WINDOWS · VERSION 1.0 · 6 NOVEMBER 2026

BUILD YOUR OWN OFFLINE DICTATION APP

A practical build brief for your coding agent, with the failure modes and tests that turn a speech-model demo into a useful daily tool.



WHAT IT TEACHES

- Speech recognition
- Local inference
- Desktop packaging

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

BEFORE YOU START

START HERE

01

Create a new folder called My Dictation App. Open a coding agent that can create files and run commands on your Windows computer.

02

Paste both parts of the build prompt into the agent as one message. Ask the agent to complete one stage at a time and keep a test log.

03

Speak a short message when the first recording screen is ready. Read the actual result. Describe what failed.

04

Run the acceptance tests before relying on the tool. Keep the repair prompts for later.

WHAT YOU NEED

A Windows PC, a working microphone, enough space for the model and its software, an internet connection for setup, and a coding agent with access to that computer. Ask the agent to inspect RAM, CPU and available disk space first.

WHAT YOU ARE BUILDING

Press a shortcut, speak, stop, and get the transcript in the text field you were using. See whether the app is preparing, ready, listening or transcribing. Retrieve recent recordings and transcripts. Keep the model loaded in the background.

KEEP THE SCOPE SMALL

One English speech model. One reliable dictation workflow. Start with a record button and transcript, then add the shortcut, text insertion and installer. Keep a working version before making changes.

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

Build a personal English dictation app for my Windows PC in this project folder. Explain each handoff in plain English and do the implementation. Work in stages; verify each stage before adding features.

GOAL

A quick everyday tool: record speech, transcribe locally, insert the words where I was typing, and keep a short recoverable history.

STAGE 1: INSPECT AND PREPARE

Inspect the Windows version, CPU, physical RAM, free disk space, Python installations and microphone devices. Record the findings. Use a compatible 64-bit Python and CPU inference. Create an isolated environment; leave unrelated projects alone.

Use ConwayResearch/Underdog-Bark-3.0-Clean. Read its current model card, licence, files and official bark_loader.py. Download the model, configuration and processor files needed by the selected loader. Record the model revision, file hashes and working dependency versions.

Check the required audio-processing libraries, including librosa when the feature extractor requires it. Test imports in the same environment used by the launcher.

STAGE 2: MAKE TRANSCRIPTION WORK

Create a small window with Record, Stop, transcript text and a Copy button.

List available microphones; let me select one. Use the capture device's actual sample rate and the model loader's required conversion. Handle stereo correctly.

Load the model once. Show Preparing until it is ready. Keep recording disabled during preparation and transcription.

Record audio, stop cleanly, and save a valid WAV before starting inference. Keep that file if transcription fails. Finalise the WAV header so another player can open it.

Display the real transcript, the audio duration and elapsed transcription time. Handle empty recordings, silence and errors without inventing output.

Keep the loaded model available for the next take. Use a worker for slow loading and inference so the window stays responsive.

Let me set a recording-length limit. Warn before it is reached and save the recording at the limit.

FIRST HANDOFF

Show the actual executable or launcher I should open. Help me make one short recording and compare the words. Record the observed result before continuing.

THE BUILD PROMPT · PART 2

CONTINUE THE SAME MESSAGE

Continue the same message. Keep the interface simple.

BUILD PROMPT · PART 2 COPY INTO YOUR AGENT

STAGE 3: ADD GLOBAL DICTATION

Use Right Ctrl + Right Shift as the initial start/stop chord, with a setting to change it if it conflicts. Require a full key release between activations. A held chord must not trigger repeatedly.

Show a small, visible Listening / Transcribing / Ready indicator without taking focus away from the app where I am typing.

Remember the target window when recording starts. Insert the transcript only if that target is still the intended active window. Otherwise copy it and show clear feedback. Keep the transcript in history either way.

Verify Unicode input, punctuation and line breaks in Notepad and the application I actually use. Report what happens in a field where insertion is blocked.

Clicking X should hide the main window to the system tray. The tray menu must offer Open and Quit. Quit releases the microphone and shortcut. Starting twice should leave one running instance.

STAGE 4: MAKE RECORDINGS RECOVERABLE

Keep the latest ten recordings and ten transcripts by default. Let me change retention, including keeping a recording until I delete it.

