

**Semester Long Internship Report**

On

**Auto Generation of Spoken Tutorials App**

Submitted by

**Hitesh,**

**National Institute Of Technology Delhi**

Under the guidance of

**Prof. Kannan M Moudgalya**

Chemical Engineering Department

IIT Bombay

August 6, 2022

Acknowledgment

I, the FOSSEE intern of the **Auto Generation of** **Spoken-Tutorial App**, is overwhelmed in all humbleness and gratefulness to acknowledge our deep gratitude to all those who have helped us put our ideas to perfection and have assigned tasks, well above the level of simplicity and into something concrete and unique

I, wholeheartedly thank **Nancy Ma’am** for having faith in us, selecting us to be a part of his valuable project and for constantly motivating us to do better.

I am very thankful to our mentors **Kirti Ma’am and Rohan Sir** for their valuable suggestions. They were and are always there to show us the right track when needed help. With help of their brilliant guidance and encouragement, we all were able to complete our tasks properly and were up to the mark in all the tasks assigned. During the process, I got a chance to see the stronger side of our technical and non-technical aspects and also strengthen our concepts. Hereby, I gladly consider ourselves to be the most fortunate batch of interns.

Last but not the least, I wholeheartedly thank all our other colleagues working in different projects for helping us evolve better with their critical advice.

With Regards.

Hitesh

(NIT Delhi)

Contents

[**1. Introduction**](#_n1vsv9947u7v) **4**

[**2. Auto**](#_9kmricj8haop) **Generation App 5**

[**3. Problem Statements**](#_saweapb1ao1k) **6**

[3.1 Problem 1](#_yhgplwrimau3) 6

[3.1.1 Problem 1.1](#_ncynf9gt26hk) 6

[Proposed Solution 1.1](#_tgmr03v75eyf) 6

[3.1.2 Problem 1.2](#_r8qy26gh81b8) 6

[Proposed Solution 1.2](#_9bvlutqjjqzb) 7

[3.1.3 Problem 1.3](#_e6o0spmmwfd5) 7

[Proposed Solution 1.3](#_xocuewxsdek4) 7

[3.2 Problem 2](#_x2jqhsxj44gb) 7

[3.2.1 Problem 2.1](#_xyjn8j3l9jcr) 7

[Proposed Solution 2.1](#_puxl8igl6bd9) 7

[3.2.2 Problem 2.2](#_tfsdua8fe4ip) 8

[Proposed Solution 2.2](#_gh4i3iois29t) 8

[**4. Design Considerations**](#_5e97tj7z35so) **9**

[4.1 Dependencies and Packages Used](#_ypl7i2vak7wf) 9

[4.2 Technologies Used](#_kzvhhg182udh) 10

[**5. Tasks in detail**](#_oqs76ed73akt) **11**

[5.1 Get audio from transcript](#_r835q07vwjzf)  11

[5.2 Merging Video with TTS audio](#_mkk6ke68q9pp)  12

[**6. Diagrams**](#_3g5a5xisixc9) **13**

[6.1 Sequence](#_7mgferpze7bv) Diagram 13

[**7. Future**](#_3g5a5xisixc9) **Scope 15**

[7.1](#_7mgferpze7bv) Using Google Cloud TTS / Microsoft TTS 15

7.2 Processing of different languages 15

[**8. References**](#_j15e0pqk49z9) **15**

# 1. Introduction

The Spoken Tutorial project is the initiative of the ‘Talk to a Teacher’ activity of the National Mission on Education through Information and Communication Technology (ICT), launched by the Ministry of Human Resources and Development, Government of India.

The use of spoken tutorials to popularize software development and its use will be coordinated through this website.

