Summer Fellowship Report

On

Video Processing System for Spoken Tutorial

Submitted by

Pratik Daigavane

Under the guidance of

Prof. Kannan Moudgalya
Chemical Engineering Department
IIT Bombay

May 2020

Acknowledgement

I, the summer intern of the **FOSSEE - Video Processing System for Spoken Tutorial Project** am overwhelmed in all humbleness and gratefulness to acknowledge my sincere gratitude to all those who have helped me put my ideas to perfection and have assigned tasks well above the level of simplicity and into something concrete and unique. I wholeheartedly thank **Prof. Kannan M. Moudgalya** for having faith in me, selecting me to be a part of his valuable project, and for continually motivating me to do better. I thank **Mr. Nagesh Karmali** and **Ms. Firuza Aibara** for providing me with the opportunity to work on this project. I am also very thankful to my mentors for their valuable suggestions. They were and are always there to show me the right track when needed help. With the help of their brilliant guidance and encouragement, I was able to complete my tasks correctly and was up to the mark in all the assigned tasks. During the process, I got a chance to see the stronger side of my technical and non-technical aspects and strengthen my concepts. Last but not least, I sincerely thank all other colleagues working in different projects under **Prof. Kannan M. Moudgalya** for helping me evolve better with their critical advice.

Declaration

I declare that this written submission represents my ideas in my own words, and whenever others' ideas or words have been included, I adequately cited and referenced the sources. I declare that I have accurately and adequately acknowledged all sources used in the production of this thesis.

I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/-source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources which have not been appropriately cited or from whom proper permission has not been taken when needed.

Pratik Daigavane

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Statement

Design and develop a web-based platform for Spoken Tutorial, for streamlining the process of editing the tutorials.

1.2 Project Objective

The project's objective is to add a module to the existing spoken-tutorial.org website so that contributors can seamlessly edit tutorials. Revisions history should be maintained after each edit and can be reverted to the previous version. A reviewer should be able to review and comment on that tutorial.

1.3 Project Outcome

The contributor will be able to edit the tutorial using a web-based dashboard. A tutorial will be broken into chunks based on subtitles, and their timings, the contributor, will be able to edit the audio and subtitle of each chunk. After each edit, revisions will be maintained, and the contributor will have the option to compare versions and revert to the previous version. Once the contributor edits the tutorial, he/she can send it for review. The existing publishing workflow will be followed in the review process.

1.4 Project Requirements

Following technologies have been used during the development of this project.

- Django (v1.11)
- React (v16.13.1)
- Docker Containers
- MySQL
- Celery
- Redis
- FFmpeg

Chapter 2

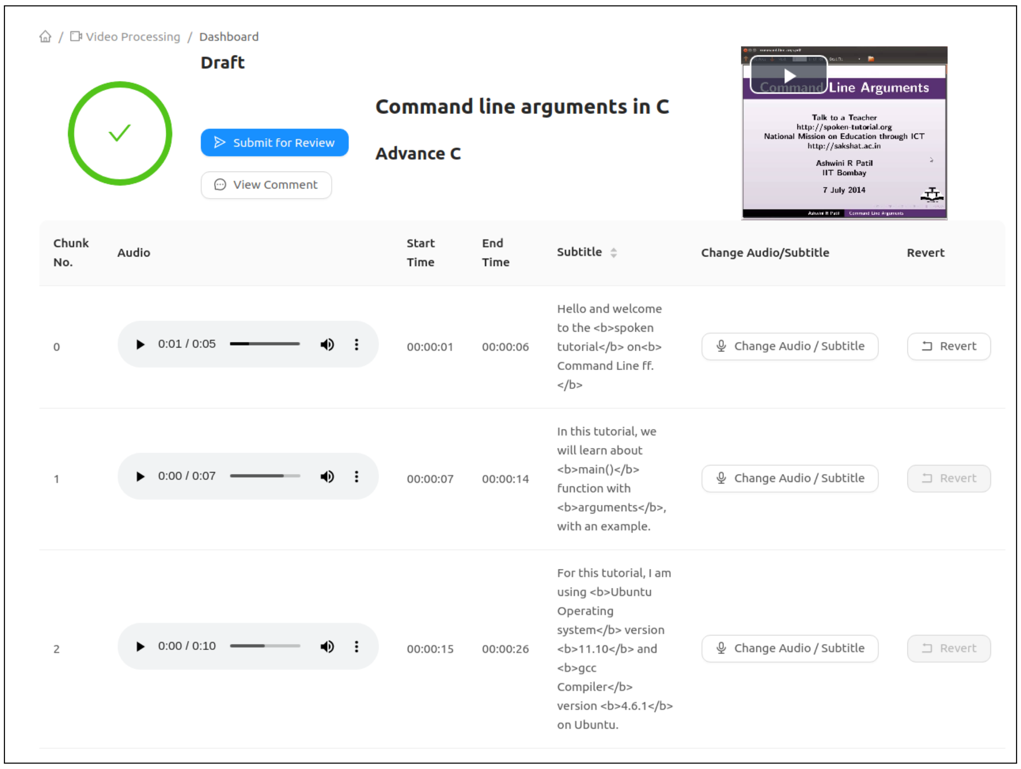
Project Overview

2.1 Current System

In the current system of Spoken Tutorial, the tutorial (video) is recorded by an individual, which is then approved through some process. The final output is a video in English. This tutorial is then passed on to others for dubbing in different languages, where each language contributor creates a tutorial (video) in the respective language. A timed script is also created manually. **The current limitation is that if a change is needed in a particular script, the entire tutorial must be re-created.**

2.2 Proposed Solution

The video (English or dubbed) will be uploaded by the contributor as usual. When the contributor selects a particular tutorial to be edited; that tutorial will be broken into various chunks based on the subtitles and their timings. Then the contributor will be provided with a facility to edit video.



/ Video Processing / Dashboard

Draft

Submit for Review

View Comment

Command line arguments in C

Advance C

Command Line Arguments

Talk to a Teacher
http://spoken-tutorial.org
National Mission on Education through ICT
http://sakshat.ac.in

Ashwini R Patil
IIT Bombay

7 July 2014

Chunk No.	Audio	Start Time	End Time	Subtitle	Change Audio/Subtitle	Revert
0	0:01 / 0:05	00:00:01	00:00:06	Hello and welcome to the <b>spoken tutorial</b> on<b> Command Line ff. </b>	Change Audio / Subtitle	Revert
1	0:00 / 0:07	00:00:07	00:00:14	In this tutorial, we will learn about <b>main()</b> function with <b>arguments</b>, with an example.	Change Audio / Subtitle	Revert
2	0:00 / 0:10	00:00:15	00:00:26	For this tutorial, I am using <b>Ubuntu Operating system</b> version <b>11.10</b> and <b>gcc Compiler</b> version <b>4.6.1</b> on Ubuntu.	Change Audio / Subtitle	Revert

Figure 2.1: Video Editing Dashboard

A chunk represents a part of the video which starts and ends at some specific times. For example: in the above diagram, Chunk 1 represents part of the tutorial, which starts at 00:00:07 and ends at 00:00:14.

The contributor will be able to change the audio and subtitle of chunks as needed. Revisions history will be maintained after each edit and can be reverted to the previous version. After editing is done, the contributor will have an option to submit the tutorial for review.

