Summer Fellowship Report

On

Script Creation System for Spoken Tutorials

Submitted by

Selvamani Kannan
Nishant Ranjan
Manan Poddar
Shubham Garg

Under the guidance of

Prof.Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

July 25, 2019

Acknowledgment

We, the summer interns of the **fossee - Script Creation Process for Spoken Tutorials** are 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. We wholeheartedly thanks **Prof. Kannan M. Moudgalya** for having faith in us, selecting us to be a part of his valuable project and for constantly motivating us to do better. We thanks **Mr. Nagesh Karmali** for providing us the opportunity to work on this project. We are also very thankful to our mentors 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, we got a chance to see the stronger side of our technical and nontechnical aspects and also strengthen our concepts. Last but not the least, we sincerely thank all our other colleagues working in different projects under **Prof. Kannan M. Moudgalya** for helping us evolve better with their critical advice.

Declaration

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

We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We 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 properly cited or from whom proper permission has not been taken when needed.

Selvamani Kannan
Nishant Ranjan
Manan Poddar
Shubham Garg

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Statement

Development of a platform for the Spoken-tutorial.org , for automating the process of script creation for the tutorials. In the current system, the manual process is followed. The workflow of the system is described as follows. The script is divided into the individual scene having a 2-column format containing Visual Cue and Narration. In each scene, Visual Cue is written as a series of events leading to actions, and Narration refers to the textual voice-over given by the speaker for its corresponding events in the visual cue

1.2 Project Objective

The objective of the project is to add modules to the existing website name **Script-manager** so that one can create the scripts on the website directly or upload a file containing the scripts and can modify them afterwards. 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 particular cue-narration pair regarding doubts/questions/suggestions on that script

1.3 Project Outcome

The creator will be able to create the script. The script will comprise of slides, each divided into 2 columns. There would be a facility to add, remove, and reorder the slides. Once the creator creates the script, he/she can send it for review. The existing publishing workflow will be followed in the review process.

1.4 Project Requirements

We are using the following tools during development

- Django (v1.11) - used for developing the back-end of the platform.
- Angular 7 (v7.0) - used for front-end development.
- MySQL - the database used for storing the data.

Chapter 2

Project Overview

In the current system, for creating Scripts and get it reviewed, the users are allowed to use services like Google Docs and Gmail only, which is pretty cumbersome and inefficient. We have replaced them by developing a platform which essentially provides features similar to Google Docs and also has facilities to get it reviewed without mailing, by simply automating the process.

2.1 Django for backend & Angular 7 for frontend

Since the existing system already uses **Django** as a server-side framework and **MySQL** as the database, we also used the same for maintaining compatibility with the existing system. Although we have used Angular to develop the frontend using Django templates (which is used in the existing system) would imply to write all the **javascript** on our own which isn't required when these frameworks are already present. With the help of them, It becomes really easy to develop and maintain the code.
Since the existing system is using Django platform as a server-side and **MySQL** as a database, so we use the same for compatibility with the existing system. Add we use Angular in this to develop Front-end using Django Templets. These templates make our work easy to develop and maintain the code

Chapter 3

Front-end

3.1 Authentication Using JWT tokens

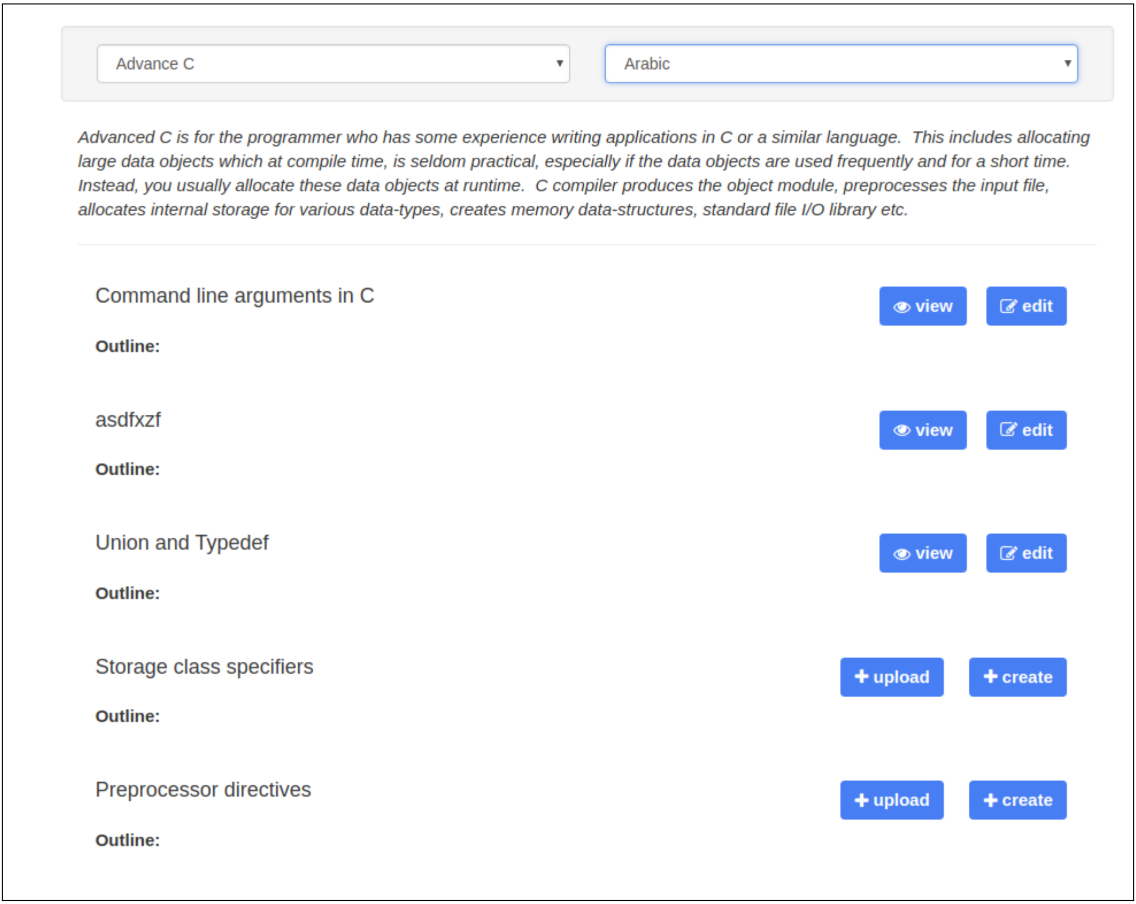
The existing system handles the login and registration part. To use our system, the user should be already logged in as we need the details of foss category and languages being assigned to him. Therefore, we have used **JWT** tokens to authenticate the user with the application server by sending this token with each API calls we make. We make an API call as soon as the user comes to our system to get this token from the application server and we are storing it in the local storage, So after that, every API call goes with this token and that's how the authentication is happening. After 3 seconds the token automatically gets refreshed to ensure the security of the system.

