📄 Speech-to-Text System Documentation

1. Introduction

This system provides a **medical-oriented speech-to-text API** built with **FastAPI** and **HuggingFace Transformers**.
It enables:

- Uploading `.wav` files for transcription.
- Live audio recording (demo mode).
- Model switching via **LoRA adapters**.
- Background transcription with optional **Celery + Redis** workers.
- Database-backed management of **patients, recordings, and transcripts**.

---

2. System Architecture

```
             ┌─────────────┐
Upload WAV → │   FastAPI   │ → Store Audio
             └─────────────┘
                  │
                  │
                  ▼
             ┌─────────────┐
             │  Database   │ (SQLModel + SQLite/Postgres)
             └─────────────┘
                  │
                  │
                  ▼
             ┌─────────────┐
             │ ASR Service │ → HuggingFace Whisper / LoRA adapters
             └─────────────┘
                  │
                  ▼
            Transcripts saved to disk
```

Folder Structure

- **app/** – Python scripts for running system

- **stt/** – virtual environment that includes all package
- **uploads/** – stores incoming audio files
- **transcripts/** – stores transcription outputs
- **workers/** – Python scripts for the specific task (ex: transcribe)
- **adapters/** – folder for fine-tuned model's adapter

Set Up for the Test

1. Install the ffmpeg for handling the media files for transformer models on the Hugging Face
   Windows instructions https://www.wikihow.com/Install-FFmpeg-on-Windows
   i) Download the zip file from the website: https://www.gyan.dev/ffmpeg/builds/
   ii) Set up the environment variables for ffmpeg
   iii) Open the command prompt terminal and input:
   set PATH=C:\ffmpeg\ffmpeg-7.1.1-full_build\bin;%PATH%
2. Open the terminal again:
   i) Switch to the directory of the folder: cd <YOUR DIRECTORY>
   ii) Activate the virtual environment: stt\Scripts\activate
   iii) Launch the service: uvicorn app.main:app –reload
   iv) There will be a interal link for you to click on, ex: http://127.0.0.1:8000
   v) Enter the following folder: http://127.0.0.1:8000/docs
   vi) Start to try the speech2text service!

---

3. Installation

3.1 Requirements

- **Python ≥ 3.10**
- **CUDA-capable GPU** (recommended for fast transcription)

Dependencies:

```
pip install fastapi uvicorn[standard] sqlmodel torch transformers peft soundfile
sounddevice scipy celery redis
```

-

3.2 Clone Repository

```
git clone <your_repo_url>
cd speech2text
```

3.3 Environment Setup

Copy `.env.example` → `.env` and adjust values.

```
STT_MODEL_NAME=Na0s/Medical-Whisper-Large-v3
DB_URL=sqlite:///./patient.db
USE_CELERY=False
```

---

4. Running the System

4.1 Local Development

```
uvicorn app.main:app --reload
```

Access API docs at: http://localhost:8000/docs

4.2 With Celery (for scaling)

Run Redis:

```
docker run -d -p 6379:6379 redis
```

Start Celery worker:

```
celery -A app.transcribe worker --loglevel=info
```

Run API server:

```
uvicorn app.main:app --reload
```

4.3 Docker Deployment

Create `Dockerfile`:

```
FROM python:3.11-slim
WORKDIR /app
COPY . /app
RUN pip install -r requirements.txt
CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Build & run:

```
docker build -t speech2text .
```

```
docker run -p 8000:8000 speech2text
```

4.4 Kubernetes (Optional)

For GPU scaling:

- Deploy FastAPI API pods.
- Deploy Celery worker pods with GPUs.
- Use Redis service for task broker.

---

5. Configuration

Settings are defined in `settings.py` and configurable via `.env`.

Setting	Default	Description
`STT_MODEL_NAME`	`"Na0s/Medical-Whisper-Large-v3"`	Base Whisper model.
`STT_ADAPTERS_DIR`	`adapters/`	Directory for LoRA adapters.
`DEFAULT_ADAPTER`	`None`	Default adapter.
`UPLOAD_DIR`	`uploads/`	Uploaded audio storage.
`TRANSCRIPT_DIR`	`transcripts/`	Output transcription storage.

`DB_URL`	`"sqlite:///./patient.db"`	Database URL.
`USE_CELERY`	`False`	Enable Celery workers.
`CELERY_BROKER_URL`	`"redis://localhost/0"`	Redis broker for Celery.
`STT_TARGET_SR`	`16000`	Target sample rate.
`STT_TASK`	`"transcribe"`	Default task.
`STT_LANGUAGE`	`"en"`	Default language.

---

6. Database Models

(SQLModel ORM – `models.py`)

- **Patient** → has many `Recordings`
- **Recording** → linked to a `Patient`, has one `Audio`, stores `adapter_key` & `transcript_path`
- **Audio** → linked to `Recording`, may have `Transcription`
- **Transcription** → text result of an `Audio`

---

7. API Documentation

7.1 Root

- `GET /`
    - Health check.

7.2 Models

- `GET /models`
    - Lists available LoRA adapters.

7.3 Upload Audio

- `POST /upload/`
    - Params:
        - `patient_id`: str
        - `file`: `.wav` file
        - `adapter`: optional adapter key
    - Returns `{ "status": "queued", "audio_id": <id> }`

7.4 Live Recording (demo only)

- `POST /record/start/`
    - Start recording on server host.
- `POST /record/stop/`
    - Stop recording, save, and queue transcription.

7.5 Fetch Transcription

- `GET /transcription/{audio_id}`
    - Retrieves transcript file.

    - Status:
        - `404`: unknown audio
        - `202`: transcript not ready
        - `200`: transcript file

---

8. Developer Guidelines

8.1 Extending Models

- Add new LoRA adapters under `adapters/{adapter_name}/adapter_config.json`.
- Modify `model_registry.py` if additional pipeline settings are needed.

8.2 Adding New Endpoints

- Use **FastAPI** decorators in `main.py`.
- For database access, depend on `get_session()` from `database.py`.

8.3 Scaling

- For GPU scaling: set `USE_CELERY=True`.
- Deploy multiple Celery workers across nodes.

8.4 Audio Preprocessing

- `audio_io.py` ensures audio is always:
    - Mono
    - 16 kHz sample rate

---

9. Example Usage

Upload & Transcribe

```
curl -X POST "http://localhost:8000/upload/" \
  -F "patient_id=123" \
  -F "file=@sample.wav"
```

Fetch Transcript

```
curl http://localhost:8000/transcription/1
```

List Adapters

```
curl http://localhost:8000/models
```

---

10. Troubleshooting

- **Error: Only WAV files accepted**
  → Ensure uploads use `.wav` format.
- **Transcript not ready (202)**
  → Background task still running.
- **CUDA out of memory**
  → Reduce batch size or switch to CPU (`device=-1` in registry).
- **Recording fails in Docker**
  → Live recording uses ALSA (`sounddevice`), not supported in containerized/cloud deployments. Use client-side recording instead.