Voice Lock

Testing speaker verification with synthetic speech

Dr. Ahmet Furkan Aydogan | aydogana@uncw.edu
Department of Computer Science, University of North Carolina Wilmington

Track	Cyber Operations, Voice Authentication
Term	Fall 2026
Deliverable	Completed student worksheet, five labelled screenshots, Q1–Q6
Points	100

Abstract. *A familiar voice can be generated from an example recording. In this lab, you will compare a local voice clone with a cloud voice clone, test a spoken-consent check, and measure how a teaching portal responds to synthetic login speech. Your evidence must distinguish voice similarity, recognized words, and the final access decision.*

PART I. Background and demonstration

The case

You are evaluating the instructor's Voice Lock training portal. The supplied **instructor-voice.wav** is your reference recording; it is provided in this folder, not hidden on the website. Use it to condition a local model and investigate cloud voice replication. You will generate new speech for the portal rather than merely identify whether a recording sounds artificial.

What you will learn

(a) Explain text-to-speech, voice cloning, speech recognition, and speaker verification as different tasks.

(b) Build a reproducible local and cloud workflow while recording model, input, and output details.

(c) Test whether a spoken-consent check distinguishes an authorized human recording from synthetic speech.

(d) Compare voice and phrase scores without treating either score as a probability of identity.

Scope of this exercise

The instructor authorizes use of the supplied instructor recording for this class experiment, including the synthetic-consent test and the designated Voice Lock portal. Use only this recording and your assigned account. Do not substitute another person's voice or test another login system. Keep generated clips and voice profiles within the course; label them as synthetic in filenames and submissions.

Part I explains the background. Start your graded records in Part II. A documented rejection is a valid experimental result; success is not guaranteed.

From text to a familiar voice

Text-to-speech (TTS) turns written words into audio. A stock voice reads in a predefined voice. A cloned voice adds a speaker reference so newly generated words resemble that person. In this lab, "local training" means **zero-shot voice conditioning**: you provide a reference to an already trained model, without training a new neural network from scratch.

Speech recognition estimates what was said. **Speaker verification** compares vocal characteristics against an enrolled reference. A speaker verifier can produce a high score even when the words differ; a recognizer can transcribe the correct words from the wrong voice. Voice Lock requires both checks to pass.

The current Gemini TTS family

Google's September 2026 release introduces Gemini 3.8 Flash TTS and Flash-Lite TTS. This exercise uses **gemini-3.8-flash-tts**. Google describes controls for expressive delivery as well as stock, designed, and replicated voices. A style instruction changes how speech is delivered; a speaker reference changes whose vocal characteristics are being approximated. Record the exact model identifier you actually use; names and account availability can change. [1, 7]

What the 2025 report found

Consumer Reports published *AI Voice Cloning: Do These 6 Companies Do Enough to Prevent Misuse?* in March 2025. It is a product assessment, not a peer-reviewed paper. Four tested services relied on permission assertions without a technical consent check. The report discusses Descript's recorded-consent requirement and cites a 2024 Proof News demonstration of using another service to generate the consent speech. [3, 4]

The instructor's authorized pilot

In the instructor's own-voice pilot, a consent clip spoken by an unrelated synthetic voice was rejected. A consent clip generated by Chatterbox from the instructor's reference was accepted and a Gemini voice profile was created. This is a session-specific observation supplied by the instructor, not a claim that every account or future version will behave the same way.

Your experiment asks whether this observation reproduces under your recorded conditions. Keep these separate: the instructor's actual permission, a service's automated consent decision, and the portal's access decision. Acceptance by software is not evidence that the speaker personally recorded the clip.

Watermarks and access decisions

Synthetic audio may carry a provenance watermark, such as SynthID or Perth. A speaker similarity check does not, by itself, test for these watermarks. Do not infer "human" from an accepted login, or "no watermark" from an unchanged similarity score. [1, 6]

PART II. Student investigation

Step 1 Prepare your workspace

Extract the student ZIP into a short local path, such as **C:\VoiceLockLab**. Keep the supplied WAV unchanged. Create **working**, **generated**, and **evidence** folders. Save new recordings in generated and screenshots in evidence.

Record your operating system, processor, RAM, Python version, and whether you will use CPU or GPU in **Log 1**. Hash the original recording, then inspect its duration, channels, sample rate, and encoding in Audacity or an equivalent audio tool. Record these in Log 1 before conversion.