3.2 The Home UI

The Home UI is a dashboard for displaying the tutorials list based on foss category and language selected by the user.

Only the foss category and language which has been assigned to the particular user is shown in the drop-down menu. After selecting them, all the tutorials of that particular category is shown.

User can now either create scripts for a particular tutorial or edit/view a previously created scripts for a tutorial.



Advanced C is for the programmer who has some experience writing applications in C or a similar language. This includes allocating large data objects which at compile time, is seldom practical, especially if the data objects are used frequently and for a short time. Instead, you usually allocate these data objects at runtime. C compiler produces the object module, preprocesses the input file, allocates internal storage for various data-types, creates memory data-structures, standard file I/O library etc.

Command line arguments in C

Outline:

asdfxzf

Outline:

Union and Typedef

Outline:

Storage class specifiers

Outline:

Preprocessor directives

Outline:

Figure 3.1: Script Dashboard

For Creating Scripts, three features are provided to the users :

- Direct Entry in the system.
- Upload docs/odt files.
- Copy and paste the whole table.

3.2.1 Direct Entry in the system

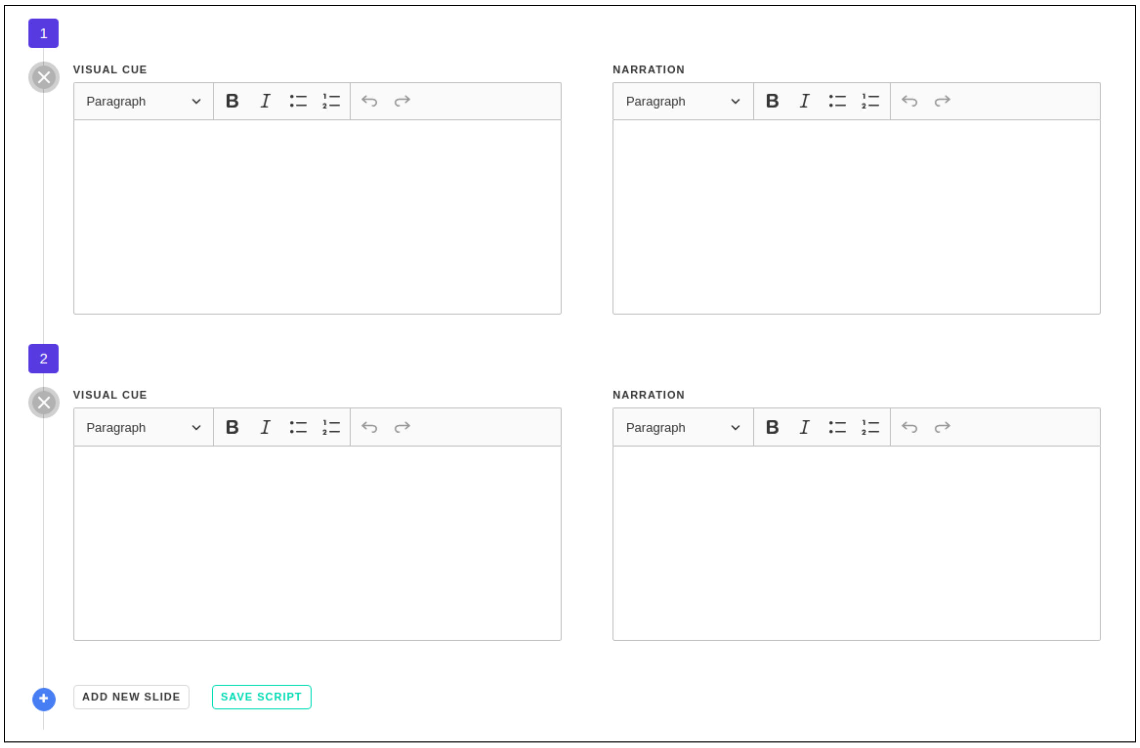
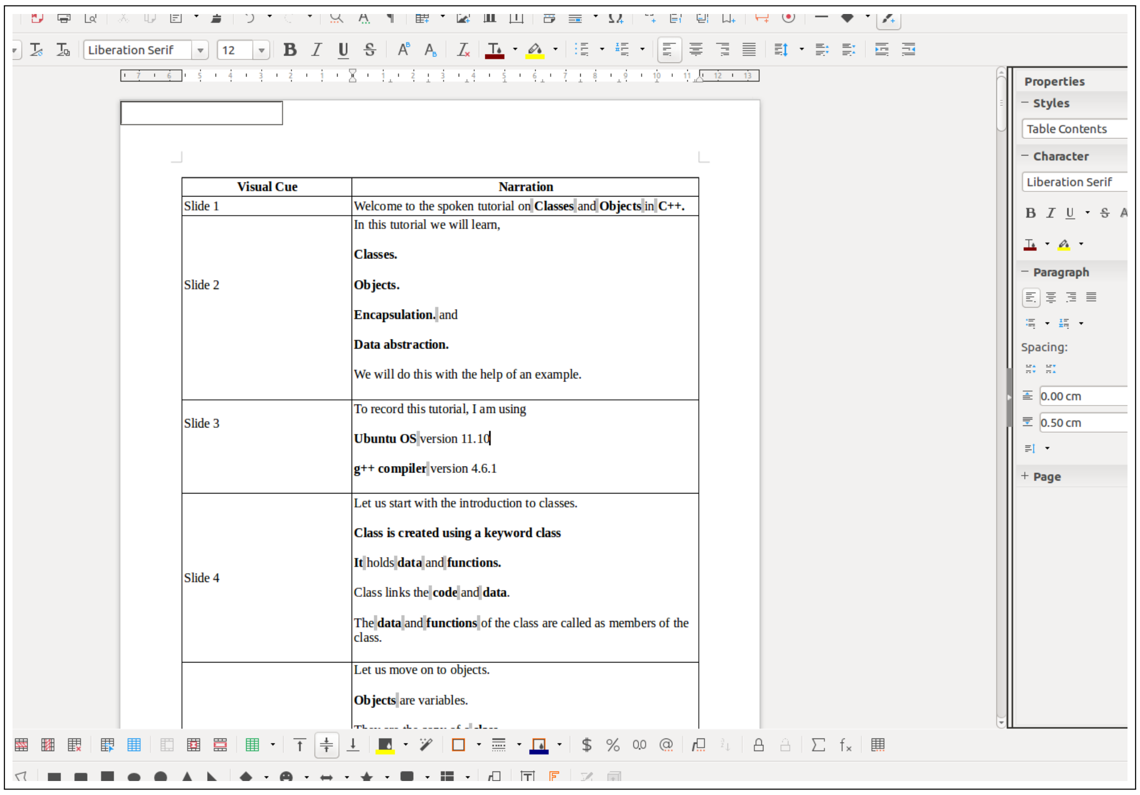


Figure 3.2: Space for writing scripts

Here user can directly enter visual cue and narration for each slide and as each tutorial contains multiple slides, therefore, the system provides the facility to the user, to add and delete the rows dynamically and save it after all the changes made.

3.2.2 Upload File

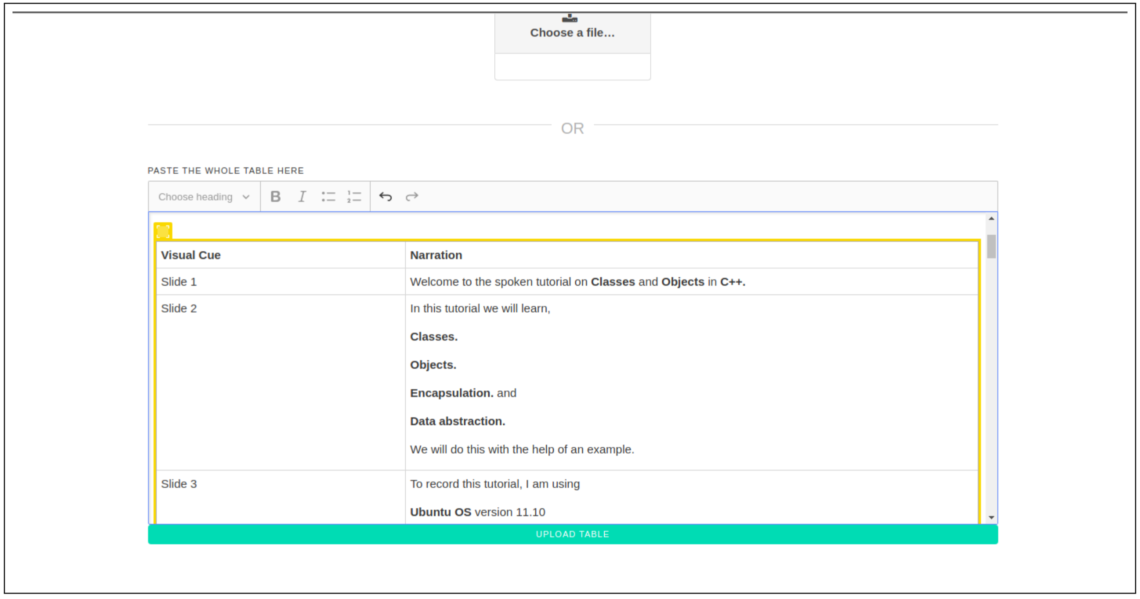
User can **Upload** the whole doc/odt files containing tables which in turn contains data of visual cue and narration. In this way, user can work "offline" as well.



