needed.
Pre-made TTS Voice (no cloning)	N/A (use built-in voice) Immediate (no training)	(Examples: Amazon Polly, Google Cloud TTS, Microsoft Cognitive Services voices, or commercial voices from Murf.ai, WellSaid Labs, etc.) Many stock voices are very natural now, including neural voices that have expressive styles. No personal touch, but simplest. Could use a voice that fits a news anchor persona. Often support SSML for fine control.

For an automated pipeline that produces daily episodes, the top priorities are **integration reliability (API), predictable quality, low onboarding friction, and cost scalability**. Based on these criteria, **ElevenLabs Instant** is the most practical choice for a first version: it can be set up

quickly with a short clean sample, generates natural speech suitable for news narration, and is easy to automate. However, if the goal is a long-term branded voice with maximum consistency and natural pacing, moving to **ElevenLabs Professional** (or a higher-data training option like Resemble with a larger donor dataset) becomes worthwhile after the workflow is validated. If the system is intended for enterprise deployment, where governance, approvals, and controlled usage matter most, **Azure Custom Neural Voice** is the strongest fit despite higher setup complexity. Finally, to ensure the pipeline never fails end-to-end, a **pre-made neural TTS voice** should be kept as a fallback for outages, quota limits, or voice model issues.For an automated pipeline that produces daily episodes, the top priorities are **integration reliability (API), predictable quality, low onboarding friction, and cost scalability**. Based on these criteria, **ElevenLabs Instant** is the most practical choice for a first version: it can be set up quickly with a short clean sample, generates natural speech suitable for news narration, and is easy to automate. However, if the goal is a long-term branded voice with maximum consistency and natural pacing, moving to **ElevenLabs Professional** (or a higher-data training option like Resemble with a larger donor dataset) becomes worthwhile after the workflow is validated. If the system is intended for enterprise deployment, where governance, approvals, and controlled usage matter most, **Azure Custom Neural Voice** is the strongest fit despite higher setup complexity. Finally, to ensure the pipeline never fails end-to-end, a **pre-made neural TTS voice** should be kept as a fallback for outages, quota limits, or voice model issues.

3) Required training samples (how much audio you need)

Voice cloning quality is primarily driven by **audio quality and phonetic coverage**, not just raw minutes.

Recommended sample tiers

Tier 1 — Fast start (good enough for MVP)

- **1–3 minutes** of clean speech (instant cloning level)
- Pros: quickest setup, excellent for a prototype
- Cons: reduced emotional range; edge-case pronunciation errors more likely

Tier 2 — Stable production

- **10–30 minutes** of clean, varied recordings
- Pros: better stability, fewer artifacts, better pacing consistency
- Cons: more recording time

Tier 3 — Broadcast-grade

- **30–60+ minutes** (or more) with controlled scripts
- Pros: best naturalness, better handling of different sentence types
- Cons: biggest time investment, sometimes higher cost tiers

What makes a “good sample” (this matters more than minutes)

To maximize reliability, record samples that are:

- **Noise-free** (quiet room, no echo, no music)
- **Consistent microphone distance** (no volume pumping)
- **Single speaker only** (no overlapping voices)
- **Varied content**: numbers, proper nouns, acronyms, finance vocabulary
  Example scripts should include: “S&P 500,” “NASDAQ,” “Dow,” “basis points,” “earnings,” “guidance,” “quarter-over-quarter,” company names and tickers.

4) Expected quality and limitations (what can go wrong)

Expected quality (realistic expectations)

With modern managed services:

- A 1–3 minute clone can sound **surprisingly close** to the user’s voice for neutral narration.
- With 30+ minutes, the clone usually becomes **more stable**, more natural, and better at maintaining consistent pacing.

Common limitations

A) Pronunciation issues

- Tickers, uncommon names, and acronyms may be mispronounced.
- Mitigation: pronunciation dictionary / hints, phonetic spellings, or SSML where supported.

B) Emotional range and expressiveness

- Short training data often yields “neutral” delivery that may sound slightly flat.
- Mitigation: use a “newscaster” style setting if available; optionally record a dataset with varied intonation.

C) Audio artifacts

- Metallic ringing, robotic consonants, or warbly vowels can occur when the sample has noise, compression, or echo.
- Mitigation: clean recording + consistent gain + WAV/FLAC input.

