PROJECT ABSTRACT

Podcast Chapter Generator using Whisper

An automatic podcast chaptering pipeline: OpenAI's pre-trained Whisper transcribes episodes with timestamps, heuristic segmentation proposes chapters, and a web UI lets creators review boundaries on a waveform before exporting YouTube, MP3 and JSON chapters — built-to-order with code, report, PPT and viva kit.

01. Title

Podcast Chapter Generator using Whisper

02. Introduction / Background

Podcast episodes routinely run one to three hours, yet most players offer nothing better than a scrub bar. Chapters — titled, timestamped segments — fix navigation, but hand-writing them for every episode is tedious enough that most creators skip it. Speech recognition changed the economics: OpenAI's Whisper, a transformer-based model pre-trained on large-scale weakly supervised audio, produces timestamped transcripts robust enough to work from. Topic shifts and long silences in the transcript then reveal natural chapter boundaries, and extractive key phrases provide draft titles. This project wires those pieces into a complete pipeline — transcribe, segment, title, review on a waveform, export — delivered as a Python backend plus a web UI.

03. Problem Statement

Long-form podcast audio is hard to navigate and expensive to chapter by hand, so most episodes ship without chapters and listeners cannot jump to the segments they want. The project addresses this with an automatic pipeline: Whisper transcription with segment timestamps, silence-plus-topic-shift segmentation into chapters, extractive titling from each segment's distinctive phrases, and an interactive waveform editor where the creator reviews and adjusts boundaries before one-click export to YouTube chapter text, MP3 chapter tags and JSON. Transcription quality and boundary placement are explicitly treated as reviewable, not perfect.

04. Objectives

- Build a transcription stage using pre-trained Whisper models (tiny through large, selectable) via faster-whisper, with segment-level timestamps.
- Implement chapter segmentation combining long-silence detection with topic-shift signals in the transcript and a configurable target chapter length.

- Generate extractive chapter titles from each segment's most distinctive phrases, kept editable and inspectable.
- Build a web UI rendering the episode as a chapter-colored waveform with draggable boundaries, split/merge and retitling.
- Implement one-click export to YouTube description chapters, MP3 ID3 chapter tags and JSON.
- Support batch queueing of multiple episodes and expose model size, language hint and thresholds as settings.
- Deliver the complete student kit: source code, setup guide, report, PPT and viva Q&A.

05. Proposed Methodology

Uploaded audio is normalized with FFmpeg to 16 kHz mono, the format Whisper expects. faster-whisper (CTranslate2 runtime) runs the selected pre-trained Whisper model and returns segments with start and end timestamps — no training is performed; the published pre-trained weights are used as-is. A segmentation stage scores candidate boundaries from pause durations and lexical topic shifts, then selects a boundary set near the configured target chapter length. Each chapter receives an extractive title drawn from its most distinctive phrases. The web UI renders the waveform with chapter regions; the creator drags boundaries, splits or merges chapters and edits titles. Exporters then serialize the reviewed chapter list to YouTube-format text (MM:SS titles), MP3 chapter tags via FFmpeg metadata, and JSON.

06. System Architecture / Working

The system has four stages. (1) Ingest: upload plus FFmpeg normalization. (2) Transcription: faster-whisper with the selected Whisper model, producing timestamped segments. (3) Structuring: silence/topic-shift segmentation and extractive titling. (4) Review and export: the waveform editor UI and the three exporters. A Flask backend orchestrates the pipeline and serves the HTML/CSS/JS frontend; a batch queue processes multiple episodes sequentially with per-episode logs. The working flow is: upload episode → normalize → transcribe → segment and title → review on waveform → export.

07. Hardware / Software Requirements

- Python 3.10, faster-whisper, FFmpeg, Flask, HTML/CSS/JS
- OpenAI Whisper pre-trained weights (open-source; one-time download)
- CPU works with smaller models; CUDA GPU strongly recommended for medium/large
- 8 GB RAM recommended; disk space for audio and intermediate files

08. Expected Outcomes

A working pipeline that converts a podcast episode into reviewed, titled chapters exportable to YouTube, MP3 and JSON, with a waveform review UI and batch queue. The reproducible outcome is the working system plus documentation of its behavior on the student's own test episodes — no word-error-rate or segmentation-accuracy figure is pre-claimed, because both depend on the recording (accents, crosstalk, music, noise), which the report discusses openly.

09. Applications

- Podcast production: show notes and chapter generation for independent creators
- Lecture and webinar recordings: navigable chapters for long educational audio
- Teaching aid for applied speech-AI, audio processing and heuristic NLP coursework
- Baseline for research into topic segmentation of spoken content

10. Future Scope

- Speaker diarization so chapters can be labeled by speaker turns
- Abstractive chapter titles via a language model, with human review
- Direct publishing integrations (YouTube API, podcast hosts)
- Multi-language titling tuned per language