Visual Cue	Narration
Slide 1	Welcome to the spoken tutorial on **Classes** and **Objects** in **C++.**
Slide 2	In this tutorial we will learn, **Classes.** **Objects.** **Encapsulation.** and **Data abstraction.** We will do this with the help of an example.
Slide 3	To record this tutorial, I am using **Ubuntu OS** version 11.10 **g++ compiler** version 4.6.1
Slide 4	Let us start with the introduction to classes. **Class is created using a keyword class** **It** holds **data** and **functions.** Class links the **code** and **data**. The **data** and **functions** of the class are called as members of the class.
	Let us move on to objects. **Objects** are variables.

Figure 3.3: Table of script in doc/odt

3.2.3 User can copy and paste the table directly on the system



Visual Cue	Narration
Slide 1	Welcome to the spoken tutorial on **Classes** and **Objects** in **C++.**
Slide 2	In this tutorial we will learn, **Classes.** **Objects.** **Encapsulation.** and **Data abstraction.** We will do this with the help of an example.
Slide 3	To record this tutorial, I am using **Ubuntu OS** version 11.10

Figure 3.4: Upload file or paste content

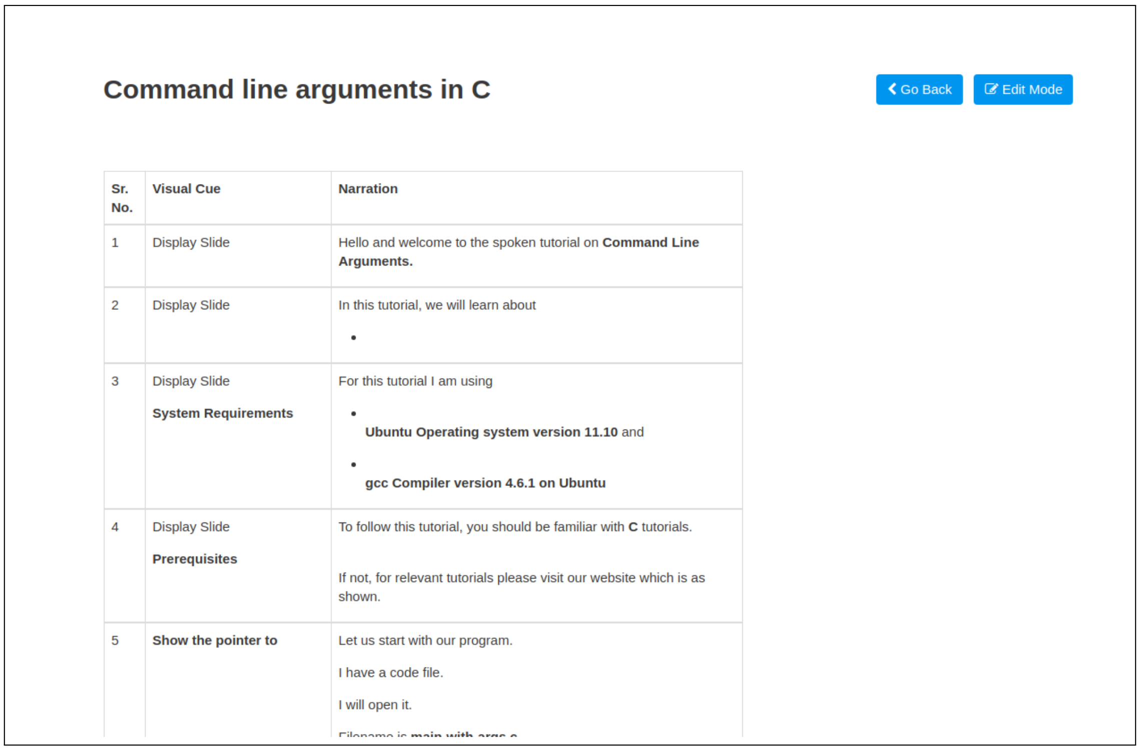
Feature 3.3.2 and 3.3.3 provide user, the facility to work offline as well.

3.3 View UI

After creating the scripts manually or uploading them directly to the system, user will be redirected to the view page. There are three main components on this page:

- Script Details Table
- Comments
- Revisions

3.3.1 Scripts Details Table



Command line arguments in C

Sr. No.	Visual Cue	Narration
1	Display Slide	Hello and welcome to the spoken tutorial on **Command Line Arguments.**
2	Display Slide	In this tutorial, we will learn about •
3	Display Slide **System Requirements**	For this tutorial I am using • **Ubuntu Operating system version 11.10** and • **gcc Compiler version 4.6.1 on Ubuntu**
4	Display Slide **Prerequisites**	To follow this tutorial, you should be familiar with **C** tutorials. If not, for relevant tutorials please visit our website which is as shown.
5	**Show the pointer to**	Let us start with our program. I have a code file. I will open it.