Pair each recording with its transcript using a stable ID and timestamp. Store them in a visible local folder, outside temporary build directories. Show the resolved path and provide an Open folder button.

Handle Windows folder redirection and names with spaces. Verify that the actual folder opens in Explorer.

Save transcript history atomically. Test reloading it after restart. Preserve user recordings and transcripts during repairs and updates.

Delete older items only under the configured retention rule. If a file is locked, keep the app running and report the failed deletion.

STAGE 5: PACKAGE AND VERIFY

Build a normal Windows application with an obvious Start-menu launcher. Include every runtime file needed by the packaged app. Keep the EXE with its support folder if the package uses one.

Complete all model downloads during setup. Restart with networking disabled and create a new transcript using the packaged app.

Run the acceptance tests in this pack. Mark each PASS, FAIL or NOT RUN. Keep exact errors and repair the failed stage before adding features.

DELIVER

The installed launch path, source folder, data folder, requirements lock file, model revision, attribution, simple README and test log.

State the actual checks completed and the checks that still need me to speak or click. Keep all audio processing local; omit cloud transcription and telemetry.

PROVE IT WORKS

ACCEPTANCE TESTS

Use your own short phrases. Keep the actual audio and output.

T01	CLEAN LAUNCH	Open from the normal user-facing launcher. Expect Preparing, then Ready. Record the cold-start time and memory use; do not count a warm model as a cold start.	PASS · FAIL · NOT RUN
T02	REAL MICROPHONE	Record about 15 seconds and stop. Expect a playable WAV, a visible transcript, an audio duration and a transcription time. Verify the saved audio contains the words you spoke.	PASS · FAIL · NOT RUN
T03	YOUR VOCABULARY	Speak everyday messages, names, numbers and a self-correction. Compare the whole sentence. Log omitted or altered meaning as well as spelling mistakes.	PASS · FAIL · NOT RUN
T04	MISSING MICROPHONE	Disconnect or disable the chosen input. Attempt recording. Expect a clear device message and a way to choose a working input. The window must remain usable.	PASS · FAIL · NOT RUN
T05	EMPTY AND SILENT TAKES	Start and stop immediately, then try silence. Expect an empty/silent result or a clear message. The app must remain ready for another take.	PASS · FAIL · NOT RUN
T06	RECOVERY AFTER INFERENCE FAILURE	Use a test-induced model error after recording. Expect the WAV to remain playable. Restore the model, then retry that file without recording it again.	PASS · FAIL · NOT RUN
T07	DURABLE HISTORY	Create three identifiable takes. Quit and reopen. Expect the same recordings and transcripts, correctly paired and readable. Open the data folder from the app.	PASS · FAIL · NOT RUN
T08	RETENTION	In a separate test profile, create twelve takes with a ten-item limit. Expect the newest ten and no deletion outside the test data folder. Pin or keep one item if the feature supports it.	PASS · FAIL · NOT RUN
T09	REPEATED SHORTCUT	Toggle recording twenty times with full release between presses. Test holding the chord. Expect one state change per activation, with no double-trigger.	PASS · FAIL · NOT RUN
T10	INSERT IN THE RIGHT PLACE	Put the cursor in Notepad, record and stop. Expect the complete transcript at the cursor. Repeat with punctuation, a line break and a Unicode character.	PASS · FAIL · NOT RUN

T11	TARGET CHANGES	Start recording in one app, switch to another, then stop. Expect a clipboard fallback and clear feedback rather than surprise text in the new app.	PASS · FAIL · NOT RUN
T12	TRAY AND QUIT	Click X, dictate again, then restore the window. Expect the shortcut to work while hidden. Quit from the tray; expect the process, microphone and shortcut to stop.	PASS · FAIL · NOT RUN
T13	DUPLICATE LAUNCH	Launch the installed app twice. Expect one running instance and one active microphone owner. Keep the existing history.	PASS · FAIL · NOT RUN
T14	OFFLINE RESTART	Finish setup, disable networking, quit, relaunch and make a new recording. Expect local model loading and a new transcript. Restore networking after the check.	PASS · FAIL · NOT RUN
T15	REAL PACKAGE	Launch through the actual Start-menu entry from the normal Windows shell. Confirm all support files travel with the EXE and the data folder opens in Explorer.	PASS · FAIL · NOT RUN
T16	UPDATE WITHOUT DATA LOSS	Back up the test profile. Apply a small app update and reopen. Expect the previous recordings, transcripts, model cache and settings to remain.	PASS · FAIL · NOT RUN

For every test record: date, build version, model revision, steps, expected result, actual result, PASS / FAIL / NOT RUN, and evidence path. Record critical failures separately from cosmetic issues.

