Whisper Transcriber

Author: Gopichand Busam

A minimal Flask web UI for uploading an audio file and downloading the transcription using OpenAI Whisper.

Features

- Drag & drop single audio file upload
- Automatic transcription with selected Whisper model size (default: `base`)
- Download resulting `.txt` file automatically
- Helper management script (`manage.sh`) to setup, run, stop, clean, and purge environment/resources
- Live progress bar with heuristic or chunk-level updates
- ETA countdown (derived from model + audio duration)
- GPU detection with adjusted performance estimates (if PyTorch CUDA available)
- Multi-job dashboard with ability to attach to any recent job
- Cancel running transcription (process-based)
- Preprocessing phase with audio caching (.npy) and ETA for all stages (preparing, queued, ready, processing)

Requirements

- macOS / Linux (bash)
- Python 3.9+ (Whisper recommends Python 3.8+; use latest stable for best results)
- Sufficient disk space for model weights (hundreds of MB depending on model size)

Quick Start

```
# Show available commands
./manage.sh help

# (1) Setup environment + dependencies + download default model (base)
./manage.sh setup

# (2) Run the app (foreground)
./manage.sh run
# Visit: http://127.0.0.1:5000

# OR run in background
./manage.sh run-bg
./manage.sh status

# (3) Stop background app (if used)
./manage.sh stop
```

```
# (4) Optional: clean caches, remove venv & uploads when done
./manage.sh purge
```

If `manage.sh` is not executable yet:

```
chmod +x manage.sh
```

Selecting a Different Whisper Model

Set `WHISPER_MODEL_SIZE` before running:

```
WHISPER_MODEL_SIZE=small ./manage.sh run
```

Choices include: `tiny`, `base`, `small`, `medium`, `large`, `large-v2`. Larger = slower + more accurate.

Model Size Trade-offs

Model	Relative Speed	Relative Accuracy	Approx RAM (loaded)	Typical Use
tiny	Very fast	Lowest	~1 GB	Quick preview, rough gist
base	Fast	Low	~1.3 GB	Simple English audio
small	Moderate	Medium	~2.5 GB	Better accuracy, still quick
medium	Slower	High	~5 GB	Higher quality multi-language
large / v2 / v3	Slowest	Highest	6–10+ GB	Maximum accuracy, diverse accents

Guideline: Start with `small` for a balance; escalate only if quality insufficient.

Decoding / Advanced Options (UI Panel)

These are exposed in the web UI. They map to Whisper decoding parameters.

Option	What It Does	Typical Range	Impact
Task	`transcribe` (keep source language) or `translate` (to English)	transcribe/translate	Changes output language
Temperature	Sampling randomness. 0 = deterministic, higher = more varied	0.0 – 1.0	Higher can help stuck decoding but may reduce consistency

Option	What It Does	Typical Range	Impact
Beam Size	Number of beams in beam search (explores alternatives)	1 – 10 (max 20)	Larger can increase accuracy & latency
Best Of	Samples N candidates and picks best by log-prob (used with sampling)	1 – 10 (max 20)	Improves quality with non-zero temperature
Language	Force language code (e.g. `en`, `es`, `fr`). Leave blank for auto-detect	ISO 639-1	Setting can speed start & accuracy if known

Tuning Tips:

1. For fastest runs: temperature 0, beam_size 1, best_of 1.
2. If output seems truncated or low quality: raise beam_size to 5, keep temperature 0.
3. For creative / ambiguous audio: temperature 0.2–0.4; optionally best_of 3.
4. Avoid setting large beam_size AND large best_of simultaneously unless you accept longer runtimes.

Resource Impact (rough qualitative):

```
Latency ↑ ~ proportional to (beam_size + best_of + model_size)
Memory ↑ with model_size only (decoding params affect compute, not
loaded weights)
```

Progress & ETA Details

Two progress strategies are supported:

1. Estimate (default): Uses audio duration and heuristic real-time factors (RTF) for the selected model (separate CPU vs GPU tables) to estimate percent complete and ETA.
2. Chunk Mode: If you set environment variable `PROGRESS_MODE=chunks` and the audio is longer than ~35s, the worker will manually segment audio into ~30s pieces, transcribing sequentially and emitting real progress updates after each chunk. This adds a small overhead but gives more tangible movement for long files.

