WINDOWS, MACOS OR LINUX · VERSION 1.0 · 7 NOVEMBER 2026

BUILD YOUR OWN AUDIOBOOK GENERATOR

A goal-driven build prompt for your coding agent, plus the mistakes and tests that turn a voice demo into a tool that can read a whole book.



WHAT IT TEACHES

- Text-to-speech
- Text preparation
- Long-job engineering

01 One build prompt you paste once
02 The mistakes that break this build
03 Acceptance tests to prove it works
04 A repair prompt for when it doesn't

DR TOLU AJIDAHUN

BEFORE YOU START

START HERE

01

Create a new folder called My Audiobook Generator. Open a coding agent that can create files and run commands on your computer (for example Codex or Claude Code).

02

Copy both parts of the build prompt into the agent as one message and send it. That is the only prompt you need. The agent works through every stage and stops only when it needs you to listen to audio or click something.

03

When the agent asks, listen to the first one-page sample. Say exactly what sounds wrong: a word, a pause, the speed.

04

Run the acceptance tests before you trust it with a full book. Keep the mistakes page and the repair prompt for later.

WHAT YOU NEED

A computer with at least 8 GB of RAM and a few GB of free disk space, an internet connection for setup, a coding agent with access to that computer, and three test files: one short article (PDF or text), one long PDF, and one EPUB you own.

WHAT YOU ARE BUILDING

Open a local page. Upload a document. Choose a narrator. Press Generate. Watch progress chapter by chapter. Get one MP3 per chapter, playable on your phone. The voice runs on your own computer with Kokoro, a small open text-to-speech model.

KEEP THE SCOPE SMALL

One speech model. English first. One document at a time. Get one clean page of audio, then a chapter, then a full book. Keep a working version before each change.

THE BUILD PROMPT · PART 1

PASTE BOTH PARTS AS ONE MESSAGE

Copy part 1 and part 2 into your coding agent as one message.

BUILD PROMPT · PART 1 COPY INTO YOUR AGENT

GOAL

Build a personal audiobook generator on this computer, in this project folder. I upload an EPUB, PDF or text file, choose a voice, and get chapter-by-chapter MP3 files that sound natural and that I can play on my phone. All speech generation runs locally with the Kokoro text-to-speech model.

DONE MEANS

A 200-page document converts to audio without crashing, without skipped text, and without me watching it. The audio has no clicks between sentences, reads numbers and abbreviations sensibly, and resumes from the last finished chapter if the job stops.

HOW TO WORK

Work through every stage below without waiting for me. Verify each stage with a real test before you start the next. Stop only when you need something that only I can do: listen to audio, choose between options, or approve a change. Explain each handoff in plain English. Keep a TEST_LOG.md with every check, its result and the exact error if it failed.

STAGE 1: INSPECT AND PREPARE

Inspect my operating system, CPU, RAM, GPU (if any), free disk space and installed Python versions. Record them. Create an isolated Python environment for this project only. Leave other projects alone.

Read the current Kokoro-82M model card and the kokoro package README on Hugging Face and GitHub. Install the package and its required system dependency, espeak-ng, using the method the README gives for my operating system. Install a CPU-only build of PyTorch unless I have a supported NVIDIA GPU. Record the package versions and model version.

Prove the setup: generate a ten-second WAV of one test sentence with an American English voice. Confirm the file plays and has the expected sample rate.

STAGE 2: GET CLEAN TEXT OUT OF DOCUMENTS

Write a text extractor for TXT, PDF and EPUB. For EPUB, use the book's own chapter structure. For PDF, remove repeated headers, footers and page numbers, join words split by end-of-line hyphens, and keep paragraph breaks. Detect scanned PDFs that contain no extractable text and show a clear message instead of producing silence.

Save the cleaned text for each chapter as a .txt file I can open and read before any audio is generated. Show me the first two pages of cleaned text from my long PDF so I can spot problems.

STAGE 3: NORMALISE TEXT FOR SPEECH

Add a normalisation step before speech. Expand numbers, dates, currency, percentages and common abbreviations (Dr, e.g., etc.) so they read naturally. Remove or shorten URLs and footnote markers. Add a pronunciation file (a simple word = spoken form list) that I can edit, and apply it before generation.

