Audio Transcription Service Specification

Overview

A web-based service that accepts audio files, transcribes them using third-party APIs, and processes speaker labels for easy formatting and export.

Client Description

> *"I want to create a simple web page where we upload a file, and it uses Assembly.AI or OpenAI's Whisper to transcribe the file. Most of the time it will be interviews with 2-3 people. When we get back the text with speaker labels like 'Speaker A' and 'Speaker B', we can input the real names like Nick or Jeremy, and it will replace all those labels. Then we can export it with proper formatting where the names are bolded."*

Project Phases

Phase 1: Upload audio file → Send to transcription API → Receive text → Display text

> *"The first step is to upload the file, and then do the transcription using a third party API. We will then receive back the text of the transcription and display it to the user."*

Phase 2: Process speaker labels → Replace with real names → Format for export

> *"When we get back the text, the first phase is to assign the speaker names to the speaker labels... Then we would move on to the next and say 'hey here's some things that Speaker B had said'... Then it will change all of those Speaker As, Speaker Bs to their names like Nick, Jeremy and Loren."*

Core Features

- Audio file upload and transcription
- Speaker label replacement system
- Text export in Markdown and HTML formats

Technical Requirements

File Handling

- Accepted formats: M4A, MP3
- File size limit: Based on selected API limitations
- Browser-based upload interface

- Original file preservation after transcription

API Integration

- Support for multiple transcription services:
  - Assembly.AI
  - OpenAI Whisper
- Agnostic implementation allowing easy switching between services

Speaker Label Processing

- Extract speaker segments from API response
- Display 4 examples per speaker (max 200 characters each)
- Maximum support for 4 speakers
- Basic name validation (no empty strings, special character handling)
- Single-pass replacement of all speaker labels
- Format: `<strong>SpeakerName:</strong> text`

"If I'm doing an interview with two or three people, Assembly AI will say before the text of each speaker 'Speaker A', 'Speaker B', and 'Speaker C'. When you read the transcript you can see what each speaker said, and then you put that name. Once the user confirms those are correct, we bold all those names and display the text in markdown format for easy copying to the clipboard."

Examples of Speaker Label Flow

1. API returns transcript with generic labels (Speaker A, Speaker B, Speaker C)
2. System shows examples of what each speaker says (first few instances, ~20 words each)
3. User inputs real names (e.g., "Nick" for Speaker A, "Jeremy" for Speaker B)
4. System replaces all instances globally
5. Text is formatted with bold speaker names

Error Handling

- Notify if no speaker labels detected
- Validate speaker names before replacement
- Preserve original transcript for recovery

Export Options

1. Markdown Format:

```
Unset
**SpeakerName:** transcript text
```

2. HTML Format:

```
Unset
<p><strong>SpeakerName:</strong> transcript text</p>
```

User Flow

Upload Phase

1. User uploads audio file (interview recording)
2. System sends to selected API
3. Display raw transcription
4. Offer download of raw transcription

Speaker Processing Phase

1. Extract and display speaker examples
2. User inputs real names for each speaker
3. System performs global replacement
4. Export in chosen format

Interface Approach

- Simple, straightforward interface
- Display all speakers' examples at once
- No progress/loading indicators needed
- Focus on essential functionality

Implementation Notes

Simplification Opportunities

1. Consider browser-side regex for replacements
2. Minimal UI - no progress indicators
3. No speaker mapping storage
4. No metadata tracking
5. Direct replacement without preview step

Security Considerations

- Validate file types before upload
- Sanitize speaker names before replacement
- Secure API key handling

Future Considerations

- Additional speaker validation
- Enhanced formatting options
- Multiple file handling

~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

More information: Questions and Answers

1. Which transcription service do you prefer to use as the default?

- **AssemblyAI** is my preferred service for transcription.

2. Do you already have API keys for the chosen transcription service?

- No, I do not have API keys yet. I will obtain them later.

3. What kind of front-end framework or library would you like to use for the web interface?

- I prefer a simple setup using plain HTML, CSS (inline), and JavaScript for the frontend.

4. Do you have any specific design preferences for the user interface?

- I have no specific design preferences; I want something simple and clean.

5. Which backend technology stack do you prefer?

- I'm open to suggestions but would like the easiest option that can be run on a hosted service like Cloudflare Workers.

6. Will the application need to store transcripts or processed data beyond the session?

- Yes, it would be nice to store transcripts, preferably on Cloudflare if possible.

7. Are there any specific security measures you want implemented?

- For now, we can skip additional security measures since I'll be the only user.

8. What are the expected load and response time requirements for this service?

- There are no specific performance requirements expected at this time.

9. Are there any additional export formats you'd like to support besides Markdown and HTML?

- No additional export formats are needed beyond Markdown and HTML.

10. What environments do you want for development, staging, and production?

- Deployment will be on Cloudflare Workers for all environments (development, staging, and production).

11. How do you plan to deploy the application?

- I plan to deploy on Cloudflare Workers for ease of setup.

12. Do you need to implement any specific privacy policies or terms of service for users uploading audio files?

- No privacy policies or terms of service are currently required.

13. How many users or simultaneous transcriptions do you anticipate this service will need to handle?

- The service will handle one user at a time with low usage, just for me.

14. Are there any specific usability enhancements or accessibility features you want to include?

- No specific usability enhancements or accessibility features are needed at this stage.