Project Documentation: Audio-to-Transcript & Search Tool

April 26, 2025

Overview

This document details the design, setup, and usage of the lightweight web application that ingests audio files, transcribes them to text using Google Cloud Speech-to-Text, and performs web searches on the transcript via Google Custom Search API. It covers both backend (Flask/-FastAPI worker) and frontend (React) components, plus deployment and security considerations.

1 Architecture

- **Frontend**: React.js + TailwindCSS providing file upload, status polling, and search result views.
- **Backend API**: FastAPI (or Flask) running on Cloud Run, exposing endpoints:
  - POST /upload-audio
  - GET /status/{jobId}
  - POST /search
- **Worker**: Python script listening to Pub/Sub, invoking Speech-to-Text, and updating Firestore.
- **Storage**: Google Cloud Storage for audio blobs with TTL lifecycle.
- **Database**: Firestore for job metadata, transcripts, and search results.
- **Pub/Sub**: Asynchronous job orchestration between API and worker.

2 Prerequisites

- Google Cloud project with Billing enabled.
- Enabled APIs: Cloud Storage, Firestore, Pub/Sub, Cloud Run, Speech-to-Text, Custom Search, Vertex AI (optional).
- Service account with roles: Storage Admin, Pub/Sub Editor, Datastore User.
- Local machine: Python 3.10+, Node.js 16+, Git.

3 GCP Resource Setup

1. **Enable APIs**: run

```
gcloud services enable \
  storage.googleapis.com \
  firestore.googleapis.com \
  pubsub.googleapis.com \
  run.googleapis.com \
  speech.googleapis.com \
  customsearch.googleapis.com
```

2. **Create Storage Bucket**:

```
gsutil mb -p YOUR_PROJECT_ID -l us-central1 gs://YOUR_BUCKET_NAME
```

3. **Create Firestore**: via Console or CLI

```
gcloud firestore databases create --region=us-central1
```

4. **Pub/Sub Topic Subscription**:

```
gcloud pubsub topics create audio-processing

gcloud pubsub subscriptions create audio-processing-worker \
  --topic=audio-processing
```

5. **Create Service Account Key**:

```
gcloud iam service-accounts create audio-backend-sa
# grant roles, then:
gcloud iam service-accounts keys create ~/keys/audio-backend-sa.json \
  --iam-account=audio-backend-sa@YOUR_PROJECT_ID.iam.gserviceaccount.com
```

4 Local Environment Setup

1. Clone repository:

```
git clone https://github.com/arshad8049/gdgShazam.git
cd gdgShazam
```

2. Create Python venv and install dependencies:

```
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

3. Frontend setup:

```
cd frontend
npm install
npm start
```

5 Configuration

Copy or create a file named ".env" in the project root with:

```
GOOGLE_CLOUD_PROJECT=YOUR_PROJECT_ID
AUDIO_BUCKET=YOUR_BUCKET_NAME
GOOGLE_APPLICATION_CREDENTIALS=~/keys/audio-backend-sa.json
SEARCH_API_KEY=YOUR_CSE_API_KEY
SEARCH_CX=YOUR_CSE_ID
```

6 Running the Application

6.1 Backend API

```
# in project root
env/bin/uvicorn main:app --reload --host 0.0.0.0 --port 8000
```

6.2 Worker

```
# in separate shell
export GOOGLE_APPLICATION_CREDENTIALS=~/keys/audio-backend-sa.json
python worker.py
```

6.3 Frontend

Access at `http://localhost:3000` to upload audio, poll status, and search the web.

7 Deployment

1. Dockerize backend and worker (provide Dockerfiles).
2. Push images to Container Registry.
3. Deploy to Cloud Run:

```
gcloud run deploy api-service \
  --image=gcr.io/YOUR_PROJECT_ID/api-image \
  --platform=managed \
  --allow-unauthenticated

gcloud run deploy worker-service \
  --image=gcr.io/YOUR_PROJECT_ID/worker-image \
  --platform=managed
```

4. Build and host React frontend on Vercel or Firebase Hosting.

8 Security Considerations

Keep service-account keys out of source control. Rotate keys if exposed. Use Secret Manager or environment variables to store credentials. Enable Audit Logs on GCP.

Acknowledgments

This tool leverages Google Cloud Speech-to-Text, Custom Search, Firestore, Pub/Sub, and FastAPI.