Chapter 3

Architecture

3.1 Overview

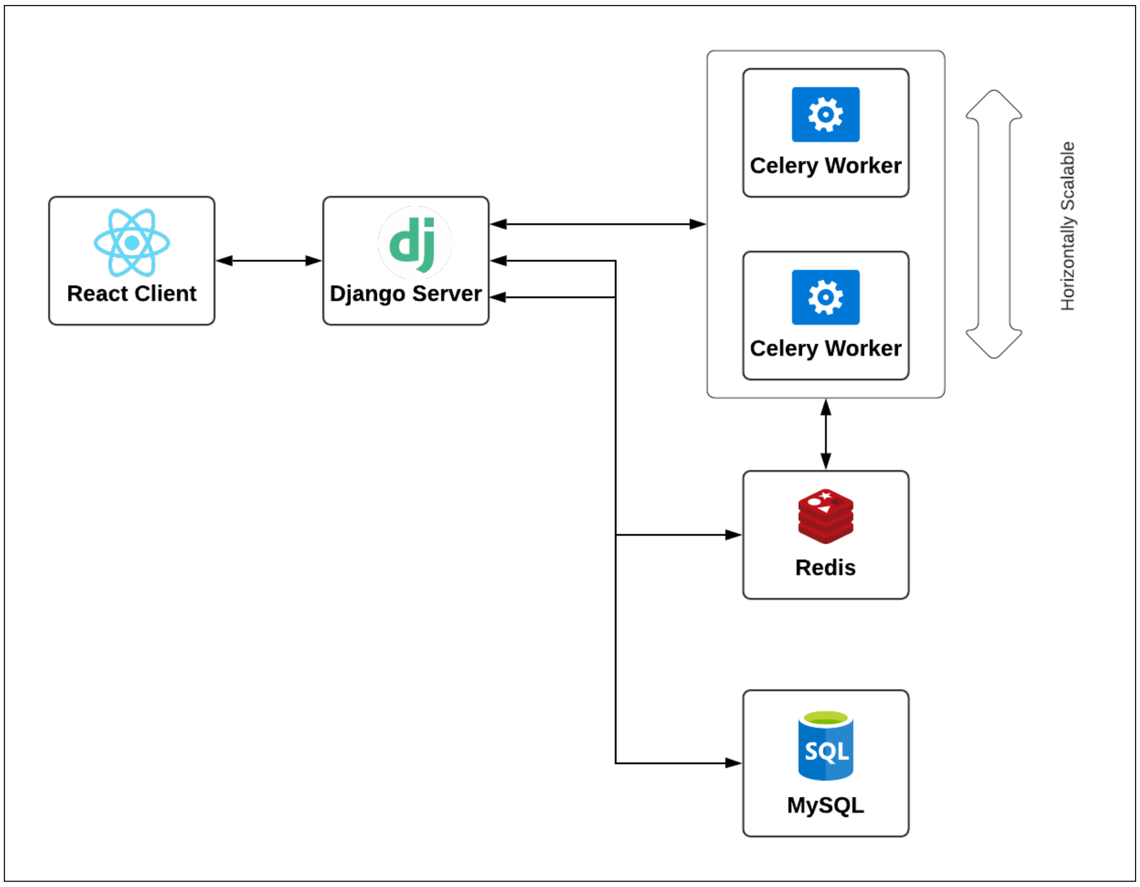


Figure 3.1: Architecture Diagram

3.2 React

React [1] is used for developing the frontend. This project comprises of two SPAs (Single Page Application): one each for contributor and reviewer.

3.3 Django

Django[2] Framework along with Django-Rest-Framework[3] have been utilised to create API endpoints required for the frontend.

3.4 Celery and Redis

Celery[4] uses the same environment as Django but runs the celery daemon with multiple workers to handle heavy processing tasks in the background asynchronously. It is also worth noting that multiple celery containers can 'discover' each other on the network and share tasks amongst themselves. Celery also requires a Task Queue, for which Redis has been utilised. Redis[5] is essentially an In-memory Key-Value Store; this is utilised to store the task details, and it's data which is later fetched and processed by a Celery worker.

3.5 FFmpeg

FFmpeg[6] is the primary media processing library used in this project. FFmpeg is a high-speed video and audio converter that can also grab from a live audio/video source. It can also convert between arbitrary sample rates and resize a video on the fly with a high-quality polyphase filter. The transcoding process in FFmpeg for each output can be described in Figure 3.2

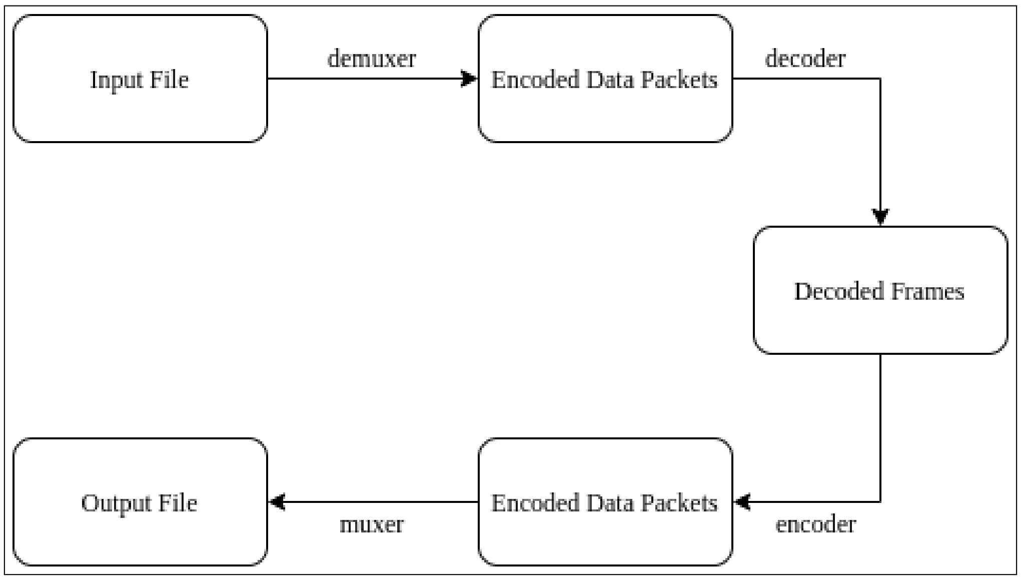


Figure 3.2: Transcoding process in FFmpeg

FFmpeg calls the libavformat library (containing demuxers) to read input files and get packets containing encoded data from them. When there are multiple input files, FFmpeg tries to keep them synchronised by tracking the lowest timestamp on any active input stream.

Encoded packets are then passed to the decoder (unless streamcopy is selected for the stream). The decoder produces uncompressed frames (raw video/PCM audio/...), which can be processed further by filtering (see next section). After filtering, the frames are passed to the encoder, which encodes them and outputs encoded packets. Finally, they are passed to the muxer, which writes the encoded packets to the output file.

3.5.1 Filtering

Before encoding, FFmpeg can process raw audio and video frames using filters from the libavfilter library. Several chained filters form a filter graph. FFmpeg distinguishes between two types of filtergraphs: simple and complex.