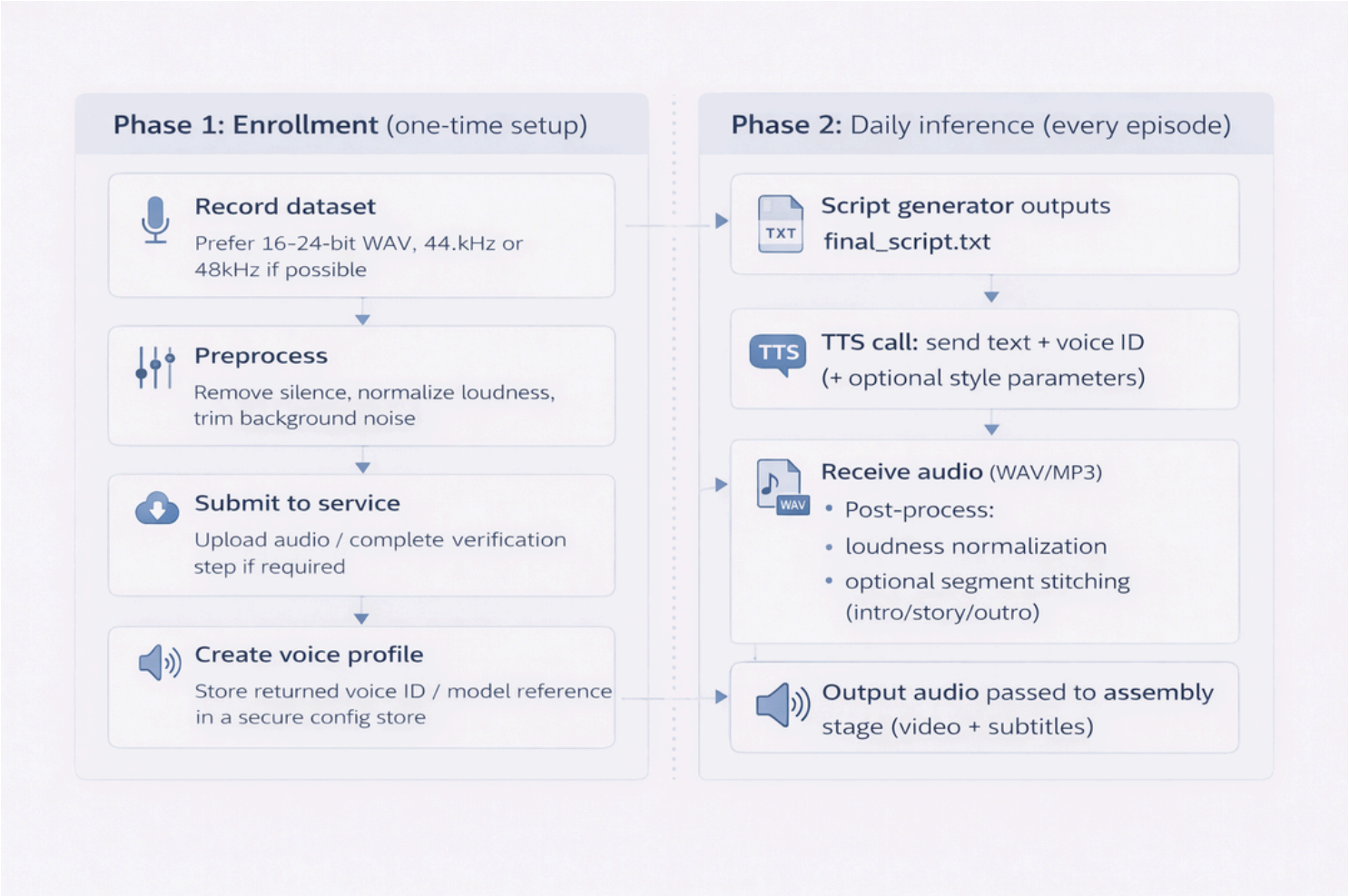
D) Voice drift / instability

- Some systems can shift tone slightly across long passages.
- Mitigation: split script into segments (intro + story blocks + outro), synthesize separately, then stitch.