(The Spoken Tutorial project is being developed by IIT Bombay for MHRD, Government of India)

The spoken Tutorial Project aims to make spoken tutorials on FOSS available in several Indian languages, for the learner to be able to learn in the language he/she is comfortable in. Our goal is to enable the use of spoken tutorials to teach in any Indian language, and to be taught to learners of all levels of expertise- Beginner, Intermediate or Advanced

This project is for the community and by the community. Through the portal, we aim to reach out to like-minded individuals to collaborate with us and with each other to create Spoken Tutorials. The next step is to get each Spoken Tutorial dubbed into as many Indian languages as possible. This will help anyone anywhere to understand the contents of the Spoken Tutorials. Each of the Tutorials, whether original or dubbed, go through a strict review procedure, after which they are uploaded on the public domain. This is to ensure that the highest possible quality is attained.

# 2. Auto Generation App

The Auto Generation App is all about *automating the process of creating spoken tutorials*, by creating audio from given transcript (with custom rate of speech and gender), merge it with the given video, and return the final merged media file. There are different roles in creating a spoken tutorial and they are assigned accordingly. There are contributors who create and upload videos on our website, then we have some reviewers who will review these uploaded tutorials, we also have an admin team which will cross-verify the audio-video quality, and if everything is proper then they will publish the video on the website.

So each and every video undergoes several processes, starting from the creation of the tutorials which are uploaded by the contributors. This is followed by several quality checks, for which we have a dedicated team. This team performs several noise, loudness, video resolution and several others audio and video quality checks. After the video has passed all the quality checks, the review team removes extra noise from the video, otherwise it is sent for need-improvement state where the contributor has to modify their video. After removing the noise, video is stored in the system and is transferred to the admin for the final review, who after a thorough review publishes the video onto the website for the users to view.

# 3. Problem Statements

## 3.1 Problem 1

The existing situation requires every contributor, irrespective of the language, to record an audio corresponding to the silent video of the slides provided to them by the Spoken Tutorials team. They, then merge their audio file recordings with the video and upload to the website for further review. Now, the video file as uploaded by the contributor may contain noise or mistakes. This highlights the problem of needing to re-record the audio as many times as needed till quality check is passed.

### 3.1.1 Problem 1.1

The audio files uploaded by the contributors are prone to the effects caused by some unwanted signals occurring in the video that may cause disturbances in the actual audio and reduce the impact and flow of the recording. Some of these “noise” include unintentional breathing sounds with the recording or sound of fan or air in the background etc.

#### Proposed Solution 1.1

The audio files can be generated using TTS tools instead of manually recording to avoid such noises and enhance the quality of speech. This also would allow the uploader to keep uniform pace of speech.

### 3.1.2 Problem 1.2

It is possible that the recording by the contributor might not get carried out as expected due to some problems in hardware/software at the contributor’s end. This may lead to change in properties of the signal that makes it unsuitable for uploading, mainly due to variation in loudness of sound that may be too loud or too dull.

#### Proposed Solution 1.2

The audio file generated by TTS would have normalized amplitude, so uploader would not have face the issue of uneven or varied sounds.

### 3.1.3 Problem 1.3

What about the files present after processing on the server? How will they be managed if they are of no use after being submitted by the contributor?

#### Proposed Solution 1.3

A cron is implemented that will run every week to clear the media folder, as it is just wasting up the space.

## 3.2 Problem 2

The spoken tutorial videos all have a Timed Script with timestamps/cues for when which line is to be said next.

### 3.2.1 Problem 2.1

What format would be suitable for transcripts and how would we make sure the right lines are said at the right timestamps?

#### Proposed Solution 2.1

The script will accept CSV (.csv) format files which can be parsed using Pandas library, additionally a function can be called that will keep track of how much difference is present between ending of one line and starting of next, so silence can be padded to previous audio track accordingly.

### 3.2.2 Problem 2.2

As timed script is created after audio recording is submitted, we need the script to be able to work regardless of the presence of timestamps.

#### Proposed Solution 2.2

The script would simply append all the audio tracks without any silence padding, there may be some timing issues with this however if recording is not done with a fix WPM (words per minute) in mind.