Simple filtergraphs

Simple filtergraphs are those that have exactly one input and output, both of the same type. In Figure 3.2, they can be represented by simply inserting an additional step between decoding and encoding:

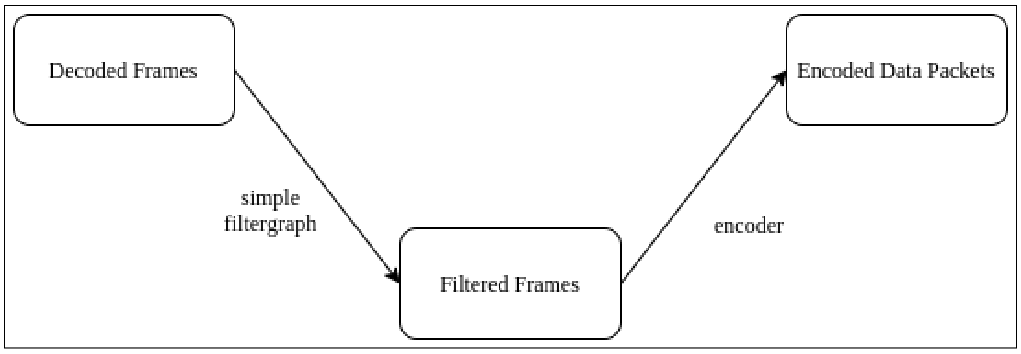


Figure 3.3: Simple filtergraph

Complex filtergraphs

Complex filtergraphs are those that cannot be described simply as a linear processing chain applied to one stream. This is the case, for example, when the graph has more than one input and/or output, or when output stream type is different from the input.

Chapter 4

Logic Flow

This chapter gives a high-level overview of the logic flow of the system.

4.1 Creating Chunks

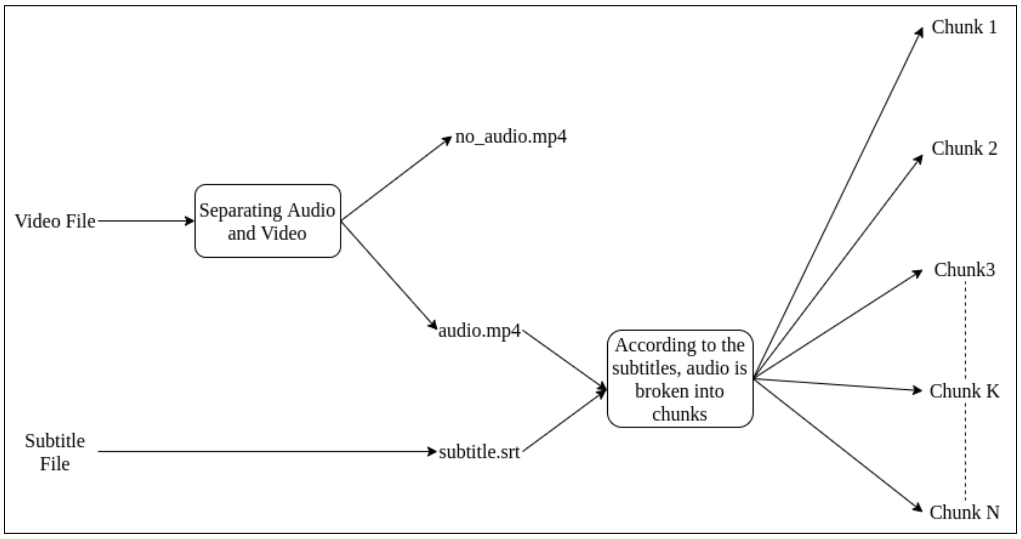


Figure 4.1: Creating Chunks

This is the initial step while processing the tutorial. The audio is first extracted from the tutorial's video file and is then broken into chunks, taking into consideration the subtitle file.

4.2 Changing Audio of a Chunk

When any change in a chunk is requested by the client, the newly uploaded audio is processed, and needful changes are made to its characteristics such as bit rate, sample rate, length, encoding, etc. Finally, the old chunk is then replaced with the new chunk.

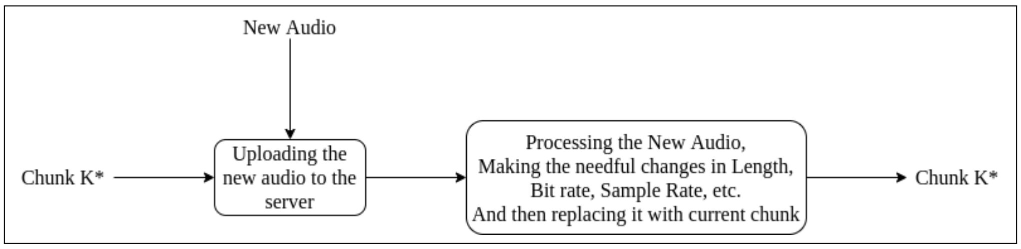

New Audio
Chunk K*
Uploading the new audio to the server
Processing the New Audio, Making the needful changes in Length, Bit rate, Sample Rate, etc. And then replacing it with current chunk
Chunk K*

Figure 4.2: Changing Audio of a Chunk

4.3 Compiling Chunks

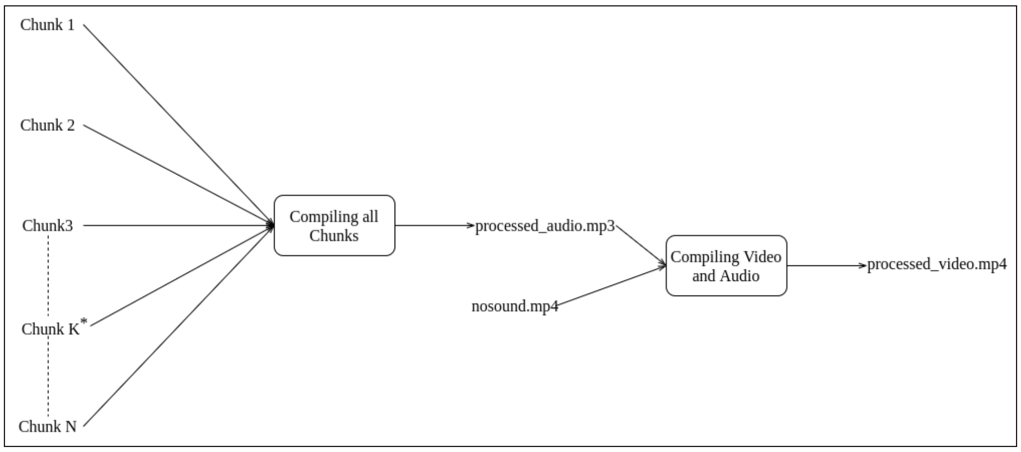


Figure 4.3: Compiling Chunks

At this stage, all the chunks are concatenated in a particular order. Then the newly processed audio stream is compiled with the video stream, and the newly processed video is created.

Chapter 5

DevOps

5.1 Docker

Docker[7, 8] is a set of platform as a service (PaaS) products that uses OS-level virtualisation to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating system kernel and therefore use fewer resources than virtual machines. In this project: Django, Celery and Redis were containerised. They were managed with the help of Docker Compose[9].