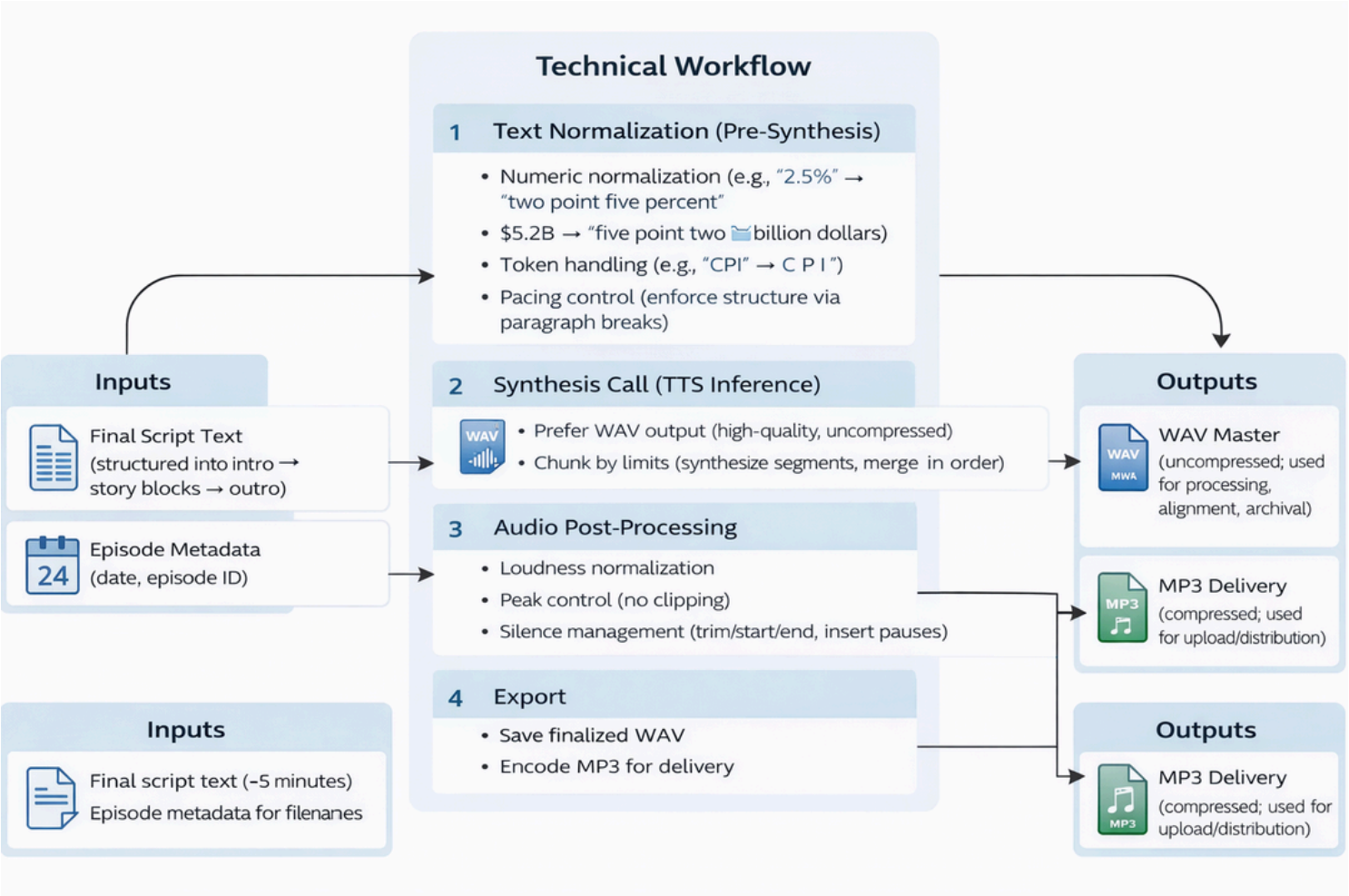
E) Legal/ethical + consent

- Voice cloning can be misused; reputable providers require consent and prohibit cloning without rights.
- Mitigation: only enroll the user's own voice; keep an audit log of enrollment and usage.

5) Engineering workflow: how to clone and use the voice daily



6. Text-to-Speech (TTS) Audio Generation Module



Engineering decisions and trade-offs

- **WAV-first pipeline:** WAV is preferred internally because it is uncompressed, easier to process, and reduces artifacts when normalizing and concatenating. MP3 is reserved for final delivery.
- **Single-pass vs chunked synthesis:** single-pass yields the most natural continuity but may fail under API limits; chunked synthesis improves reliability and allows partial retries at the cost of requiring careful concatenation and pause control.
- **Deterministic formatting:** strict text normalization reduces pronunciation variance, improves alignment for subtitles, and lowers the chance of TTS failures caused by unusual characters.

Reliability and failure handling

- **Retry at the segment level:** if one chunk fails, only that chunk is regenerated (avoids re-running the full episode).

- **Validation gates:** the module verifies duration is plausible (roughly proportional to word count), checks file integrity, and confirms loudness is within the expected range before passing to video assembly.
- **Graceful degradation:** if synthesis repeatedly fails due to formatting, the module falls back to a simplified text variant (removing special characters and reducing punctuation) and regenerates.

Deliverable claim (what the module guarantees)

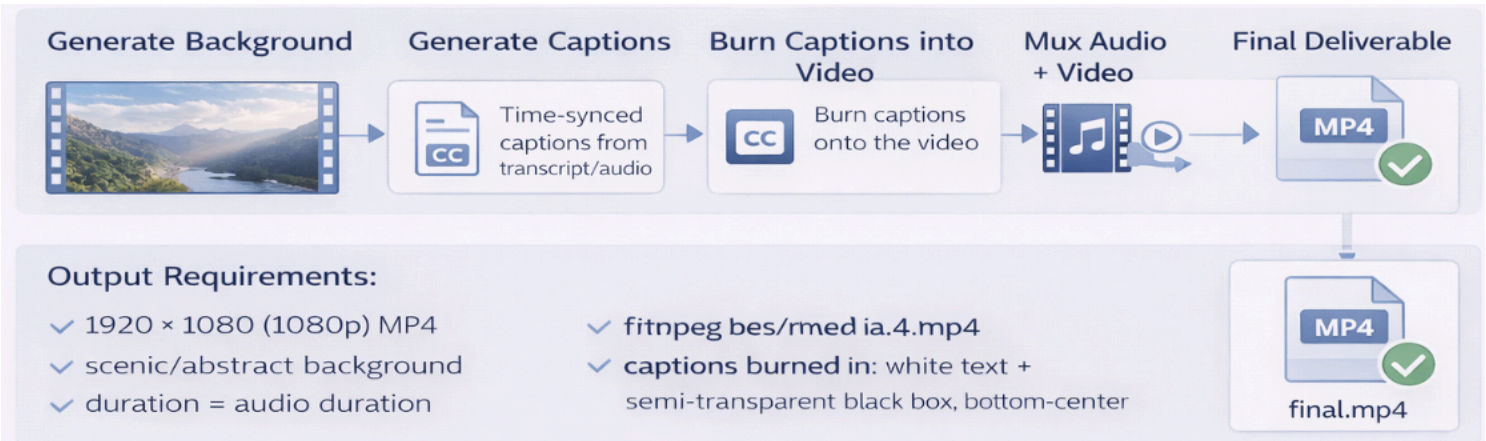
The TTS module reliably converts the final script into **WAV and MP3** with consistent loudness and controlled pauses, producing audio that is ready for subtitle timing and 1080p MP4 assembly.