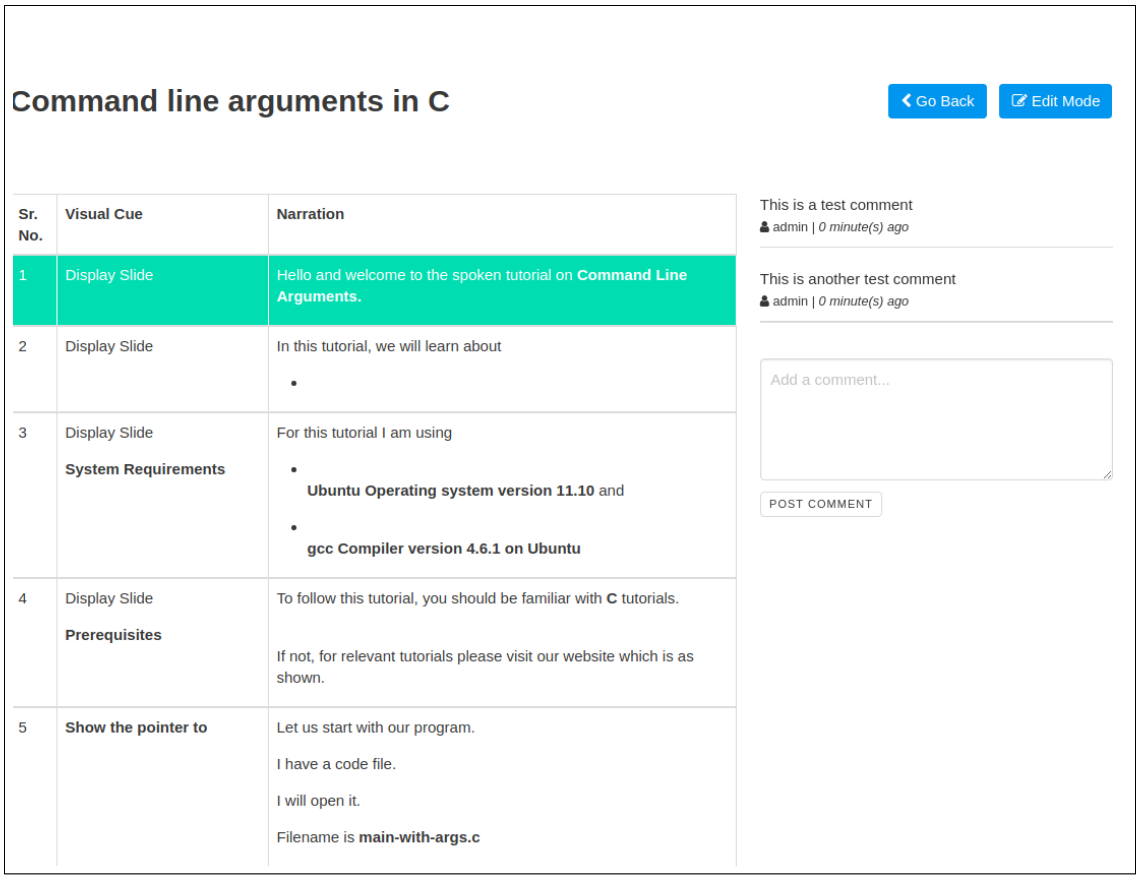
Figure 3.5: Table of script

This table occupies two-thirds of the page and contains three columns Serial No, Visual Cue and Narration.

- If the script is created manually on the system then data from each slide (i.e., each cue narration pair) will be inserted into a table row
- If the script is uploaded as a file or copy-pasted onto the system then the data will be parsed from the file and then inserted into the table row.

When hovered over the particular row of the table, it will show two buttons Comment and Revision

3.3.2 Comments



Command line arguments in C

Sr. No.	Visual Cue	Narration
1	Display Slide	Hello and welcome to the spoken tutorial on **Command Line Arguments.**
2	Display Slide	In this tutorial, we will learn about •
3	Display Slide **System Requirements**	For this tutorial I am using • **Ubuntu Operating system version 11.10** and • **gcc Compiler version 4.6.1 on Ubuntu**
4	Display Slide **Prerequisites**	To follow this tutorial, you should be familiar with **C** tutorials. If not, for relevant tutorials please visit our website which is as shown.
5	**Show the pointer to**	Let us start with our program. I have a code file. I will open it. Filename is **main-with-args.c**

