**AudioApp Architecture Document**

**Audio Recording & Transcription App**

**Overview**

AudioApp is for transcribing audio into text. It is designed to do the following:

* Record audio from the microphone in **user-defined segments** (default 30 sec),
* Efficiently manage memory and battery during long recordings,
* Transcribe each segment via **OpenAI Whisper API** with fallback to **Apple Speech Recognizer**,
* Store the recordings & transcriptions locally using **SwiftData**,
* Provide full **VoiceOver accessibility support**.

It is built using a **Swift 6,** **SwiftUI + SwiftData architecture**, leveraging:

* AVAudioEngine for audio streaming,
* SwiftData for persistence of sessions and segments,
* @Observable for model observation,
* NavigationStack + List for scalable UI with large datasets.

**Architectural Decisions**

**Core Recording Logic —**

**AudioManager**

* **Responsibility**: Handles audio session configuration, recording pipeline, segmentation, writing to disk, and transcription.
* Uses AVAudioEngine for audio input with an installed tap on the input node.
* Buffers short audio chunks, AVAudioPCMBuffer, into segments until the total time exceeds the specified segmentDuration, then writes to disk.

**Why**:

* This allows long recordings without growing memory usage indefinitely. Each segment is written to disk and the buffer is cleared.

**Memory & Battery Optimization**

* Keeps only the current segment in memory.
* Pauses & stops AVAudioEngine when needed, releasing hardware resources.
* Uses AVAudioSession category .playAndRecord + .notifyOthersOnDeactivation for smooth interruption handling.

**Why**:

* Ensures low memory and CPU usage even for long recordings, aligns with battery life requirements in the iOS Audio Recording & Transcription Take-Home Assignment.

**Data Persistence —**

**SwiftData**

**Models**

* **RecordingSession**: groups multiple AudioSegments, with date and optional notes.
* **AudioSegment**: stores unique ID, file path, duration, transcription, upload state.
* **Transcription**: stores text, creation date.

**Why**:

* Clean separation of duties; easy to extend with tags or cateogies later.
* SwiftData handles relationship management.

**Transcription Pipeline**

* Primary: uploads audio to OpenAI Whisper API (OpenAITranscriptionService).
* Retries failed uploads up to maxRetries with exponential backoff + jitter.
* Fallback: uses AppleSpeechRecognizerService with SFSpeechRecognizer if OpenAI fails.

**Why**:

* Protecting against network drops or API issues.
* Ensures the user always gets a transcription, even offline or with local speech recognition.

**UI Architecture — SwiftUI**

* ContentView: shows all RecordingSessions in a List. Scales smoothly to thousands of sessions.
* SessionDetailView: lists segments with transcriptions.
* RecordingSessionSheet: modal recording screen with live timer and waveform.

**Why**:

* Using List leverages SwiftUI’s internal cell reuse, enabling scroll performance with thousands of items.
* Using NavigationStack + sheet ensures simple, maintainable view transitions.

**Accessibility & VoiceOver**

* All UI elements have .accessibilityLabel and .accessibilityHint.
* NavigationLink and buttons describe their purpose clearly for VoiceOver users.

**Why**:

* Meets iOS accessibility guidelines.

**Additional Notes on Non-Functional Requirements**

**Memory Management**

* Buffers hold only the active segment, then clear immediately after writing to disk.

**Battery**

* Uses pause() / stop() on the engine to let iOS suspend audio hardware.

**Storage**

* Deletes audio files after transcription if “Keep Audio Clips” is not set.

**Security**

* OpenAI API Key is stored securely in the Keychain.
* Currently, **audio files on disk are not encrypted**, though the architecture can accommodate that by swapping AVAudioFile writes with a secure file container.

**Why This Architecture?**

**Simple**:

* Seperates audio, data persistence, and transcriptions.
* Easy to swap out transcription services.

**Scalable**:

* Designed to handle thousands of sessions & segments with minimal memory.

**Accessible**:

* Fully supports VoiceOver for assecibility.

**Extendable**:

* Easy to add features like exporting transcriptions, tagging sessions, or encryption at rest.