7. Background Video Generation & Captioning

Goals:

Convert the narrated podcast audio into a **YouTube-ready 1080p MP4** by:

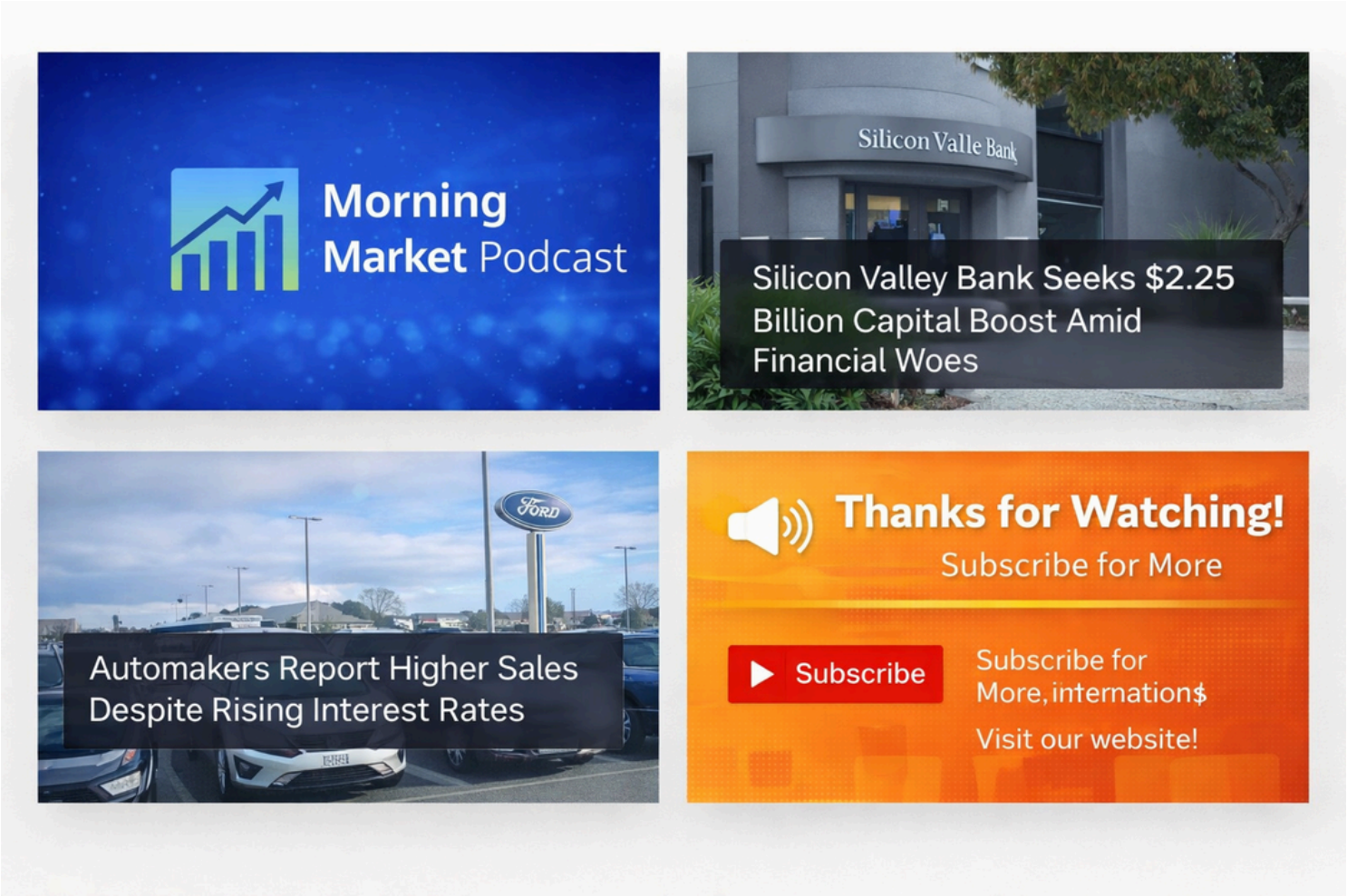
- generating a **scenic or abstract background video**
- producing **time-synced captions** from the transcript/audio
- **burning captions into the video** (white text + semi-transparent black box)
- muxing the **narration audio + video** so duration matches perfectly



A) Automatically creating background video

Option	Concept	Pros	Cons
1) Curated stock footage + loop/trim (Recommended)	Maintain a folder of **10–50 pre-approved scenic/abstract clips**. Each episode selects one (random/rotate) and **loops/trims** to match audio duration.	Most professional look; low compute; consistent; no AI artifacts; easy to scale by adding clips.	Requires building/maintaining a library + licensing; one-time normalization work (1080p, FPS, bitrate).
2) Ken Burns effect (zoom/pan on image)	Use a high-res image (skyline/abstract/market themed) and apply slow **zoom + pan** for the full episode duration.	Extremely simple; very reliable; lightweight; clean if image quality is strong.	Can feel repetitive and less “premium” than video unless images rotate frequently.
3) Audio-reactive waveform/spectrum visualizer	Generate a waveform/spectrum animation from narration audio over a subtle background.	Visually matches audio; no external footage required; classic “podcast visualizer” style.	Can look generic without good styling; still needs captions; may distract if overly active.
4) Procedural abstract animation	Programmatically generate animated gradients/geometric patterns.	No licensing issues; always unique; full creative/brand control.	Higher dev/design effort to look premium; can look like a “tech demo” if not tuned.
5) AI video generation (e.g., Sora) as background source	Generate **10–30s abstract/scenic AI clips** and **loop/stitch** to reach full duration, or periodically refresh the clip library.	Fresh visuals; potentially cinematic; reduces repetition; good “modern AI” aesthetic.	Less deterministic (variable quality); higher cost/latency; possible artifacts; must constrain prompts (avoid logos/text/faces) and still loop/trim for exact duration.