THE BUILD PROMPT · PART 2

CONTINUE THE SAME MESSAGE

Continue the same message.

BUILD PROMPT · PART 2 COPY INTO YOUR AGENT

STAGE 4: GENERATE ONE PAGE, THEN ONE CHAPTER

Split text at sentence and paragraph boundaries into chunks the model handles well. Never cut a sentence in the middle. Generate audio chunk by chunk and join the chunks at the model's native sample rate. Add a short, consistent pause between paragraphs and a longer pause between chapters. Check the joins for clicks and silence gaps.

Generate one page with three different voices and stop. HANDOFF: ask me to listen and pick a voice and a speed. Then generate one full chapter with my choices and ask me to listen to the first two minutes.

STAGE 5: BUILD THE APP

Create a simple local web page: upload a file, see the detected chapters, choose voice and speed, press Generate. Run generation in a background worker so the page stays responsive. Show progress per chapter, elapsed time and estimated time left.

Save each finished chapter immediately as its own audio file. If the job stops, a restart skips finished chapters and continues from the next one. Add a Cancel button that stops cleanly.

Export MP3 at a sensible speech bitrate, with the book title and chapter number in the file name and the tags. Keep the WAV only until the MP3 is verified. Store finished books in a visible Audiobooks folder outside the project's build files, and add an Open folder button.

STAGE 6: PROVE IT ON A WHOLE BOOK

Convert my long PDF and my EPUB from start to finish. Measure total time, audio duration and real-time factor (seconds of processing per second of audio). Compare the word count of the cleaned text with the text actually sent to the model, so no section is skipped.

Run every acceptance test in this pack. Mark each PASS, FAIL or NOT RUN in TEST_LOG.md. Repair failures before you add any feature.

DELIVER

The command or launcher I use to start the app, the project folder, the Audiobooks folder, the pronunciation file, a requirements lock file, the Kokoro version, a short README and the test log. List which checks still need my ears. Keep all speech generation local; no cloud text-to-speech and no telemetry.

WHEN SOMETHING BREAKS

MISTAKES TO AVOID

Give this page to the agent when something goes wrong.

ESPEAK-NG IS MISSING OR NOT FOUND

Kokoro converts words to sounds with espeak-ng. If it is missing, words are skipped or mispronounced, or the import fails. Install it with the method the README gives for your operating system, then test in the same environment the app uses.

READING THE PDF AS IT LOOKS, NOT AS IT READS

Raw PDF text includes page numbers, running headers, broken hyphenated words and two-column jumbles. The narrator reads them all. Clean the text first and read it yourself before you generate audio.

SENDING THE WHOLE BOOK IN ONE CALL

Long input runs out of memory or gets cut off silently. Split at sentence boundaries, generate in chunks, and save each chapter as soon as it finishes.

CUTTING SENTENCES IN HALF

Splitting by character count breaks a sentence across two chunks. The voice drops its tone in the wrong place. Split at sentence ends only.

CLICKS AND GAPS BETWEEN CHUNKS

Joining audio at different sample rates, or with no pause control, gives clicks and uneven gaps. Keep one sample rate and add deliberate pauses.

"DR" READ AS "DRIVE", 2026 READ AS "TWO ZERO TWO SIX"

The model reads what it gets. Normalise numbers, dates and abbreviations before speech, and keep a pronunciation file for names and medical terms.

STARTING OVER AFTER A CRASH

A four-hour job that dies at chapter 18 should not restart from chapter 1. Save per chapter and resume from the first unfinished one.

THE PAGE FREEZES DURING GENERATION

Generation is slow work. Run it in a background worker and report progress.

SCANNED PDFS PRODUCE SILENCE

A scanned page is an image, not text. Detect it and say so. OCR is a separate project.

TESTING ONLY A SHORT PARAGRAPH

A ten-second clip hides every long-document problem. Test a complete book before you call it done.

PROVE IT WORKS

ACCEPTANCE TESTS

Use your own files. Keep the audio and the test log.