5.2 Travis CI

Travis CI[10] is a hosted continuous integration service used to build and test software projects hosted on GitHub. Travis CI was used to check linting in every commit. The linter used is Flake8[11]

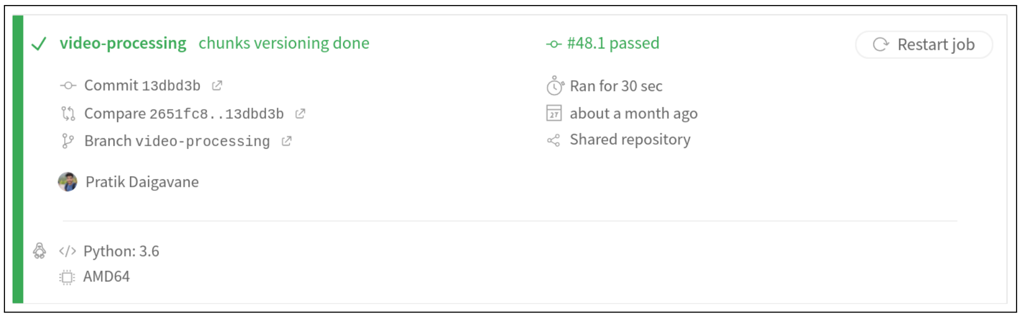
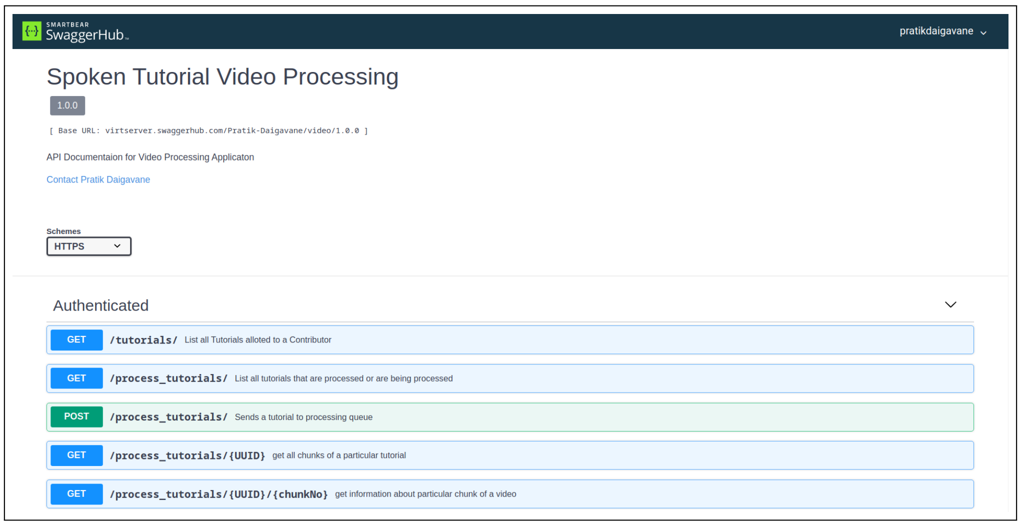

video-processing chunks versioning done
#48.1 passed
Commit 13dbd3b
Compare 2651fc8..13dbd3b
Branch video-processing
Ran for 30 sec
about a month ago
Shared repository
Pratik Daigavane
Python: 3.6
AMD64

Figure 5.1: Travis CI Job

5.3 OpenAPI Compliant API Documentation

Swagger[12] has been used to generate OpenAPI compatible schema and also documenting the endpoints. This also provides various API endpoints which are easy to try out, from the browser itself.



Spoken Tutorial Video Processing

1.0.0

[Base URL: virtserver.swaggerhub.com/Pratik-Daigavane/video/1.0.0]

API Documentaion for Video Processing Applicaton

Contact Pratik Daigavane

Schemes: HTTPS

Authenticated

GET `/tutorials/` List all Tutorials alloted to a Contributor

GET `/process_tutorials/` List all tutorials that are processed or are being processed

POST `/process_tutorials/` Sends a tutorial to processing queue

GET `/process_tutorials/{UUID}` get all chunks of a particular tutorial

GET `/process_tutorials/{UUID}/{chunkNo}` get information about particular chunk of a video

Figure 5.2: API Documentation Screenshot

Chapter 6

Frontend

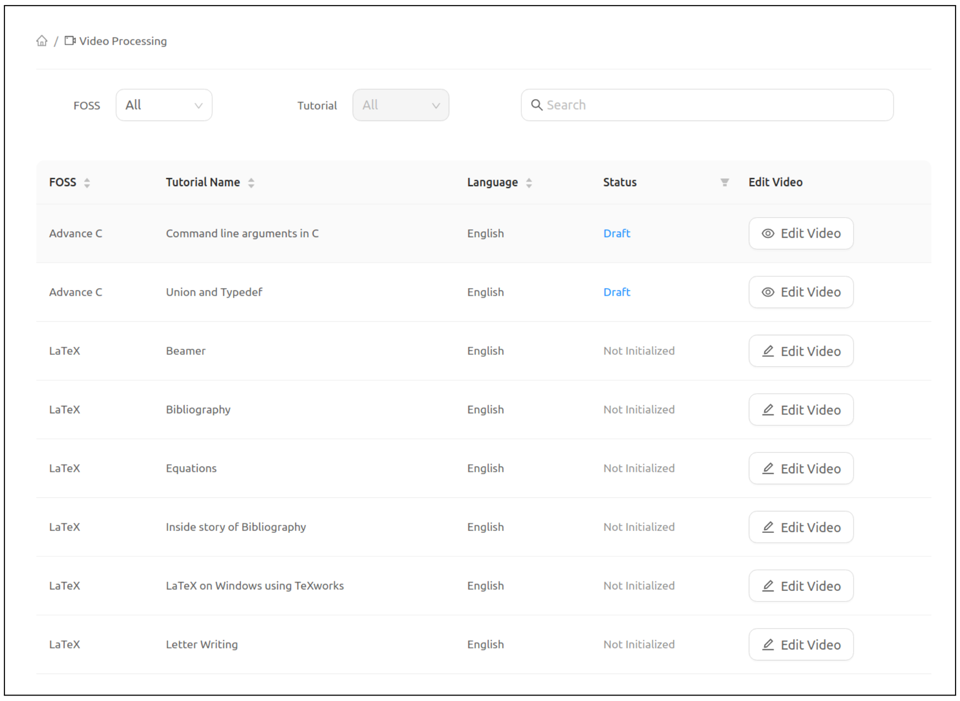
This project has two user interfaces.

- Contributor UI
- Reviewer UI

6.1 Contributor User Interface

This user interface will be used by a contributor to edit a video and then submit it for review.

