

Take-Home: Two-Host AI Newscast (Podcast)

Goal: Build a small app that turns today's top news into a short, personality-driven **two-host podcast**. Hosts are defined by input personas; the app fetches news, writes a back-and-forth script with ChatGPT, and renders audio with **Cartesia** TTS.

Example:

<https://www.youtube.com/watch?v=DReu3yKAEAk>

Inspiration: dynamic, fast-paced banter like the example video you shared.

1) Requirements (scope & acceptance)

Inputs

personas.json

describing two hosts (names, voices, personalities, styles).

Optional config: target duration (e.g., 6–8 min), topics or regions, profanity filter on/off.

News ingestion

Use **NewsAPI** to fetch 3–6 timely stories (default: last 24 hours).

Deduplicate and keep a short **facts blob** per story: title, url, source, 1–2 sentence summary.

Script generation (ChatGPT)

Create a **rundown** (cold open → stories → kicker) then a **dialogue** where hosts alternate turns.

Each line must be grounded in the fetched facts (no invented details) and annotated with

[src: i]

indices.

Keep the hosts distinct: different vocab, energy, and POV; light disagreement and callbacks are encouraged.

Tone: smart, accessible, no clickbait; add a one-sentence disclaimer if topics are sensitive.

Audio rendering (Cartesia)

Map each host to a Cartesia voice.

Convert each line to speech and **stitch** into a single **MP3** (or WAV) with ~1s natural pauses.

Optional: low-volume royalty-free intro/outro bed. (If used, document the license.)

Outputs (in

/out

)

episode.mp3

(final mixed audio).

transcript.jsonl

(one JSON object per line:

{t, speaker, text, src}

) and

episode.vtt

(timestamps).

show_notes.md

with bullet summaries and clickable links to sources.

Runtime UX

Single command:

```
python main.py --personas personas.json --minutes 7 --topics tech,world
```

(Node is fine too; just keep it one clear entry point.)

Graceful failure: if NewsAPI rate-limits or returns too little, explain in console and fall back to fewer stories.

Ethics & IP guardrails

Do not impersonate real voices; use “celebrity-style” as *personality*, not literal cloning.

Include a brief **parody/transformational use** disclaimer in

show_notes.md

.
No medical/financial advice; avoid slurs/harassment.

Acceptance checklist

Runs with

.env

keys for **NEWSAPI_KEY**, **OPENAI_API_KEY**, **CARTESIA_API_KEY**.

Produces MP3, transcript (JSONL + VTT), and show notes with source links.

Dialogue shows clear, consistent personalities and cites sources via

[src: i]

.
No obvious hallucinations; facts match links.

2) What we'll provide

API keys for **Cartesia**, **ChatGPT**, and **NewsAPI**.

3) Deliverables

Link to repo (or zip) containing:

Source code, **README** with setup/run steps, and

.env.example

(no real keys).

Sample output files in

/out

.
Brief **notes.md** (~300–500 words): design choices, prompt strategy, trade-offs, what you'd do next.

(Optional) Short screen capture (≤2 min) demoing run & result.

4) Evaluation (what we'll look for)

Content quality (35%) – Personality feels real; banter flows; facts are correct and attributed.

Tech execution (35%) – Clean structure, clear prompts, robust error handling, deterministic runs (seed).

Audio polish (15%) – Natural pacing, sensible pauses, levels not jarring.

DX & docs (15%) – One-command run, tidy README, reproducibility.

5) Getting started (templates)

Environment

```
#. env.example NEWSAPI_KEY=your_key
```

```
OPENAI_API_KEY=your_key
```

```
CARTESIA_API_KEY=your_key
```