T01	CLEAN START	Start the app from the launcher. Expect the page to load and the model to be ready. Record the start time.	PASS · FAIL · NOT RUN
T02	FIRST WORDS	Generate one paragraph. Expect clear speech, the right voice and a playable file.	PASS · FAIL · NOT RUN
T03	CLEAN TEXT	Open the cleaned text of the long PDF. Expect no page numbers, headers or split words.	PASS · FAIL · NOT RUN
T04	HARD WORDS	Generate a passage with a date, a price, a percentage, ”Dr”, a URL and a name from the pronunciation file. Expect each to sound natural.	PASS · FAIL · NOT RUN
T05	JOINS	Listen to five minutes across many chunk joins. Expect no clicks, no repeated words and even pauses.	PASS · FAIL · NOT RUN
T06	WHOLE BOOK	Convert the long PDF. Expect one audio file per chapter and a word count that matches the cleaned text.	PASS · FAIL · NOT RUN
T07	EPUB CHAPTERS	Convert the EPUB. Expect chapter files that match the book’s table of contents.	PASS · FAIL · NOT RUN
T08	RESUME	Stop the job halfway through a book. Restart it. Expect finished chapters to be kept and work to continue from the next chapter.	PASS · FAIL · NOT RUN
T09	SCANNED PDF	Upload a scanned PDF. Expect a clear message, not silent audio.	PASS · FAIL · NOT RUN
T10	PHONE TEST	Copy one MP3 to your phone. Expect it to play with the correct title and chapter.	PASS · FAIL · NOT RUN

For every test record: date, app version, steps, expected result, actual result, PASS / FAIL / NOT RUN.

FIX IT · LEARN FROM IT

REPAIR PROMPT AND EXPERIMENTS

USE THIS REPAIR PROMPT

REPAIR PROMPT — COPY INTO YOUR AGENT

The failed test is [test ID].

I did: [exact steps].

Expected: [result].

Actual: [result, the exact error, or the timestamp where the audio is wrong].

Find the cause, make the smallest repair, and rerun this test plus any test the change can affect. Keep my finished audiobooks and pronunciation file. Update the test log and tell me what to listen to.

EXPERIMENTS THAT TEACH YOU HOW SPEECH AI WORKS

Change one thing at a time and listen to the difference.

01 Generate the same paragraph with three voices. The words are the same; the voice is a different set of numbers the model conditions on.

02 Turn normalisation off for one page. Hear how much of "natural" speech is text preparation, not the model.

03 Measure the real-time factor on a short clip and on a full chapter. See why a book is an engineering problem and a clip is not.

04 Change chunk size. Find the point where prosody (the rise and fall of the voice) gets worse.

KEEP THIS WITH THE PROJECT

SOURCES, LIMITS AND BUILD RECORD

SOURCES

Kokoro-82M model card
https://huggingface.co/hexgrad/Kokoro-82M

Kokoro package and install guide
https://github.com/hexgrad/kokoro

espeak-ng
https://github.com/espeak-ng/espeak-ng

LIMITS

This pack is an agent-assisted build brief; run the tests on your own computer. Kokoro's weights are Apache 2.0, but its espeak-ng dependency is GPLv3; check both before you distribute the app. Only convert books and documents you have the right to use; this tool is for personal listening. Long books take a long time on a CPU-only laptop. Coding-agent access has its own cost.

YOUR BUILD RECORD

Operating system _____

CPU / RAM / GPU / free disk _____

Kokoro version / voice chosen _____

Real-time factor on your machine _____

Project folder _____

Audiobooks folder _____

First book converted _____

BUILD SOMETHING. THEN UNDERSTAND HOW IT WORKS.

Change one thing at a time. Switch a model. Change an instruction. Measure how long something takes. Read the error. That is where the learning happens.

ALL FIVE STARTER PACKS · FREE · NO EMAIL NEEDED

01	AUDIOBOOK GENERATOR	Text-to-speech · Text preparation · Long-job engineering
02	OFFLINE DICTATION APP	Speech recognition · Local inference · Desktop packaging
03	PERSONAL KNOWLEDGE SEARCH	Embeddings · Semantic search · Retrieval (RAG)
04	AI RECEIPT TRACKER	Multimodal AI · Structured output · Validation in code
05	AI FILE ORGANISER	AI agents · Tool calling · Safe permissions

Get all five at drtolu.com/resources/guides-and-prompts