6.1.1 Tutorials List

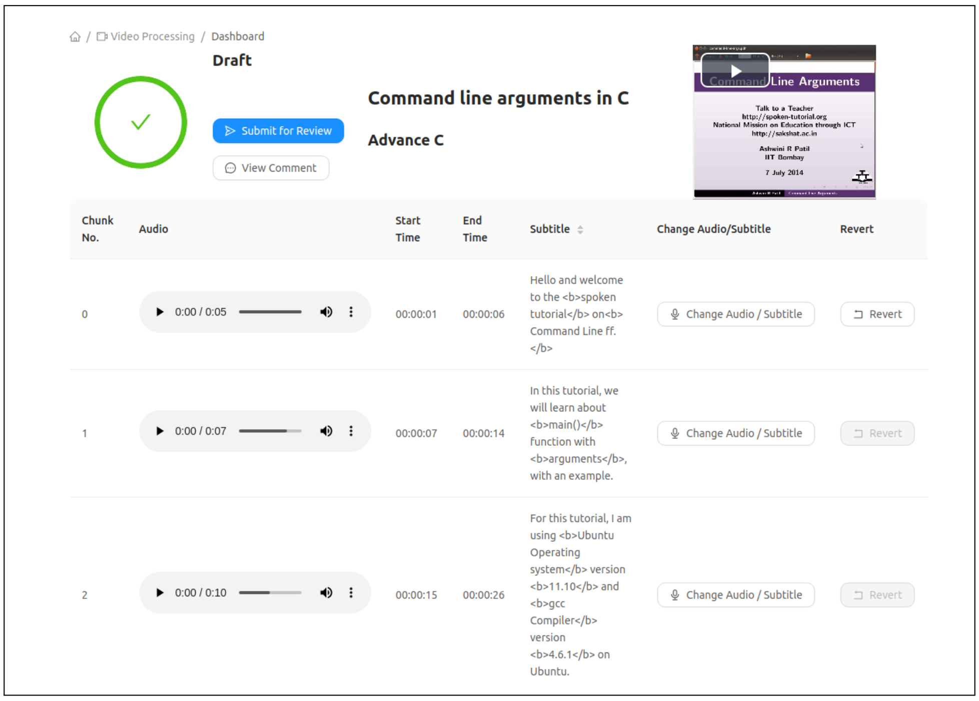


FOSS	Tutorial Name	Language	Status	Edit Video
Advance C	Command line arguments in C	English	Draft	Edit Video
Advance C	Union and Typedef	English	Draft	Edit Video
LaTeX	Beamer	English	Not Initialized	Edit Video
LaTeX	Bibliography	English	Not Initialized	Edit Video
LaTeX	Equations	English	Not Initialized	Edit Video
LaTeX	Inside story of Bibliography	English	Not Initialized	Edit Video
LaTeX	LaTeX on Windows using TeXworks	English	Not Initialized	Edit Video
LaTeX	Letter Writing	English	Not Initialized	Edit Video

Figure 6.1: Tutorials List

Tutorials List page will enlist in a tabular manner: all the tutorials allotted to a contributor. The table will have FOSS, Tutorial name, Language, Status and Edit video button. After clicking on edit video button of a particular tutorial, the contributor will be directed to a dashboard page from where he/she will be able to edit the video.

6.1.2 Dashboard



Draft

Command line arguments in C

Advance C

Chunk No.	Audio	Start Time	End Time	Subtitle	Change Audio/Subtitle	Revert
0	0:00 / 0:05	00:00:01	00:00:06	Hello and welcome to the <b>spoken tutorial</b> on<b> Command Line ff. </b>	Change Audio / Subtitle	Revert
1	0:00 / 0:07	00:00:07	00:00:14	In this tutorial, we will learn about <b>main()</b> function with <b>arguments</b>, with an example.	Change Audio / Subtitle	Revert
2	0:00 / 0:10	00:00:15	00:00:26	For this tutorial, I am using <b>Ubuntu Operating system</b> version <b>11.10</b> and <b>gcc Compiler</b> version <b>4.6.1</b> on Ubuntu.	Change Audio / Subtitle	Revert

Figure 6.2: Video editing dashboard

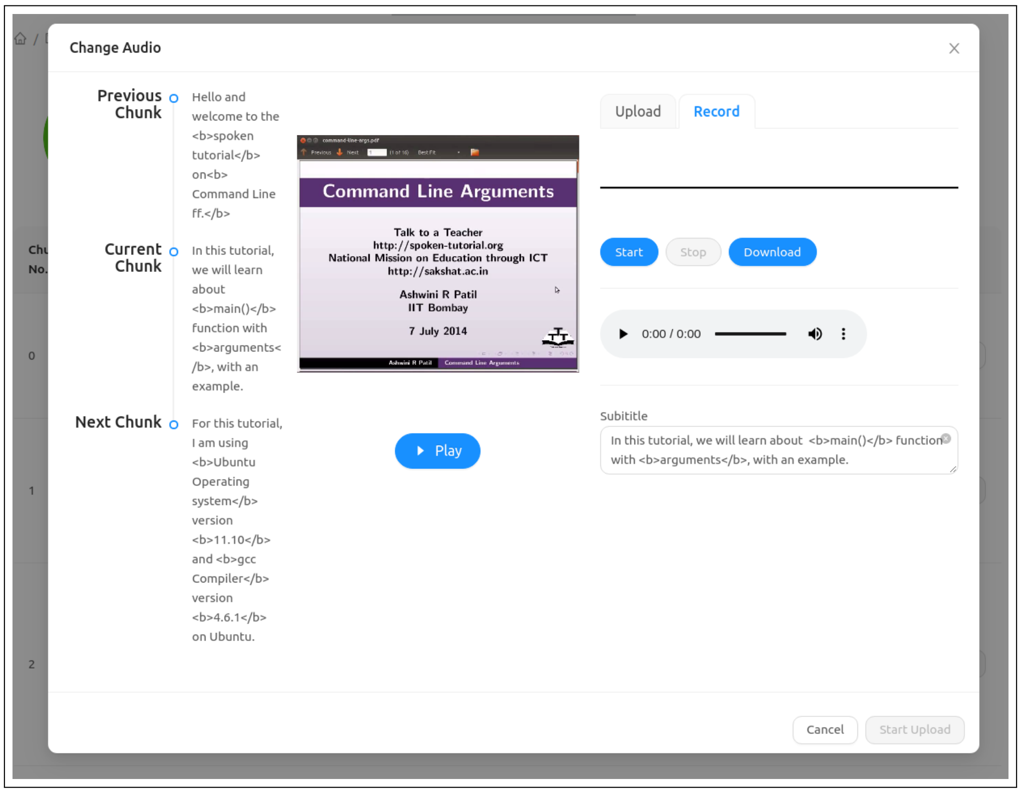
The dashboard page will have all the chunks of tutorial listed in a tabular manner. Each row in the table has Chunk No, Audio, Subtitle, Change Audio and Revert button. The status, along with the preview of the video, is present in the top section. For each chunk, the contributor will be able to edit its audio and subtitle. After each edit, the previous versions are maintained; and if needed, the contributor can revert to the previous version.

Change Audio/Subtitle

When clicked on Change Audio/Subtitle button, a modal will appear as shown in Figure 6.3. Here, the contributor can edit the subtitle and change the audio. The changing of audio can be done in two ways: either by uploading a new one or by recording at that instant by using the record feature.

On the left side, a timeline will be shown, where the contributor can preview the previous chunk, current chunk and next chunk in sync to get an idea about the flow of the video.

After the needful changes are made, the contributor has to click on the Upload Button to confirm changes. After clicking the button, the changes will be processed by the server, and the video preview will be updated with new changes.



Hello and welcome to the <b>spoken tutorial</b> on<b> Command Line ff.</b>

In this tutorial, we will learn about <b>main()</b> function with <b>arguments</b>, with an example.

For this tutorial, I am using <b>Ubuntu Operating system</b> version <b>11.10</b> and <b>gcc Compiler</b> version <b>4.6.1</b> on Ubuntu.

In this tutorial, we will learn about <b>main()</b> function with <b>arguments</b>, with an example.