Background Video Generation:



To produce a YouTube-ready version of the podcast, the pipeline generates a **1080p (1920×1080) MP4** by pairing the narration audio with a **scenic or abstract background video** that is **looped/trimmed to match the exact audio duration**. It then creates **time-synced captions** from the transcript/audio and **burns them into the video** as **white text inside a semi-transparent black box** at the bottom center for readability. Finally, an automated assembly step muxes **video + captions + narration** into a single deliverable (final.mp4). For reliability and daily automation, curated stock loops (or Ken Burns) are the default, while **AI video generation (e.g., Sora)** can be treated as an optional source for refreshing background clips when higher novelty is desired.

B) Caption Overlay From Transcript

The system should use an **open captions** (burned directly into the video frames) to ensure the text is always visible on platforms like YouTube. The design targets **high readability on mobile** and a consistent “news subtitle” look:

- **White text**
- **Semi-transparent black caption box**
- **Bottom-center placement**

- **1–2 lines maximum per caption**
- Typical line length: **~32–42 characters per line** (balanced for phones and TV)

Method	How it works	Strengths / Output	Limitations / Best fit
1 (Most reliable): ASR on final audio	Speech-to-text alignment is run on the **final narrated audio** to generate timecodes.	Matches real TTS pacing; sync aligns with actual delivery. output : subtitles.srt or .ass	ASR can mishear tickers/names → best default, with manual cleanup or fallback.
2: Forced alignment (known script + audio)	The **known script text** is aligned to the generated audio to compute timestamps.	Extremely precise when text matches; great for tickers and proper nouns. output: subtitles.srt or .ass	Requires alignment tooling; sensitive if spoken output differs from script → best when finance terms must be perfect.
3: TTS-provided word timings (if supported)	Uses vendor-provided word/phoneme timing metadata from the TTS synthesis step.	Potentially the cleanest/most accurate when available. output : Timing JSON → converted to SRT/ASS	Not consistently available in public APIs; still needs chunking into readable caption blocks → best only if vendor supports it reliably.

Wrap-up: Caption timing can be generated three ways: the most reliable default is **ASR on the final narrated audio** because it matches the real pacing and pauses. If precision on **tickers and proper nouns** matters most, **forced alignment** using the known script provides tighter control. Where available, **TTS-provided word timings** can streamline the pipeline, but they're not consistently supported and still require chunking into readable subtitle lines.

C) Automated Audio + Video Assembly

Recommended pipeline order

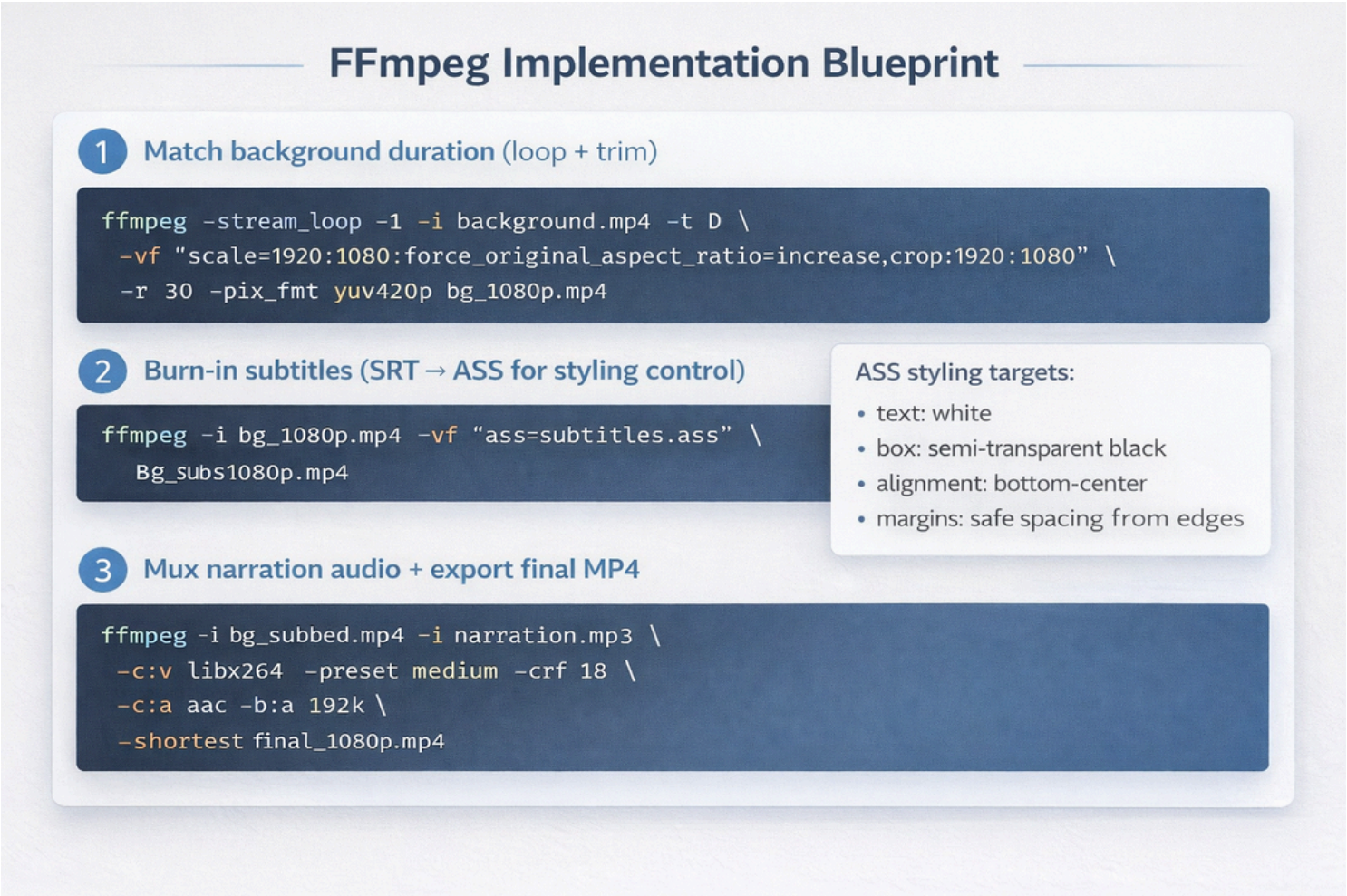
To prevent drift and end-of-video issues, the cleanest workflow is:

1. Measure **audio duration** (seconds)
2. Generate the background video to the same duration → loop/trim so **video length == audio length**
3. Generate captions (.srt or .ass) aligned to the audio
4. Burn captions into the video
5. Mux narration audio into the captioned video
6. Export the final **1080p MP4**

This prevents:

- subtitles continuing after audio ends
- video longer/shorter than audio
- black frames at the end

D) FFmpeg Implementation Blueprint



E) Practical Recommendation

For daily automation with minimal operational risk, the strongest configuration is:

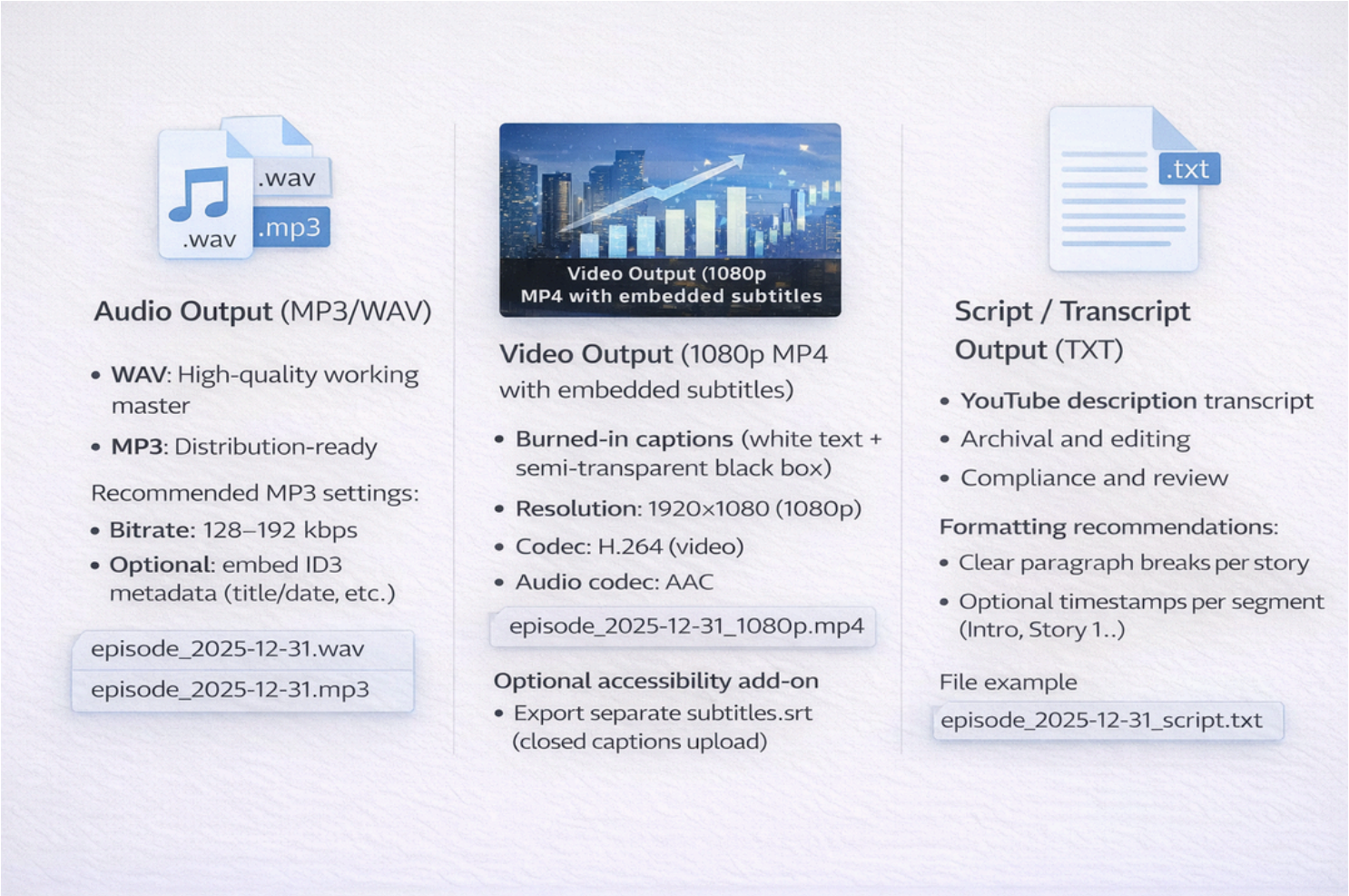
- **Background:** curated scenic/abstract stock clips (loop/trim)
- **Captions:** SRT via ASR alignment (or forced alignment if tickers must be perfect)
- **Assembly:** FFmpeg end-to-end (scale → burn subtitles → mux audio → export)

This approach meets all output requirements while remaining reliable, repeatable, and easy to maintain.