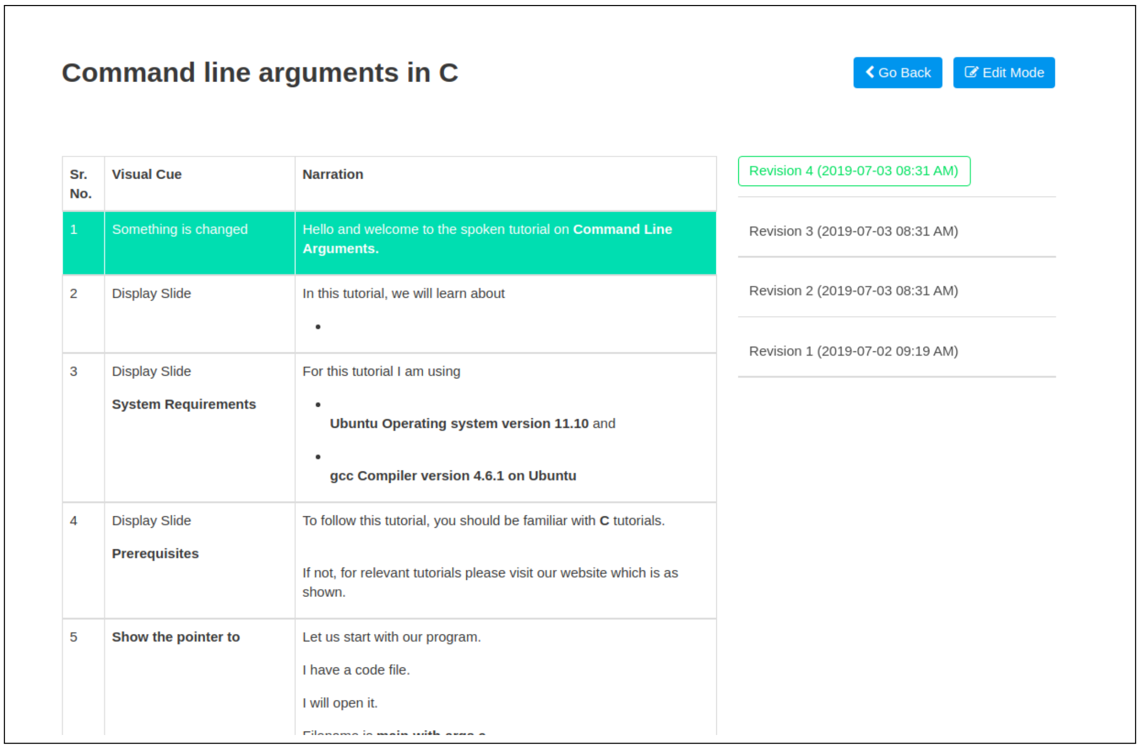
This is a test comment
admin | *0 minute(s) ago*

This is another test comment
admin | *0 minute(s) ago*

Figure 3.6: Add comment here

This will show all the comments for a particular slide and highlight that row in the table. Comments can be created by the reviewers and the user who created the script, both.

3.3.3 Revisions



Command line arguments in C

Sr. No.	Visual Cue	Narration
1	Something is changed	Hello and welcome to the spoken tutorial on **Command Line Arguments.**
2	Display Slide	In this tutorial, we will learn about •
3	Display Slide **System Requirements**	For this tutorial I am using • **Ubuntu Operating system version 11.10** and • **gcc Compiler version 4.6.1 on Ubuntu**
4	Display Slide **Prerequisites**	To follow this tutorial, you should be familiar with **C** tutorials. If not, for relevant tutorials please visit our website which is as shown.
5	**Show the pointer to**	Let us start with our program. I have a code file. I will open it.

Revision 4 (2019-07-03 08:31 AM)

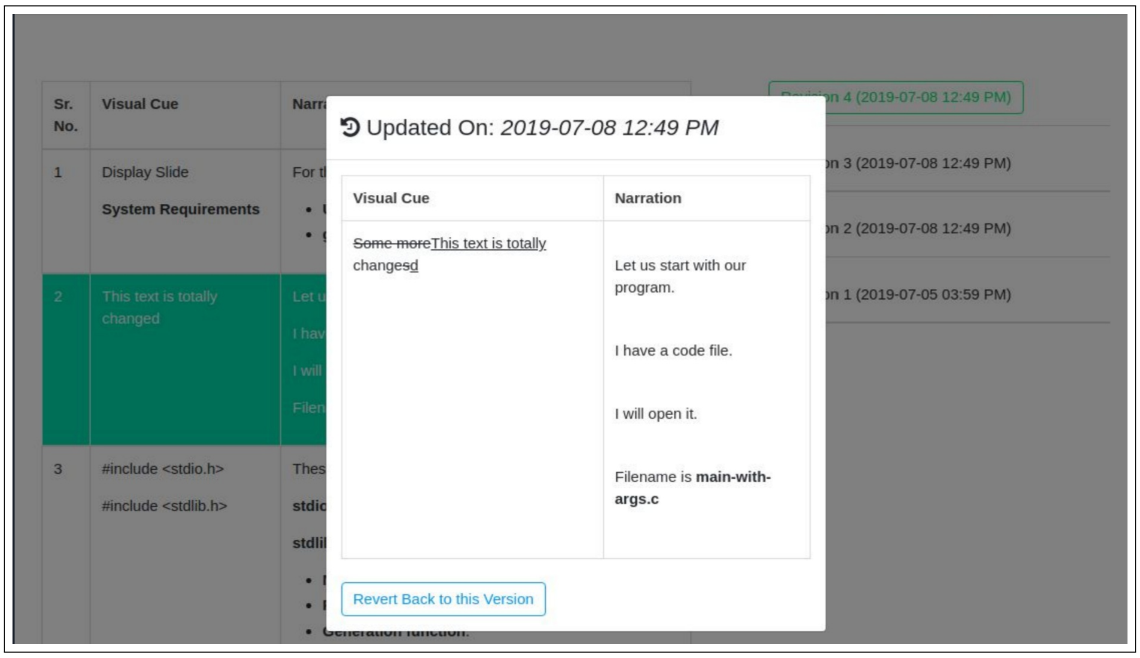
Revision 3 (2019-07-03 08:31 AM)

Revision 2 (2019-07-03 08:31 AM)

Revision 1 (2019-07-02 09:19 AM)

Figure 3.7: Revisions with date & time

This will show all the edit history made to the particular slide. Clicking on the revision will show that particular version of the slide, with the difference between the current version and the previous version. User has the facility to revert back to previous version at any point.