# 4. Design Considerations

## 4.1 Dependencies and Packages Used

* **FFMPEG**: It is a complete, cross-platform solution to record, convert and stream Audio and Video. All the important conversions between different file formats are done using FFMPEG.
* **espeak-ng**: Open source Text-to-Speech synthesizer that supports more than hundred languages and accents.
* **mbrola**: The MBROLA is an open source speech engine with collection of diphone voices for speech synthesis. MBROLA does not include any text-to-phoneme translation, so this must be done by another program. eSpeak-NG can be used as a front-end to MBROLA, to provide spelling-to-phoneme translation and intonation, which MBROLA then uses to generate speech sound.

## 4.2 Technologies Used

* **HTML**: It is the main mark-up language for displaying web pages and other information that can be displayed in a web browser.
* **CSS**: It is a style sheet language used for describing the presentation semantics (the look and formatting) of a document written in a mark-up language.
* **JavaScript**: It is a prototype-based scripting language that is dynamic, weakly typed and has first-class functions and is mainly used for validation etc.
* **PYTHON**: High-Level Scripting Language for Back-End coding and for server side programming. Python features a dynamic types system and automatic memory management and supports multiple programming paradigms.
* **DJANGO**: Python framework for web application development. It follows MVC (Model View Controller) structure for managing the models and controlling the views.
* **MUTAGEN:** Mutagen is a Python module to handle audio metadata. It supports ASF, FLAC, MP4, Monkey’s Audio, MP3, Musepack, Ogg Opus, Ogg FLAC, Ogg Speex, Ogg Theora, Ogg Vorbis, True Audio, WavPack, OptimFROG, and AIFF audio files.
* **PANDAS:** It is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool.

# 5. Tasks in detail

## 5.1 Get audio from transcript

* Transcript file is processed first, it parses the CSV file and generates output, which is then passed for further processing.





## 5.2 Merging Video with TTS audio

* Ffmpeg picks up where espeak-ng leaves. It concatenates audios line by line and merges final result with the given video.

 

|  |  |
| --- | --- |
| **Task** | **Status** |
| **Generate audio with timed script**: Returns audio file with timed script. |  **✅** |
| **Generate audio without timed script**: Returns audio file as well. |  **✅** |
| **Merge generated audio with given video**: Returns a merged video. Audio is first stripped then generated audio is added. |  **✅** |

# 6. Diagrams

## 6.1 Sequence Diagram

The sequence diagram for generating audio from transcript is as followed:



The sequence Diagram for merging of files is as followed:



# 7. Future Scope

## 7.1 Using Google Cloud TTS / Microsoft TTS

Current implementation uses open source TTS (e-speak and MBROLA) but with neural voices, the result would be far better compared to the one in use.

## 7.2 Processing of different languages

As of now, only English language is supported but different languages can be parsed and processed using a similar approach, given the TTS engine supports them.

# 8. References:

* **Popcorn.js:** [**https://github.com/mozilla/popcorn-js**](https://github.com/mozilla/popcorn-js)
* **FFmpeg package:** [**http://ffmpeg.org/ffmpeg.html**](http://ffmpeg.org/ffmpeg.html)
* **eSpeak-NG:** [**https://github.com/espeak-ng/espeak-ng**](https://github.com/espeak-ng/espeak-ng)
* **MBROLA:** [**https://github.com/numediart/MBROLA**](https://github.com/numediart/MBROLA)
* **Django:** [**https://www.djangoproject.com**](https://www.djangoproject.com)
* **gTTS:** [**https://gtts.readthedocs.io/en/latest/**](https://gtts.readthedocs.io/en/latest/)
* **Microsoft TTS:** [**https://azure.microsoft.com/en-us/services/cognitive-services/text-to-speech/**](https://azure.microsoft.com/en-us/services/cognitive-services/text-to-speech/)
* **Google Cloud TTS:** [**https://cloud.google.com/text-to-speech/**](https://cloud.google.com/text-to-speech/)