Figure 6.3: Change Audio Dialog Box

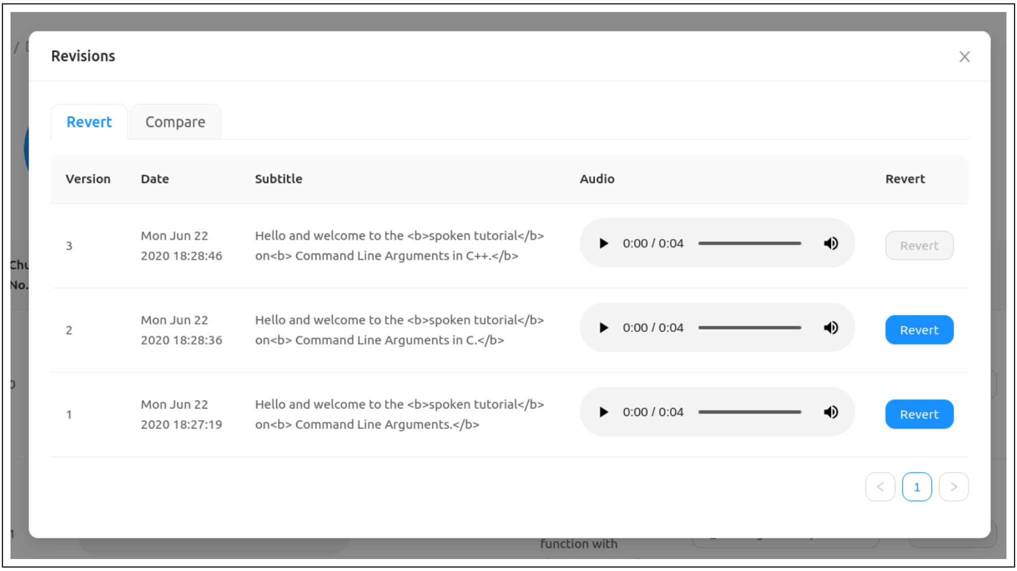
Revisions

When clicked on the Revert button, a modal will appear as shown in Figure 6.4. All the revisions will be listed, and the contributor will have an option to revert to any version of that chunk.

The contributor can also compare any two versions of a chunk using the Compare tab shown in Figure 6.5. This feature will highlight the differences between the two versions. The implementation of text differencing is based on the algorithm proposed in "An O(ND) Difference Algorithm and its Variations" (Myers, 1986)[13].

Submitting for Review

The contributor can submit the tutorial for review after making all the needful changes in the tutorial. The tutorial will then be reviewed by a reviewer, and the verdict will be communicated back to the contributor.

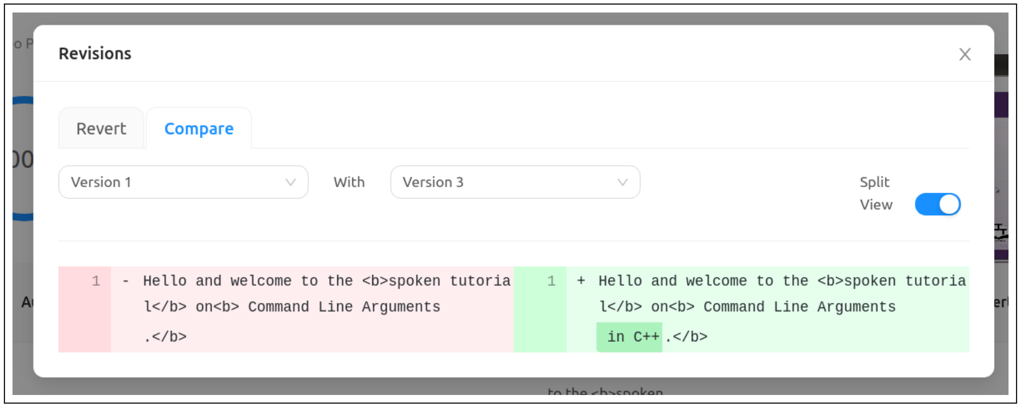


Revisions

Revert | Compare

Version	Date	Subtitle	Audio	Revert
3	Mon Jun 22 2020 18:28:46	Hello and welcome to the <b>spoken tutorial</b> on<b> Command Line Arguments in C++.</b>	0:00 / 0:04	Revert
2	Mon Jun 22 2020 18:28:36	Hello and welcome to the <b>spoken tutorial</b> on<b> Command Line Arguments in C.</b>	0:00 / 0:04	Revert
1	Mon Jun 22 2020 18:27:19	Hello and welcome to the <b>spoken tutorial</b> on<b> Command Line Arguments.</b>	0:00 / 0:04	Revert

Figure 6.4: Revisions of a chunk



Revisions

Revert | Compare

Version 1 With Version 3 Split View

```
1  - Hello and welcome to the <b>spoken tutoria     1  + Hello and welcome to the <b>spoken tutoria
     l</b> on<b> Command Line Arguments                  l</b> on<b> Command Line Arguments
     .</b>                                               in C++.</b>
```

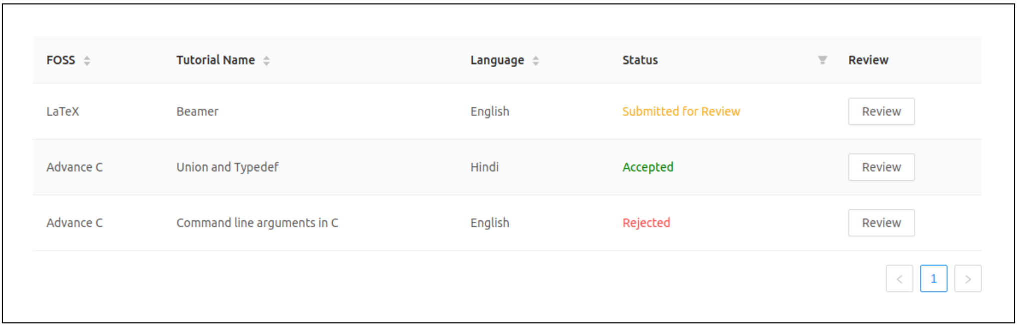
Figure 6.5: Comparing Two Versions

6.2 Reviewer User Interface

This user interface will be used by the reviewer to review tutorials and communicate their verdicts to the contributor.

6.2.1 Tutorials List

In this page, all tutorials to be reviewed are listed along with their status such as Submitted for Review, Accepted, Rejected. After clicking on the Review button of a particular tutorial, the reviewer will be directed to a dashboard page from where the tutorial can be reviewed.