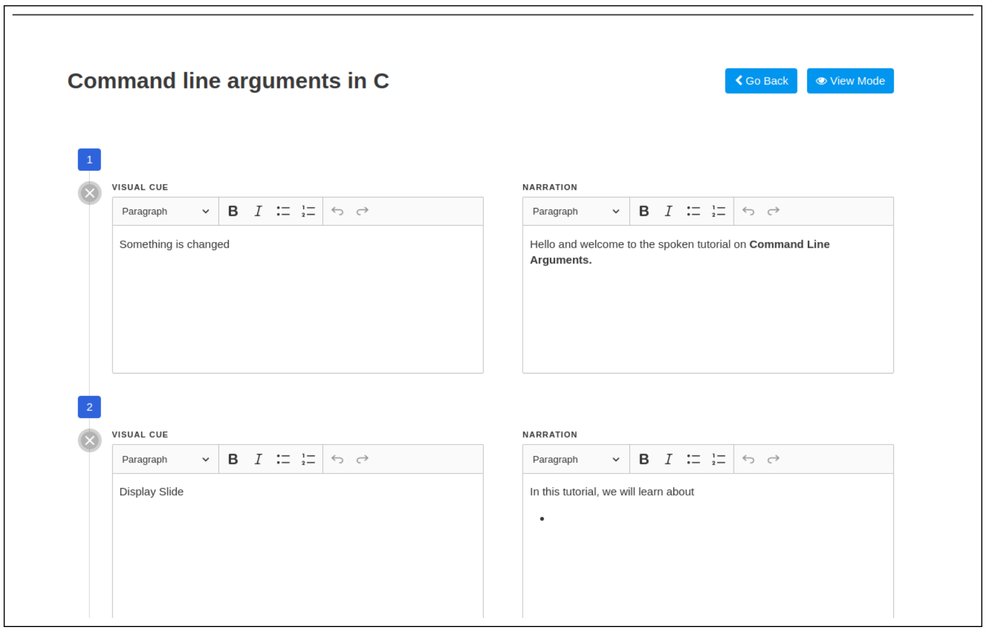
Updated On: *2019-07-08 12:49 PM*

Visual Cue	Narration
~~Some more~~This text is totally changesd	Let us start with our program. I have a code file. I will open it. Filename is **main-with-args.c**

Figure 3.8: Diff is shown with revert opt

3.4 Edit UI

Edit mode can be accessed from the navigation button on the top-right corner of the View Mode and can be toggled back from the same button.



Something is changed

Hello and welcome to the spoken tutorial on **Command Line Arguments.**

Display Slide

In this tutorial, we will learn about

-

Figure 3.9: Edit script here

This mode will show all the slides cue and narration data in an editor which can be used to edit that slide. It has an **autosave** feature so everything will be saved after making an edit and moving out of focus from the editor. More slides can also be added to the script in the edit mode.

Chapter 4

Module/libraries Used for Developing Frontend/Backend

1. Bulma as SASS/CSS library
2. ngx-text-diff for implementing a feature similar to ‘git diff
3. Libreoffice for converting doc/odt files to HTML

Chapter 5

Backend

The backend is written in **django**. Inside spoken tutorial project, a new module script manager is created. All the back end API are written in django-rest-framework. By calling the end points in the different method, the ***C.U.R.D*** (Create, Update, Read, Delete) is performed in the appropriate django models.

5.1 Analyze & Approach

Before starting, we analyzed the existing models and understand the flow of the system. In most of the cases, we used their models to fetch data. When required new models are created for scripts details, comments and reversions.

5.2 Rest API

- **/api-token-auth/**
  It takes the user credentials in **POST** request and it returns the JWT(JSON Web Token). This API doesnt require any authentication. Expect this API all the API has to be called with a valid JWT token in the header.

- **/api/foss/**
  This API only accept **GET** request and the user should be authenticated. It returns all foss category and its languages assigned to the current user. This data is retrieved from Contributor Role model.

- **/scripts/api/foss/{fid}/language/{lid}/tutorials/**
  This API only accept **GET** request and the user should be authenticated. It takes foss-id and language-id as a query parameter and it returns the tutorials for the given foss category and language. This data is retrieved from Tutorial Detail model.

- **/scripts/api/tutorial/{tid}/scripts/**
  This accepts GET, POST,PATCH, DELETE request and the user should be authenticated. This API handles the entire script data in slide level also.

GET

It returns the scripts details for the given particular tutorial id. If any script is created for the given tutorial id then for that script id, script details will be returned. That is if any instance is present in the Script model for given tutorial id and if any instance present in **Script Details** model for script instance id then that script details data will be returned.