WHEN SOMETHING BREAKS

MISTAKES TO AVOID

Give this page to the agent when a stage fails.

STUCK ON PREPARING

Check the startup log, missing imports and whether the model download completed. Show a useful error with a retry action.

WORKS IN TERMINAL, FAILS IN APP

Check that the launcher uses the same interpreter, dependencies, configuration and working directory as the test.

MISSING PYTHON DLL

Check the packaged support folder and native libraries. Test the complete bundle from the path the user will launch.

SHORTCUT WORKS ONLY SOMETIMES

Check right-side key detection, full-release handling, focus and duplicate instances. Repeat real key presses.

X MAKES DICTATION DISAPPEAR

Check whether closing hides the window or quits the process. Implement explicit tray Open and Quit actions.

WORDS LAND IN ANOTHER APP

Check the remembered target and current focus. Fall back to the clipboard and preserve the transcript.

RECORDING LOST AFTER MODEL ERROR

Save and finalise the audio before inference. Keep failed takes available for retry.

HISTORY POINTS TO MISSING FILES

Use stable recording IDs, visible persistent paths and a tested retention rule. Update references after moves.

EXPLORER CANNOT OPEN THE FOLDER

Resolve Windows' actual folder location. Test outside the agent's shell and support redirected Documents folders.

OFFLINE MODE DOWNLOADS AGAIN

Cache every required model, processor and configuration file. Test a full restart with networking disabled.

SECOND TAKE IS SLOW AGAIN

Check that the model stays loaded. Measure startup and transcription separately.

UPDATE DELETES RECORDINGS

Separate source, installed app, model cache and user data. Back up before migration; test the update with a test profile.

FIX IT · LEARN FROM IT

REPAIR PROMPTS AND HANDOFF

USE THIS REPAIR PROMPT

REPAIR PROMPT — COPY INTO YOUR AGENT

The failed test is [test ID].

I did: [exact steps].

Expected: [result].

Actual: [result and exact error].

Inspect the relevant log and the installed build. Identify the cause, make the smallest repair, and rerun this test plus the behaviour the change can affect.

Preserve recordings, transcripts, settings and model cache. Keep a rollback copy of the last working app.

Show the real result and the launch path I should test. Update the test log.

USE THIS PROMPT FOR SLOW PERFORMANCE

PROMPT FOR SLOW PERFORMANCE — COPY INTO YOUR AGENT

Measure model loading and transcription separately on this PC. Use three saved recordings of different lengths. Record CPU, RAM, audio duration and elapsed time. Check that the model stays loaded between takes. Identify the biggest delay and fix one cause at a time. Keep output quality visible and compare the same recordings after each change.

FINAL HANDOFF CHECKLIST

The Start-menu entry opens the installed app. The microphone works. The shortcut and recording indicator work across apps. History survives restart. The recordings folder opens. Offline restart works. Quit exits fully. Test results identify every remaining failed or unrun check.

Keep the project source, model revision, dependency versions, installer, README and test log together. Keep user data in its own visible folder. Make a backup before the next repair.

KEEP THIS WITH THE PROJECT

SOURCES, LIMITS AND BUILD RECORD

SOURCES

Model and CPU-loader instructions
https://huggingface.co/ConwayResearch/Underdog-Bark-3.0-Clean

Official loader file
https://huggingface.co/ConwayResearch/Underdog-Bark-3.0-Clean/blob/main/bark_loader.py

ToluWhispr build account
https://drtolu.com/blog/how-i-made-toluwhispr

LIMITS

This pack is an agent-assisted build brief; run its tests on your own PC. Bark Clean targets English and removes speech disfluencies. Coding-agent access, setup and hardware have their own costs. The model card currently lists CC BY-NC 4.0; commercial use requires checking the licence with its publisher.

YOUR BUILD RECORD

Windows version _____

CPU / RAM / free disk _____

Microphone _____

Model revision / hash _____

App version / dependency lock _____

Source folder _____

Installed launcher _____

Recordings / transcript folder _____

Test log / backup location _____

First useful task _____

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