8. Output Packaging

The export module packages the final deliverables in standard formats so they are **ready for upload, distribution, archiving, and reuse** (podcast platforms + YouTube).

A) Final Deliverables



B) Output Folder Structure

To keep exports organized, the pipeline stores outputs per episode:

- /exports/2025-12-31/
  - episode_2025-12-31.mp3
  - episode_2025-12-31.wav
  - episode_2025-12-31_1080p.mp4
  - episode_2025-12-31_script.txt
  - *(optional)* episode_2025-12-31_subtitles.srt

C) Final Validation Before Marking “YouTube-Ready”

Before labeling files final, the export module verifies:

- **Audio duration = video duration**
- Video is **1080p**
- Captions are visible, readable, not cut off at bottom
- Audio loudness is normalized (no clipping, not too quiet)
- Output filenames follow consistent episode naming

D) Feasibility Notes (Production Reality)

This packaging step is highly feasible with current tooling:

- MP3/WAV conversion and tagging is standard
- 1080p MP4 export with burned-in captions is routine via FFmpeg
- Script export is trivial once script generation is complete

E) Optional Delivery Methods (nice-to-have)

Depending on deployment, the system can deliver outputs via:

- cloud storage (Drive/S3)
- email link to files
- direct upload via YouTube API *(future enhancement)*

9.Future Enhancements

This pipeline can be further enhanced in various ways:

- **Automate Story Selection via AI**: While a human touch is great, if one wanted full automation (for days user is unavailable), an AI could rank or filter summaries. For example, use a sentiment of significance analysis – look for certain keywords (like “record high” or large numeric changes) to decide importance. Or simply always pick the top 5 by some source priority. This is risky (it might miss context), but could serve as a fallback. One could also incorporate social media trends (which finance topic is buzzing today?) to pick stories.
- **Content Variety**: The pipeline currently just reads news. In the future, it might include a short **“market data” segment – e.g., key index levels or stock movers**. This could be automated by querying an API for market data (Yahoo Finance API or others) for the day’s closing prices and having the script generator include a line like “At market close, the Dow was X, the Nasdaq Y…”. This adds value for listeners. Similarly, a quick weather or crypto update could be integrated if relevant to the theme.
- **Dynamic Visuals per Story**: Instead of one generic background, the video could show a relevant image or short clip for each story. For example, when talking about BigBank, show their logo or a stock chart (ensuring it’s allowed). Or for a story about oil prices, show an oil rig image. One could pre-compile a small library of thematic images (tech, finance, markets, etc.) and have the script tagged by category to pull the appropriate visual. The editing would then involve switching visuals at story transitions (which is doable by splitting the video at those timecodes and inserting different images). This complicates video generation but would make the video more engaging.
- **Higher resolution or alternative formats**: Possibly output also a vertical video for platforms like TikTok or Instagram (with big captions). That would require reformatting the content to shorter

snippets. Probably not in scope for the daily longform, but it's an idea to repurpose content.

- **Interactive Translations**: If the pipeline were used by a bilingual creator, an automatic translation of the script followed by using a different voice (either the user's clone speaking the other language, if supported, or a native speaker TTS) could produce a second video in another language. This opens to a larger audience.
- **Real-time or On-demand Generation**: Currently it's scheduled daily. One might consider an on-demand mode (e.g., type a date or hit a button and it generates that day's episode). Or even real-time updates (though a 5-min daily summary is by nature a batch process, not continuous streaming).
- **Optimize Cost**: Using local open-source models (for summarization, for TTS) could eliminate API costs. For example, running a local instance of Whisper for transcriptions and using smaller transformers for summarizing means after initial setup, there's no recurring fee. But for highest quality, using the big cloud models (GPT-4, ElevenLabs, etc.) might be worth the cost for a polished product. The user can decide based on budget. In many cases, the costs are modest (a 750-word script via GPT-4 is maybe $0.02, and a 5-min TTS on ElevenLabs perhaps a few cents as well, making each episode well under $0.50 in API usage).
- **Monitoring and Logging**: Implement logs for each run (which stories were chosen, any errors). This helps maintain reliability. If a step fails (e.g., voice API is down), the system could notify the user or retry. As this might be running unattended each morning, robustness is key.
- **User Voice Updates**: Perhaps schedule re-cloning the user's voice periodically if needed (maybe if their voice changes or they want a different style). Some services allow adding more data to refine the voice over time. Resemble, for instance, allows incremental improvement by recording more. If the user records real podcast episodes, those could feed back to improve the clone.
- **Ad or Personalization Insertion**: A future extension could allow adding a short ad read or personalized message in the intro or outro. The script generator could be prompted to always include e.g. "This podcast is for informational purposes only, not investment advice" disclaimer (which might be wise legally). Or a custom sign-off tagline.