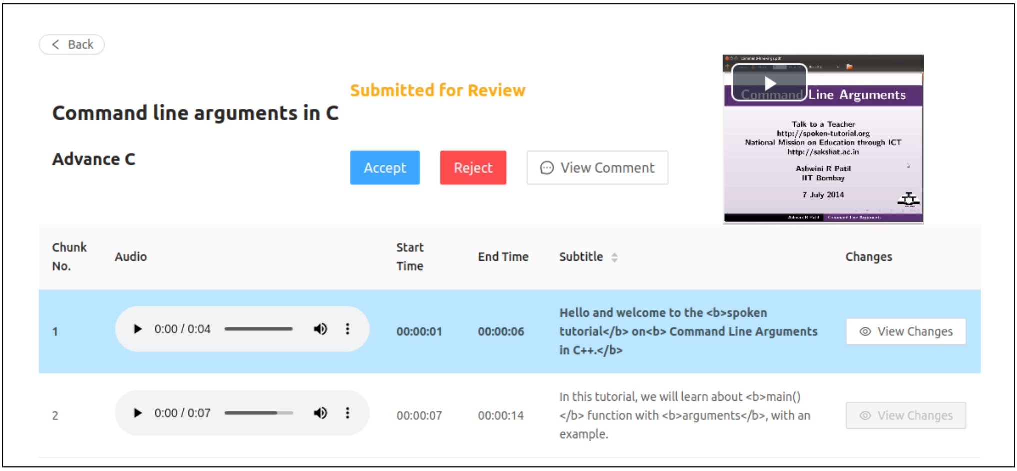


FOSS	Tutorial Name	Language	Status	Review
LaTeX	Beamer	English	Submitted for Review	Review
Advance C	Union and Typedef	Hindi	Accepted	Review
Advance C	Command line arguments in C	English	Rejected	Review

Figure 6.6: Tutorials List for Reviewer

6.2.2 Dashboard

The dashboard lists all the chunks of a tutorial and the chunks which have changed are highlighted. For the chunk which is changed, the reviewer can compare the old and new versions proposed by the contributor.



< Back

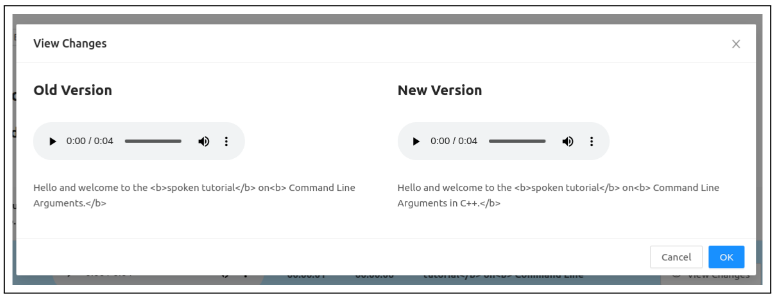
Command line arguments in C

Advance C

Submitted for Review

Chunk No.	Audio	Start Time	End Time	Subtitle	Changes
1	0:00 / 0:04	00:00:01	00:00:06	Hello and welcome to the <b>spoken tutorial</b> on<b> Command Line Arguments in C++.</b>	View Changes
2	0:00 / 0:07	00:00:07	00:00:14	In this tutorial, we will learn about <b>main()</b> function with <b>arguments</b>, with an example.	View Changes

Figure 6.7: Reviewer Dashboard



View Changes

Old Version

0:00 / 0:04

Hello and welcome to the <b>spoken tutorial</b> on<b> Command Line Arguments.</b>

New Version

0:00 / 0:04

Hello and welcome to the <b>spoken tutorial</b> on<b> Command Line Arguments in C++.</b>

Figure 6.8: Comparing the new and the old version

In the top section, the reviewer can preview the tutorial with all the proposed changes. An option to either Accept or Reject the video is provided. When clicked on Accept, the changes implemented by the contributor will be approved, and when clicked on Reject, the reviewer can reject the video along with the reason for rejection. After rejecting, the contributor can view the reason for rejection and then make appropriate changes, after which the tutorial can be resubmitted for review, following the same cycle.

Chapter 7

API Endpoints

In the following API documentation, all API Endpoints are listed.

Spoken Tutorial Video Processing

Overview

API Documentaion for Video Processing Applicaton

Version information

Version : 1.0.0

URI scheme

Host : localhost
BasePath : /videoprocessing/api/
Schemes : HTTPS, HTTP

Paths

Sends a tutorial to processing queue

```
POST /process_tutorials
```

Description

Sends a tutorial particular to processing queue

Parameters

Type	Name	Description	Schema
Body	**tutorial details** *optional*	Tutorial Id and Language ID	VideoSubmissionCreate

Responses

HTTP Code	Description	Schema
202	Request Accepted	VideoTutorialSubmission

HTTP Code	Description	Schema
400	invalid input, object invalid	No Content
403	forbidden	No Content

Consumes

- `application/json`

Produces

- `application/json`

Tags

- Contributor Endpoints

List all tutorials that are processed or are being processed

```
GET /process_tutorials
```

Description

This endpoint will return all the tutorials that are processed or are being processed

Responses

HTTP Code	Description	Schema
200	result listing all the processed tutorials	< VideoTutorialSubmission > array
403	forbidden	No Content
500	Internal Server Error	No Content

Produces

- `application/json`

Tags

- Contributor Endpoints

Submit tutorial for review

```
POST /process_tutorials/submit
```

Description

Submit tutorial for review

Parameters

Type	Name	Description	Schema
Body	**comment** *optional*	Message for the reviewer	string

Responses

HTTP Code	Description	Schema
200	Tutorial sent for review	VideoTutorialSubmission
400	bad request	No Content
409	request already submitted	No Content

Consumes

- `application/json`

Produces

- `application/json`

Tags

- Contributor Endpoints

get all chunks of a particular tutorial

```
GET /process_tutorials/{UUID}
```

Description

This endpoint return will all the chunks created for a particular video tutorial

Parameters

Type	Name	Schema
Path	**UUID** *required*	string (uuid)

Responses

HTTP Code	Description	Schema
200	result listing all the chunks	inline_response_200
400	Bad Request	No Content
403	forbidden	No Content
500	Internal Server Error	No Content

Produces

- `application/json`

Tags

- Contributor Endpoints

get information about particular chunk of a video

```
GET /process_tutorials/{UUID}/{chunkNo}
```

Description

This endpoint return information about a particular chunk of a video

4

Parameters

Type	Name	Schema
Path	**UUID** *required*	string (uuid)
Path	**chunkNo** *required*	number

Responses

HTTP Code	Description	Schema
200	result listing information of a particular chunk	inline_response_200_1
400	Bad Request	No Content
403	forbidden	No Content
500	Internal Server Error	No Content

Produces

- `application/json`

Tags

- Contributor Endpoints

change audio of a particular chunk

```
PATCH /process_tutorials/{UUID}/{chunkNo}
```

Parameters

Type	Name	Description	Schema
Path	**UUID** *required*		string (uuid)
Path	**chunkNo** *required*		number

Type	Name	Description	Schema
Body	**Audio File** *optional*	Upload new audio	Audio File

Responses

HTTP Code	Description	Schema
200	result listing information of a particular chunk	inline_response_200_1
400	bad request	No Content
404	not found	No Content

Produces

- `application/json`

Tags

- Contributor Endpoints

Revert chunk back to previous version

```
PATCH /process_tutorials/{UUID}/{chunkNo}/{historyId}
```

Description

This endpoint revert chunk back to previous version

Parameters

Type	Name	Schema
Path	**UUID** *required*	string (uuid)
Path	**chunkNo** *required*	number
Path	**historyId** *required*	number

Responses

HTTP Code	Description	Schema
200	result listing information of a particular chunk	inline_response_200_1
400	Chunk already up to date	No Content
404	not found	No Content

Produces

- `application/json`

Tags

- Contributor Endpoints

List all Tutorials which are to be reviewed

```
GET /review
```

Description

This endpoint will return all the tutorials which are to be reviewed

Responses

HTTP Code	Description	Schema
200	result listing all the tutorials	< Tutorial > array
400	bad request	No Content
403	forbidden	No Content