POST

It takes an array of script details data. At first this checks if any Script instance is created for the tutorial id. If not new instance will be created. Then with that script id and received data will create a new instance in the **Script Details** table. Here when inserting larger no of data i.e iterating through each dictionary and inserting will be costly. So, the data will be serialized and whole array will inserted simultaneously. This is an optimal way of creating array of instance rather than inserting one by one. With respect to the actions the API returns True or False.

PATCH

It takes a dictionary contains 'pk' of that **Script Details** model. The update is based on pk. For that particular instance it will be replaced with received data. The update can not be made for a bulk data. It can be done in slide level only.

DELETE

It takes the slide id as query parameter and it deletes the instance in the **Script Details** model. After deleting, this function checks that there is any instance of same script. If not then it deletes the script data also. So, that new script can be crate.

5.3 API We Use

In API documentation we are going to discuss about **API** used in this project.

5.3.1 FOSS category List API:

This api returns a list of foss category and language assigned the current user
Allowed Method : `GET`
URL: /scripts/api/foss/
Permissions:
    Must be Authenticated
GET:
    Sample JSON:

```
[
    {
        "foss_category": {
        "id": integer,
        "name": string,
        "description": string
    },
    "language": {
        "id": integer,
        "name": string,
    },
    "user": string,
    "status": boolean
    }
]
```

5.3.2 Tutorials Details List API:

This api returns a list of tutorials for a particular foss category and language
Allowed Method: **GET**
URL: /scripts/api/foss/¡foss_id¿/language/¡langage_id¿/tutorials/
Url parameters:
foss_id: It is the id of the foss category that the user wants to view tutorial
langage_id: It is the id of the language that the user wants to view tutorial
Permissions:
Must be Authenticated
Must be assigned to the foss category and language

GET:
    Sample JSON:

```
[
    {
       "id": integer,
        "foss":integer ,
        "language": integer,
        "tutorial": integer,
        "level": integer,
        "order": integer,
        "script_status":boolean,
        outline:string
    }
]
```

5.3.3 Script Details List API:

This api returns a list of scripts,create scripts,update scripts for a tutorial id
Allowed Method: **GET, POST**
URL: /scripts/api/tutorial/¡tutorial_id¿/language/¡language_id¿/scripts/
Url parameters:
tutorial_id:It is the id of the tutorial that the user wants to view script details
langage_id: It is the id of the language that the user wants to view tutorial
Permissions:
Must be Authenticated
Tutorial must be assigned to the current user

GET:

Sample JSON:

```
[
    {
            "id":id,
            "cue": string(html by CKeditor),
            "narration": string(html by CKeditor),
            "order": integer,
            "script": integer
    }
]
```

POST:
It takes arrays of script details json and it returns True or False

Sample JSON:

```
{
"details":
    [
            {
                 "cue": string,
                 "narration": string,
                 "order": integer
            }
    ]
}
```

5.3.4 Script Details API:

This api update a script and delete script
Allowed Method: **PATCH, DELETE**
URL: /scripts/api/tutorial/¡tutorial_id¿/language/¡language_id¿/scripts/¡script_details_
Url parameters:
tutorial_id:It is the id of the tutorial that the user wants to view script details
langage_id: It is the id of the language that the user wants to view tutorial
script_details_id: It is the id of a particular slide, that the user wants to edit / delete
Permissions: Must be Authenticated
Tutorial must be assigned to the current user

PATCH:

Sample JSON:

```
[
    {
            "id": integer,
            "cue": string,
            "narration": string,
            "order": integer,
            "script": integer
    }
]
```

5.3.5 Comments API:

This api returns a list of comments,also it creates comments for a slide (script_detail_id)
Allowed Method: **GET, POST**
URL: api/scripts/¡script_detail_id¿/comments/
Url parameters:
script_detail_id:It is the id of a particular slide, that the user wants to view comment
Permissions:
Must be Authenticated
GET: Sample JSON:

```
[
    {
        "id": integer,
        "comment": string,
        "user": string,
        "script_details": integer,
        "time": string
    }
]
```

POST:
It takes a comment in json and returns True / False. Sample JSON:

```
        {
                "comment":string
        }
```

5.3.6 Reversions List API:

This api returns a list of reversion for a slide(script_detail_id) Allowed Method: **GET**
URL: api/scripts/¡script_detail_id¿/reversions/
Url parameters:
script_detail_id:It is the id of a particular slide, that the user wants to view comment
Permissions:
Must be Authenticated
GET:
Sample JSON:

```
[
    {
        "reversion_id": integer,
        "id": integer,
        "cue": string,
        "narration": string,
        "order": string,
        "script_id": string,
        "date_time": string,
        "user": string
    }
]
```

5.3.7 Reversions Revert API:

This api is to revert from current version to the some other version
It takes the reversion id and return True / False
Allowed Method: **PATCH URL: api/scripts/¡script_detail_id¿/reversions/¡reversion_id¿/**
Url parameters:
script_detail_id:It is the id of a particular slide, that the user wants to view comment
reversion_id:It is the id of the particular slide, that the user wants to revert
Permissions:
Must be Authenticated.

Bibliography

1. https://docs.djangoproject.com/en/2.2/topics/install/
2. https://angular.io/
3. https://angular.io/tutorial