Command reference

```
Get-FileHash .\instructor-voice.wav -Algorithm SHA256
py -0p
```

Install the basic tools

On Windows, install 64-bit Python 3.12 from the official Python downloads and include the launcher and pip. Open a new PowerShell window and run **py -3.12 –version**. Install Audacity for listening and WAV export. Run the remaining commands from your extracted lab folder. Use the version assigned by the instructor if your managed computer restricts installation.

Local computing requirements

Use the original English **ChatterboxTTS** implementation, package **chatterbox-tts 0.1.7**, to match the instructor's experiment. The instructor ran this on Windows with Python 3.12 and CPU inference. Upstream documents a Python 3.11 environment; your OS and dependency combination may behave differently. [6]

For planning, use a modern 64-bit computer, preferably 16 GB RAM and at least 10 GB free disk space. These are course recommendations, not published vendor minimums. CPU generation can be slow. A compatible NVIDIA GPU can accelerate inference; it is optional. Apple Silicon support depends on the framework and model path and was not tested in the instructor's Windows pilot. Initial model downloads require internet access.

Accounts and services

You need access to Google AI Studio and the Gemini API, plus your own assigned student ID for the portal. Check model availability, quotas, and any billing requirement before submitting requests. If access or cost prevents a step, tell the instructor and document the limitation. Do not invent results or buy a plan just to fill a worksheet field.

Students do not need to install Cloudflare, Docker, or Podman. The instructor runs the verifier on the server. Your local model generates recordings; the website evaluates them.

Evidence: Screenshot 1 shows the source hash and Python version. Exclude unrelated paths or personal account information. Complete Log 1.

Step 2 Create a cloud TTS baseline

Open Google AI Studio, sign in, and use its API key page to create a key for a project you control. Keep the key private. Never paste it into the worksheet, screenshots, a shared script, or a submitted archive. Follow the current key guide if your account shows a different flow. [5]

Create a separate Python environment for Google. The following commands use Windows PowerShell and an installed Python 3.12. On macOS/Linux, use your Python interpreter and the environment's bin/python path instead of Scripts/python.exe.

Command reference

```
py -3.12 -m venv .venv-google
.\.venv-google\Scripts\python.exe -m pip install "google-genai>=2.25.0"
```

Set GEMINI_API_KEY for the current shell without placing the literal key in command history. Paste it at the hidden prompt. Close the shell after the lab.

Command reference

```
$labKey = Read-Host "Gemini API key" -AsSecureString
$env:GEMINI_API_KEY = [System.Net.NetworkCredential]::new(
    "", $labKey).Password
Remove-Variable labKey
```

First create a short **stock-voice** sample using the current speech-generation quickstart. Use a neutral sentence of your choice, not the login phrase. A stock voice confirms that your API setup works; it is not your instructor clone. Save the audio and record the exact model, chosen voice, input text, and outcome in **Log 2**. [1]

Using a replicated voice later

Once Step 4 returns your own voice ID, the following pattern generates a new WAV. Replace the two placeholder values and save it as cloud_speech.py; run it with the Google environment. The ID identifies your profile and is not your API key. [2]

Command reference

```
import base64
from pathlib import Path
from google import genai
client = genai.Client()
voice_id = "YOUR_VOICE_ID"
text = "YOUR_EXPERIMENT_TEXT"
r = client.interactions.create(
    model="gemini-3.8-flash-tts",
    input=[{"type": "user_input", "content": [
        {"type": "text", "text": text}]}],
    response_format={"type": "audio"},
    generation_config={"speech_config": [{"voice": voice_id}]})
Path("generated/cloud-output.wav").write_bytes(
    base64.b64decode(r.output_audio.data))
```

Step 3 Generate a local voice clone

Keep the local dependencies separate from the Google environment. Install the pinned package in a fresh environment. Save the final installed versions so another student can identify the environment that actually worked.

Command reference

```
py -3.12 -m venv .venv-local
.\.venv-local\Scripts\python.exe -m pip install --upgrade pip
.\.venv-local\Scripts\python.exe -m pip install chatterbox-tts==0.1.7
.\.venv-local\Scripts\python.exe -m pip freeze > evidence\local-packages.txt
```

Save the example below as **local_speech.py**. Put a short, neutral test sentence in **working/text.txt**, then run the script using the local environment. Keep a copy of each input text alongside its output; rename files before the next run. The first load downloads model files and may take substantially longer than later runs.