Produces

- `application/json`

Tags

- Reviewer Endpoints

get all chunks of a particular tutorial

```
GET /review/{UUID}
```

Description

This endpoint return will all the chunks created for a particular video tutorial

Parameters

Type	Name	Schema
Path	**UUID** *required*	string (uuid)

Responses

HTTP Code	Description	Schema
200	result listing all the chunks	inline_response_200

Produces

- `application/json`

Tags

- Reviewer Endpoints

Set Verdict of a particular tutorial

```
POST /review/{UUID}/verdict
```

Description

This endpoint return will set verdict of a particular tutorial

Parameters

Type	Name	Description	Schema
Path	**UUID** *required*		string (uuid)
Body	**Verdict** *optional*	Verdict of Video	Verdict

Responses

HTTP Code	Description	Schema
200	result listing all the chunks	inline_response_200
400	Bad Request	No Content
403	forbidden	No Content
500	Internal Server Error	No Content

Produces

- `application/json`

Tags

- Reviewer Endpoints

get information about a particular chunk of a video

```
GET /review/{UUID}/{chunkNo}
```

Description

This endpoint return information about particular chunk of a video

Parameters

Type	Name	Schema
Path	**UUID** *required*	string (uuid)

Type	Name	Schema
Path	**chunkNo** *required*	number

Responses

HTTP Code	Description	Schema
200	result listing information of a particular chunk	inline_response_200_1
400	Bad Request	No Content
403	forbidden	No Content
500	Internal Server Error	No Content

Produces

- `application/json`

Tags

- Reviewer Endpoints

List all Tutorials allotted to a Contributor

```
GET /tutorials
```

Description

This endpoint will return all the tutorials uploaded earlier on the server

Responses

HTTP Code	Description	Schema
200	result listing all the tutorials	< Tutorial > array
403	forbidden	No Content
500	Internal Server Error	No Content

Produces

- `application/json`

Tags

- Contributor Endpoints

Definitions

Audio File

Name	Description	Schema
audio_file *optional*	**Example** : `"new_audio.mp3"`	string (binary)
subtitle *optional*	**Example** : `"this is new subititle"`	string

Tutorial

Name	Schema
foss_category *required*	Tutorial_foss_category
language *required*	Tutorial_language
tutorial_detail *required*	Tutorial_tutorial_detail

Tutorial_foss_category

Name	Description	Schema
description *optional*	**Example** : `"Advanced C is for the programmer who has some experience writing applications in C."`	string
id *optional*	**Example** : `61`	integer

Name	Description	Schema
name *optional*	**Example** : `"Advance C"`	string

Tutorial_language

Name	Description	Schema
id *optional*	**Example** : `22`	integer
name *optional*	**Example** : `"English"`	string

Tutorial_tutorial_detail

Name	Description	Schema
id *optional*	**Example** : `605`	integer
tutorial *optional*	**Example** : `"Command line arguments in C"`	string

Verdict

Name	Description	Schema
comment *optional*	**Example** : `"Audio not clear"`	string
verdict *optional*	**Example** : `"accepted"`	string

VideoChunk

Name	Description	Schema
audio_chunk *required*	**Example** : `"http://127.0.0.1:8000/media/videoprocessing/2e58b1fc-c501-46c7-9ca5-2031409cc5a8/chunks/5.mp3"`	string (uri)

Name	Description	Schema
chunk_no *required*	**Example** : `5.0`	number
end_time *required*	**Example** : `"00:00:11.350000"`	string (time)
revisions *optional*	**Example** : `2`	integer
start_time *required*	**Example** : `"00:00:09.350000"`	string (time)
subtitle *required*	**Example** : `"this is a sample subtitle"`	string
video_chunk *required*	**Example** : `"http://127.0.0.1:8000/media/videoprocessing/2e58b1fc-c501-46c7-9ca5-2031409cc5a8/chunks/5.mp4"`	string (uri)

VideoChunkRevisions

Name	Description	Schema
audio_chunk *required*	**Example** : `"http://127.0.0.1:8000/media/videoprocessing/2e58b1fc-c501-46c7-9ca5-2031409cc5a8/chunks/5.mp3"`	string (uri)
history_data *optional*	**Example** : `"2020-06-23T08:05:31.602674+05:30"`	string (date-time)
history_id *optional*	**Example** : `273.0`	number
subtitle *optional*	**Example** : `"this is a sample subtitle"`	string

VideoSubmissionCreate

Name	Description	Schema
language *required*	**Example** : `22`	integer

Name	Description	Schema
tutorial_detail *required*	**Example** : `605`	integer

VideoTutorialSubmission

Name	Description	Schema
checksum *optional*	**Example** : `"44e437f0e54c0d839e44109b9544ecf1"`	string
comment *optional*	**Example** : `"Chunk 1 is modified"`	string (string)
id *required*	**Example** : `"2e58b1fc-c501-46c7-9ca5-2031409cc5a8"`	string (uuid)
language *optional*	**Example** : `22`	integer
processed_video *optional*	**Example** : `"http://127.0.0.1:8000/media/videoprocessing/2e58b1fc-c501-46c7-9ca5-2031409cc5a8/processed_video.mp4"`	string (uri)
status *optional*	**Example** : `"done"`	string (string)
submission_status *optional*	**Example** : `"accepted"`	string (string)
subtitle *required*	**Example** : `"http://127.0.0.1:8000/media/videoprocessing/2e58b1fc-c501-46c7-9ca5-2031409cc5a8/subtitle.srt"`	string (uri)
total_chunks *optional*	**Example** : `30.0`	number
tutorial_detail *optional*	**Example** : `605`	integer
video *required*	**Example** : `"http://127.0.0.1:8000/media/videoprocessing/2e58b1fc-c501-46c7-9ca5-2031409cc5a8/video.mp4"`	string (uri)

inline_response_200

Name	Schema
chunks *optional*	< VideoChunk > array
video_data *optional*	VideoTutorialSubmission

inline_response_200_1

Name	Schema
VideoChunk *optional*	VideoChunk
history *optional*	< VideoChunkRevisions > array

Bibliography

[1] https://reactjs.org/ (Last accessed 30 May 2020).

[2] https://www.djangoproject.com/ (Last accessed 30 May 2020).

[3] https://www.django-rest-framework.org/ (Last accessed 30 May 2020).

[4] https://www.celeryproject.org/ (Last accessed 30 May 2020).

[5] https://redis.io/ (Last accessed 30 May 2020).

[6] https://www.ffmpeg.org/ (Last accessed 30 May 2020).

[7] https://www.docker.com/ (Last accessed 30 May 2020).

[8] https://en.wikipedia.org/wiki/Docker_(software) (Last accessed 30 May 2020).

[9] https://docs.docker.com/compose/ (Last accessed 30 May 2020).

[10] https://travis-ci.org/ (Last accessed 30 May 2020).

[11] https://flake8.pycqa.org/en/latest/ (Last accessed 30 May 2020).

[12] hhttps://swagger.io/ (Last accessed 31 May 2020).

[13] E. W. Myers, "An o(nd) difference algorithm and its variations," *Algorithmica*, vol. 1, pp. 251–266, 1986.