To enable chunk mode:

```
PROGRESS_MODE=chunks ./manage.sh run
```

GPU Acceleration

If PyTorch detects a CUDA-capable GPU at runtime, the app switches to GPU real-time factor estimates and displays a banner (e.g., `GPU Acceleration Active: NVIDIA ...`). Install an appropriate GPU

build of PyTorch separately—this project does not pin a CUDA wheel by default. Falling back to CPU prints no banner.

Multi-Job Dashboard

Audio Preprocessing & .npy Cache

When you upload a file the app enters a short "preparing" phase:

1. Loads and resamples audio to 16 kHz mono (Whisper requirement).
2. Normalizes dtype to float32.
3. Saves the waveform as a NumPy array (`original.ext.npy`).

Why a .npy file?

- Faster subsequent access: direct memory map instead of re-decoding container.
- Enables potential reuse across different parameter runs without reloading raw media.
- Simplifies chunk/segment slicing (already in aligned sample space).

If the .npy exists it is used preferentially by the worker to reduce I/O latency. The preprocessing step shows its own progress & ETA derived from audio length (heuristic) and then transitions the job to `ready`.

ETA Semantics

Status	ETA Meaning
preparing	Remaining estimated preprocessing time (audio load/normalize/cache)
ready	Estimated transcription time if started now (heuristic model RTF)
queued	Same as ready (processing not begun)
processing	Remaining transcription time (elapsed-adjusted)
done/error/cancelled	0

Below the main upload area a Recent Jobs table lists recent transcriptions with status and progress. Click a row to attach your UI (logs, progress, download) to that job. This is in-memory only; restarting the app clears history. For persistence, integrate a database / queue (e.g., Redis, Postgres, RQ, Celery).

Directory Layout

```
app.py              # Flask application
manage.sh           # Management / automation script
requirements.txt    # Python dependencies
uploads/            # Uploaded audio + generated text (created runtime)
output/             # (Reserved) Additional outputs / logs
static/             # Static assets (CSS)
templates/          # HTML templates
```

Management Script Commands

Command	Description
setup	Create virtualenv + install dependencies + warm model cache
run	Run app in foreground
run-bg	Run app in background (PID stored in .app.pid)
stop	Stop background run
status	Show background run status
clean	Remove **pycache** + .pyc files
reset	Remove venv + Whisper model cache (~/.cache/whisper)
purge	reset + delete uploads/ & output/
shell	Activate venv subshell

Production Considerations

This project is a minimal prototype. For production deployment:

- Use a proper WSGI/ASGI server (e.g., gunicorn + gevent/uvicorn workers)
- Put behind a reverse proxy (Nginx / Caddy)
- Add file size limits & validation (e.g., restrict mime types)
- Add user authentication / rate limiting if exposed publicly
- Use persistent storage or object store if you need transcripts later
- Consider batching / queueing heavy transcriptions
- Monitor GPU/CPU utilization if using larger models

Manual Setup (Without Script)

```
python3 -m venv .venv
source .venv/bin/activate
pip install --upgrade pip
pip install -r requirements.txt
python app.py
```

Cleaning Up

```
./manage.sh stop      # if running in background
./manage.sh purge     # removes env, model cache, uploads, outputs
```

Troubleshooting

Issue	Fix
`Command not found: python3`	Install Python 3 from python.org or via Homebrew (`brew install python`)
Slow first run	First model download & load can take time; subsequent runs faster
Out of disk space	Use `./manage.sh reset` or `purge` to remove model + env
Need GPU acceleration	Install PyTorch with CUDA matching your GPU (not covered in default requirements)

Author

Gopichand Busam
Website: https://gopichand.me

License

Contributing

Contributions are welcome! Please feel free to submit a Pull Request. For major changes, please open an issue first to discuss what you would like to change.

Support

If you find this project helpful, please consider:

- ⭐ Starring the repository
- 🐛 Reporting bugs or issues
- 💡 Suggesting new features
- 🔗 Sharing with others

Built with ❤️ by Gopichand Busam