Command reference

```
from pathlib import Path
import soundfile as sf
from chatterbox.tts import ChatterboxTTS

model = ChatterboxTTS.from_pretrained(device="cpu")
text = Path("working/text.txt").read_text(encoding="utf-8")
audio = model.generate(
    text, audio_prompt_path="instructor-voice.wav")
sf.write("generated/local-output.wav",
         audio.squeeze(0).cpu().numpy(), model.sr)
```

Command reference

```
.\.venv-local\Scripts\python.exe local_speech.py
```

Listen to the result. Note intelligibility, speaker resemblance, duration, and any repetitions or missing words. Record the exact text, reference file, output filename, model/package version, device, and elapsed generation time in **Log 3**. Listening is an observation, not a substitute for the portal measurements.

Preparing working copies of audio

For cloud replication, prepare clean 10–30 second reference audio; 24 kHz mono 16-bit WAV is the documented recommendation. Use Audacity to export a working copy, or use FFmpeg if it is installed. Keep the original and record the conversion. [2]

Command reference

```
ffmpeg -i instructor-voice.wav -ar 24000 -ac 1 -c:a pcm_s16le working/reference-24k.wav
```

If installation fails, save the command and final error lines. Check that you used the intended environment and interpreter. Do not mix random framework versions until the error is understood. If CPU generation appears slow, allow one run to finish before starting another.

Evidence: Screenshot 2 shows a completed local run and the output file or waveform. Complete Log 3.

Step 4 Investigate the spoken consent check

The documented replication flow asks for a source recording and a consent recording from the same adult speaker. The English statement is: [2]

"I am the owner of this voice and I consent to Google using this voice to create a synthetic voice model."

This experiment deliberately tests a synthetic consent clip under the instructor's authorization. It does not follow the documented requirement for a real human consent recording. Your report must identify the clip as synthetic; do not describe it as a fresh human recording.

Before testing: In **Log 4**, predict whether the service will accept the clip and explain your reason. Identify the reference and the model that will generate the consent speech. Keep this prediction even if the result differs.

Use your local workflow to generate the statement from the supplied instructor reference. Check the words by listening. Save this clip separately from the login clips; record its file hash and any export conversion.

Open the Voice Replication experience linked from the official guide. Supply the working reference as source audio and your locally generated statement as consent audio. Use your own project and record the date, model, and response. If you use the API instead, adapt the guide's **voices.create** example and retain your request settings without credentials. [2]

Record acceptance or rejection in Log 4. If accepted, record a shortened profile identifier and generate a brief test clip to verify the profile is usable. If rejected, preserve the actual error. Distinguish a consent rejection from an unavailable model, invalid WAV, quota limit, or network error. Do not relabel a service error as proof that synthetic consent was detected.

Bounded follow-up

Make at most one revised consent attempt if the first fails. Choose one change, justify it, and retain both outcomes. If a profile cannot be created, continue the local and original-audio portal tests. Report the Gemini portion as unavailable with evidence and request an instructor-provided comparison result; do not substitute a stock voice and label it an instructor clone.

Evidence: Screenshot 3 shows the actual consent outcome or blocking error. Redact the API key, account details, and full profile ID. A well-documented rejection receives experimental credit.

A question to carry forward

A spoken statement can have the correct words and resemble the expected speaker. Which additional fact would a service need to establish before it could conclude that the person knowingly made that statement? Answer from your evidence, not from whether your request happened to succeed.

Step 5 Test the Voice Lock portal

Open aydogana.com/voicelock and enter your assigned student ID. After that gate, read the requested speech on the page. The intended phrase for this lab is:

"Hello, this is the admin. I would like to log in and access my account."

The configured system compares the speaker embedding with the enrolled instructor reference and transcribes the same uploaded audio. Voice similarity must reach **0.60**; normalized text similarity must reach **0.85**. Capitalization and punctuation are ignored, and "login" is normalized to "log in." A small explicit negation check can also reject a phrase. This is approximate word matching, not full semantic understanding.

The displayed voice percentage is a scaled similarity score, not the probability that the instructor spoke. Keep voice score, text score, transcript, and access decision in separate columns. If the live page lacks these fields or shows a different phrase, notify the instructor before comparing results.

Design and run the comparison

Complete the prediction and variable fields in **Log 5** before uploading. Keep the same source reference and requested phrase for the first local/cloud comparison. Generate the login phrase independently using each clone. Submit WAV files, within the page's upload limits; the intended limits are 0.5–60 seconds and 10 MB.