In summary, the pipeline is quite extensible. Even in its basic form, it achieves the heavy lifting of content creation. Going forward, as AI models improve (e.g., more factual summarization, even more realistic voice cloning requiring only seconds of data, etc.), the quality and ease will only get better.

Conclusion

With today's AI technology, creating an automated daily financial news podcast pipeline is quite possible. RSS news feeds, NLP summarization, scriptwriting LLMs, voice cloning, and video assembly tools can all be combined to create a fully produced video in a matter of minutes with little human intervention. In order to maintain the final product's editorial integrity and accuracy, the system we described strikes a balance between automation and a small amount of user control (story selection and content review).

Every module has options. For instance, one could use ElevenLabs instant cloning first, then upgrade to a professionally trained voice later for even more realism, or start using GPT-3.5 for summarization to

save money and see if quality is sufficient. Because of the modular architecture, enhancements in one area, for example, a better summarizer model, can be added without completely changing the system.

Above all, the content will be produced in accordance with journalistic standards, which include factual reporting devoid of bias or advice and clearly attributed (implicitly, by adhering to what sources reported). Finally, the creator should include a brief disclaimer such as "Sources include Yahoo Finance, Investing.com, etc." if this podcast is being distributed publicly. This is not investment advice; rather, it is an automated news summary. This preserves the transparency that the content is algorithmically generated from news, even though the voice sounds like the user.

With this pipeline in place, the user can reliably produce a daily financial news update in a variety of formats (text, audio, and video) and concentrate their efforts on quality control and personal touches rather than the tedious tasks of writing or recording. It serves as an example of how automation and artificial intelligence can enhance media production by providing audiences with timely information with little delay.

Overall, the integration of these tools offers a **powerful end-to-end solution** for content creators in the finance domain, and the same blueprint could even be adapted to other domains (tech news, sports news, etc.) by adjusting sources and tone. This project, therefore, not only solves the immediate need but also provides a template for AI-driven multimedia content creation going forward.

Resources

A) News sources / ingestion

1. **Investing.com** | *RSS Feeds*
   https://www.investing.com/webmaster-tools/rss
2. **Reddit (r/investing)** | *“yahoo rss news feed is not accessible anymore?”*

   https://www.reddit.com/r/investing/comments/1gfrcg2/yahoo_rss_news_feed_is_not_accessible_anymore/

B) Summarization + factuality (AI summaries)

1. **Google Cloud** | *AI summarization (use case overview)*
   https://cloud.google.com/use-cases/ai-summarization
2. **Lindy** | *9 Best AI Summarizer Tools in 2025 (tested 20+)*
   https://www.lindy.ai/blog/best-ai-summarizer
3. **Reddit (r/PromptEngineering)** | *Best practices for generating neutral news summaries with AI?*

   https://www.reddit.com/r/PromptEngineering/comments/1jufe9i/question_best_practices_for_generating_neutral/

4. **arXiv** | *Improving Factual Consistency of News Summarization*
   https://arxiv.org/html/2310.19347v4

C) Podcast scripting / structure

1. **Content Allies** | *Podcast Scripting Made Easy: Examples, Templates & Tools*
   https://contentallies.com/learn/podcast-scripting-guide-examples-templates

D) Voice cloning + TTS (audio generation)

1. **ElevenLabs** | *AI Voice Cloning (product page)*
   https://elevenlabs.io/voice-cloning
2. **ElevenLabs Help Center** | *How many voice samples should I upload for Instant Voice Cloning?*
   https://help.elevenlabs.io/hc/en-us/articles/13434364550801-How-many-voice-samples-should-I-upload-for-Instant-Voice-Cloning
3. **Resemble AI Knowledge Base** | *How much data is needed?*
   https://knowledge.resemble.ai/how-much-data-is-needed
4. **Microsoft Learn (Azure AI Speech)** | *What is personal voice? (overview)*
   https://learn.microsoft.com/en-us/azure/ai-services/speech-service/personal-voice-overview
5. **Microsoft Tech Community** | *How to Create a Custom Neural Voice*
   https://techcommunity.microsoft.com/blog/azure-ai-foundry-blog/how-to-create-a-custom-neural-voice/3028275
6. **BentoML** | *The Best Open-Source Text-to-Speech Models in 2026*
   https://www.bentoml.com/blog/exploring-the-world-of-open-source-text-to-speech-models

E) Video generation, captions, and FFmpeg assembly

1. **Cloudinary** | *How to Use FFmpeg to Add Subtitles to Videos*
   https://cloudinary.com/guides/video-effects/ffmpeg-subtitles
2. **InMotion Hosting** | *How to Create Audio Spectrum Visuals with FFmpeg*
   https://www.inmotionhosting.com/support/edu/live-broadcasting/audio-spectrum-visuals-ffmpeg/
3. **Video StackExchange** | *How do I turn audio into video (waveforms)?*
   https://video.stackexchange.com/questions/9644/how-do-i-turn-audio-into-video-that-is-show-the-waveforms-in-a-video

F) Neutral reporting / tone benchmark (journalism standard)

1. **Reuters / Media Reform** | *Reuters Handbook of Journalism (PDF)*
   https://www.mediareform.org.uk/wp-content/uploads/2015/12/Reuters_Handbook_of_Journalism.pdf

G) Clarification, Research, Generation for visuals help :

1. **OpenAI** | *ChatGPT (used for research synthesis, system design, and report drafting)*
   https://chat.openai.com/

(Model used: ChatGPT — GPT-5.2 Thinking)