Iteration 3 Report
EECE 2140: Computing Fundamentals
Speech-to-Text Notes App

Justin Glabicki, Cassidy Sakamoto, and Nathan Tan
Department of Electrical and Computer Engineering
Northeastern University
glabicki.j@northeastern.edu, sakamoto.ca@northeastern.edu,
and tan.nat@northeastern.edu

November 4, 2025

GitHub Link: `https://github.com/cassidysakamoto/FinalProject.git`

Contents

1 Summary of Team Progress and Development Updates

During Iteration 3, our team advanced the Speech-to-Text Notes App from a prototype into an integrated tool with a working audio pipeline, transcription logic, and a full UI for creating, searching, tagging, and exporting notes. Our key goals were: (i) reliable real-time capture and chunking of audio, (ii) practical transcription with error handling and domain-word normalization, and (iii) a user interface that supports start/stop recording, visual mic feedback, search/filter, autosave editing, and export.

Major achievements this iteration include: 20 ms frame capture aggregated into 2 s chunks, per-chunk retry logic, and light punctuation/domain-word replacements (for terms like PWM, Hz, and kHz). On the UI side, we delivered an app with Start/Stop, a live mic-level bar (quiet/good/loud), timestamp markers, JSON-backed note storage, keyword search, tag filtering, and export to .txt and .md files. Testing focused on file I/O, JSON persistence, export correctness, and stability under continuous recording. Pending tasks include hotkeys, settings surfacing (language/model), and topic tagging.

2 Implemented Core Features

Offline/Online Transcription Pipeline

Goal: Robust speech-to-text chunking with resilience to brief API/network issues. **Implementation:** Audio sampled at 16 kHz; frames of 20 ms are aggregated into 2 s chunks. Each chunk is recognized with a retry loop (up to three attempts) and light normalization (sentence-final punctuation and domain replacements for technical terms). The pipeline reports warnings for low input energy. **Validation:** Exercised on continuous speech; verified partial transcripts emitted per chunk with resilient behavior under transient network issues.

Microphone Input Noise Gate

Goal: Stable capture of live audio, early rejection of low-level noise, and mic-level metering hooks. **Implementation:** A microphone input class based on `sounddevice` captures audio, computes RMS, applies a simple noise gate, and exposes a callback to feed buffered chunks to the transcriber. Logging includes stream status and per-chunk volume; a simple console bar visualizes levels for tuning. **Validation:** Manual tests verified thresholding and chunk forwarding; buffer limits (e.g., 5–10 chunks) exercised to ensure older frames are dropped safely.

Desktop UI: Recording, Editing, Search/Filter, Export

Goal: Provide a complete notes interface with recording controls and note management. **Implementation:** Tkinter app with Start/Stop buttons, elapsed timer, timestamp marker insertion, and a live mic meter ("Too quiet/Good/Too loud"). Notes are persisted as JSON (id/title/content/tags/timestamps/duration); there is keyword and tag filtering, and export to .txt or .md files. Autosave is triggered via user action (Save) and the UI includes a recorder adapter interface; a dummy recorder enables UI dev without the live audio engine. **Validation:** Verified JSON read/write integrity, list referesh ordering by `updatedAt`, and correct export filenames using slugified titles; confirmed meter/labels update at 10-12 Hz and timestamp markers insert at the correct elapsed time.

3 Challenges and Resolutions

- **Chunk Latency and Partial Output:** Early versiosn produced slow partials when chunks were too large. **Resolution:** Standardized on 20 ms frames with 2 s aggregation, balancing API calls and perceived latency; added per-chunk timestamps for easier debugging.
- **Low Mic Level / Background Noise:** Quiet environments triggered "empty" transcripts. **Resolution:** Added an energy thresholdgate and user feedback (quiet warning), plus a UI meter label to guide input gain.
- **Technical Terms Misrecognized:** Domain words ("p w m") hurt readability. **Resolution:** Post-recognition normalization table that joins common terms (PWM, Hz, kHz, RC, SoC) and enforeces light punctuation.

4 Leadership Rotation and Team Contributions

Leadership Summary

Week/Span	Leader	Responsibilities	Key Outcomes
Week 1	Justin Glabicki	UI Integration (Start/Stop, meter, notes, export)	Delivered working U
Week 2	Cassidy Sakamoto	Mic input, RMS noise gate, buffering	Stable stream, thres
Week 3	Nathan Tan	Chunking retries, normalization, timing	20 ms/2 s framing,

Individual Contributions

Team Member	Contributions (Technical / Documentation)	Hours
Justin Glabicki	Tkinter UI: Start/Stop, elapsed timer, timestamp markers; live mic meter label; JSON notes (title/content/tags/timestamps), keywork/tag filtering, export to .txt/.md files; recorder adapter and dummy recorder for UI dev; documentation of UI flows.	11 hrs
Cassidy Sakamoto	`sounddevice` mic stream; RMS-based noise gate; callback wiring; circuilar buffer (max-chunk) aggregation; console meter for tuning; start/stop lifecycle; notes on threshold selection and stream status handling.	10 hrs
Nathan Tan	16 kHz capture with 20 ms frames; 2 s chunk aggregation; per-chunk retry logic; low-energy warnings; partial transcription emission; domain-word replacements and light punctuation; timing prints for profiling.	10 hrs

Statement by the Individual Submitter

I, **Justin Glabicki**, confirm that the above table accurately reflects my personal contributions during Iteration 3.
I, **Cassidy Sakamoto**, confirm that the above table accurately reflects my personal contributions during Iteration 3.
I, **Nathan Tan**, confirm that the above table accurately reflects my personal contributions during Iteration 3.