Run	Input	Purpose
A	Supplied original instructor recording	Baseline for an authentic reference recording.
B	Local clone reading the requested phrase	Measure the local generation path.
C	Gemini clone reading the same phrase	Compare the cloud generation path.
D	One controlled variation of B or C	Change one factor; state a prediction first.

Choose the variation yourself: for example, a delivery setting or an audio preparation choice. Keep other factors fixed and record them. Do not change server thresholds. Save all attempts, including failures; do not report only your best recording. A repeated run is not a new independent speaker.

For each upload, copy the actual transcript and scores into Log 5. Note model/filename, settings, and the decision. Explain discrepancies between what you hear and what the recognizer writes. Use "unavailable" and a reason for missing measurements; never enter a guessed zero.

Evidence: Screenshots 4 and 5 show local and Gemini portal outcomes, including scores and transcript, or the documented reason the Gemini test could not run. Hide your student ID.

Step 6 Interpret and preserve the evidence

Complete **Log 6** with the source and generated filenames, hashes, and the role of each file. Keep the originals of your measurements. Then answer Q1–Q6 on the worksheet in your own words, referring to specific runs where appropriate.

Q1. Explain TTS, voice cloning, recognition, and verification. Which system produces each observed result?

Q2. Compare your consent prediction with the actual result. What does that result establish, and what remains untested?

Q3. Compare local and Gemini voice scores and text scores. Explain why this small experiment cannot rank the models generally.

Q4. Use Run D to explain the effect of your one changed factor. Identify a confound or alternative explanation.

Q5. Explain the difference between accepting the right words in the wrong voice and the right voice saying the wrong words. What tradeoff appears if either threshold is lowered?

Q6. Propose an additional defense, describe a test of that defense, and explain one usability cost or remaining weakness.

Step 7 Submit your work

Submit **Lastname_Firstname_VoiceLock.zip** containing the completed worksheet, Screenshots 1–5, your input text files, generated experiment WAVs, relevant redacted error records, and local-packages.txt. Preserve clear run identifiers. Do not include API keys, virtual environments, model caches, cookies, or the full instructor source again. Leave unperformed runs visibly marked with a reason.

After preserving the evidence, remove the test voice profile from your project using the current voice-management controls or API guide. Record cleanup status in Log 6. Keep only the course evidence for submission; do not publish the instructor clone. [2]

Assessment	Points
Environment, source identity, and cloud baseline	15
Reproducible local generation	15
Consent experiment and accurate outcome evidence	20
Controlled portal comparison and complete records	25
Q1–Q6 reasoning and limitations	20
Submission organization and cleanup record	5

Scores assess method, evidence, and reasoning. An access denial or documented service limitation is not automatically an incorrect result. Report what occurred.

Sources and technical references

Documentation checked 25 September 2026. Recheck model access and SDK examples before running the lab. The Google workflow, local generation workflow, and teaching portal are separate systems.

[1] Google. Gemini speech generation guide.
https://ai.google.dev/gemini-api/docs/speech-generation

[2] Google. Voice replication and profile management.
https://ai.google.dev/gemini-api/docs/voice-replication

[3] Consumer Reports. AI Voice Cloning: Do These 6 Companies Do Enough to Prevent Misuse? March 2025.
https://innovation.consumerreports.org/AI-Voice-Cloning-Report-.pdf

[4] Janus Rose, Proof News. AI Tools Make It Easy to Clone Someone's Voice Without Consent. June 25, 2024.
https://www.proofnews.org/ai-tools-make-it-easy-to-clone-someones-voice-without-consent/

[5] Google. Using Gemini API keys.
https://ai.google.dev/gemini-api/docs/api-key

[6] Resemble AI. Chatterbox repository and installation examples.
https://github.com/resemble-ai/chatterbox

[7] Google. Gemini 3.8 text to speech announcement. September 2026.
https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-8-text-to-speech/

How to interpret the evidence

The historical report and news demonstration provide background. The instructor's pilot provides a local observation. Your worksheet provides evidence of your own experiment. Do not merge these into a claim that all consent systems fail or all clones defeat authentication.

Portal thresholds and normalization in this handout describe the course implementation. They are not Google or Chatterbox defaults. The software's approximate text score measures word edits after normalization; it does not prove the sentence has the intended meaning.