Expertise
and insight
for the future

Guy Wardi

Online video presentation editor

Metropolia University of Applied Sciences

Bachelor of Engineering

Media engineering

Bachelor's Thesis

26 November 2018

Author Title Number of Pages Date	Guy Wardi Online video presentation editor 35 pages 26 November 2018
Degree	Bachelor of Engineering
Degree Program	Media Engineering
Professional Major	Media Engineering
Instructors	Kari Salo, Principal Lecturer

Learning via the internet has become somewhat of a standard for the young generation. Using multimedia as learning content fulfills the objectives of the traditional “teacher classroom” model. Online learning companies such as Udemy, CourseEra and Academic institutions such as Stanford, Yale, MIT, Harvard, Berkeley and Oxford use a combination of video, audio, text content in their online courses.

This thesis study focuses on an online tool for creating intuitive educational videos. Implementing modern programing languages such as JavaScript, Angular.js, jQuery, NodeJS and ExpressJS the goal is to build a prototype for editing and creating video presentations targeted at company employees and students who have no prior editing skills.

Keywords	Online video presentation editor, JavaScript, Angular, jQuery

Table of content

1 Introduction 1

2 Online video and presentation editors 2

2.1 Video editing tools 2
2.2 Presentation editing tools 2

3 Software overview 5

3.1 Back end technologies 5
3.2 Front end technologies 5

4 Piranha 8

4.1 Requirements 8
4.2 UI overview 10
4.3 Source - Media assets and Overlay assets 15
4.4 Data model 18
4.4.1 Populating the model 18
4.4.2 Model manipulation 22
4.4.3 Overlay assets 27

5 Discussion 34

6 Conclusion 35

References 36

List of Abbreviations

NLE	Non Linear Editing
NPM	Node Package Manager
MVC	Model View Controller
UI	User Interface
AWS	Amazon Web Services
API	Application Programming Interface
UX	User Experience
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
URL	Uniform Resource Locator
CRUD	Create Read Update Delete, are the four basic functions of persistent storage

1 Introduction

This thesis will discuss online video editors and presentation editors. A prototype editor combination both video and presentation editors was developed for DataFisher Oy, which is a media production and E-learning company. Due to the naming conception of the company the application is called Piranha. This work is divided between me and one other developer. The scope of my work is front end development, prototype the application and check if the prototype could be made into a viable product. The scope of this work is limited mainly to the client side.

Datafisher provides its customers with an LMS (learning management system). Learning management system is a software for administration, documentation, tracking, reporting and delivery of electronic educational technology. They are also called E-learning courses or training programs [1]. The client has noticed that a major part of the content tracked by the LMS is educational videos. Educational videos are basically video content for educational purposes.

Datafisher required a web based application that will be similar to a video editor but the end user could integrate and overlay some informative such as text, company logo and preconfigured animations from an animation library as in a presentation editor [2]. The potential customers are not professional video editors, and some have no experience in video editing.

The goal of this project is to build a cloud-based tool with an easy to use user interface for amateur users, who are not experienced with video editing, to create educational videos with. Such tool should combine features from both traditional video and presentation editing tools. The output should be a video. The output is the product of multimedia assets such as audio, video and images, called clips. The user should be able to trim and concatenate clips in order to generate a new video sequence. As a hybrid between a video editor and a presentation editor, it should allow the user to overlay animations and effects over the original media assets from a pre-configured animation library. Clips should also act as slides by allowing the user to input informative data similarly to presentation software as the common MS PowerPoint [2].

2 Online video and presentation editors

2.1 Video editing tools

Video editing is the process of editing segments of motion video footage, special effects and sound recordings as part of the post-production process. There are different types of video editing, relevant to this study is the video editing software that is responsible for post-production video editing of digital video sequences on a computer called non-linear editing system (NLE). It has replaced linear editing where traditionally flatbed celluloid film was cut and pasted to create a sequence [3]. NLE software is typically based on a timeline interface paradigm where sections of moving image video recordings, known as clips, are laid out in sequence and played back. The NLE offers a range of tools for trimming, splicing, cutting and arranging clips across the timeline. As digital NLE systems have advanced their toolset, their role has expanded and most consumer and professional NLE systems alike now include a variety of features for color manipulation, titling and visual effects, as well as tools for editing and mixing audio synchronized with the video image sequence [4].

Native desktop video editing applications such as Adobe Premiere and Windows Movie Maker are the basis for video editing software, they are designed according to the Non Linear Editing (NLE) approach. based on a timeline interface paradigm where sections of moving image video recordings, known as clips, are laid out in sequence and played back[4]. As an inspiration for video editing features, the customer introduced the developers with an online video editor called WeVideo.

WeVideo is an easy-to-use cloud-based video editor. Built as SPA (single page application) WeVideo has an intuitive user interface with four components:

- Assets – Where media assets such as image, audio and video that have been uploaded by the user are stored.

- Timeline - The order of the clips arranged on the timeline determine the chronological order of the final export file (video). Timeline objects are media assets. The timeline also allows to add transitions and trim the clips by starting and ending time.

- Preview – Displaying the selected/ current clip with the modifications the user has made in order for the user to see and decide if he is pleased with the changes

made to the sequence. The preview allows the user to view the sequence before the final export.

- Single video edit dialog - additional manipulation options such as; transforming, animating and positioning of the clip.

Media items in the timeline integrate into a sequence, with additional sound effects and audio the user will be able to create a high-quality video.

Single video manipulations:

- Text and watermark overlay – The user can set overlaid text by indicating start and end times, as well as location parameters. However, the text location is limited to preconfigured options.
- Transformations – Rotating, scaling and positioning current frame
- Adjusting frame size – Allows the user to add black bars to the frame, so that it fits the export resolution.
- Volume – Available only for videos and audios clips. Allowing the user to set the volume and the audio fade.

2.2 Presentation editing tools

Presentation software package is used to display information in the form of a slide show. It has three major functions: an editor that allows text to be inserted and formatted, a method for inserting and manipulating graphic images, and a slide-show system to display the content [2]. Text, graphics, movies and other objects are positioned on individual pages or "slides" and played as a sequence by the user's choice as in MS PowerPoint. Most presentation software allows transitioning between slides. As inspiration for presentation editor features, the customer introduced the developers with an online presentation editor called Powtoon.

Powtoon is a cloud-based presentation editor that allows users to create animated presentations using pre-created animation objects. Powtoon and WeVideo output a video file. However, Powtoon is a presentation editor based on slides and WeVideo is a video editor using a timeline sequence. PowToon is Web-based animation software that allows users to create animated presentations by manipulating pre-created objects, imported images, provided music and user created voice-overs [5]. Powtoon output is a

video. The resulting video in Powtoon consists of scenes with clear start and end, whereas WeVideo and Piranha editors can have "continuous change" in the video content. Powtoon is designed similarly to Powerpoint and is more similar to a presentation editor than to a video editor. It delivers a DIY (Do It yourself) solution for making ubiquitous marketing videos, demos and educational videos. Powtoon provides out of the box collection of cartoons, animation library. Its cartoons are sourced from designers, animators, voice actors, and sound artists. With Powtoon, one can create videos and animations. The videos are created as slide sets, where each slide contains an animation and the next slide will have a next "scene" in animation, typically continuing the story fluently. The resulting video is an animated video, not a slide-set. Powtoon is aimed at non-professional designers and video editors, and it allows the user to drag-and-drop animation objects onto slides as shown in figure 1.

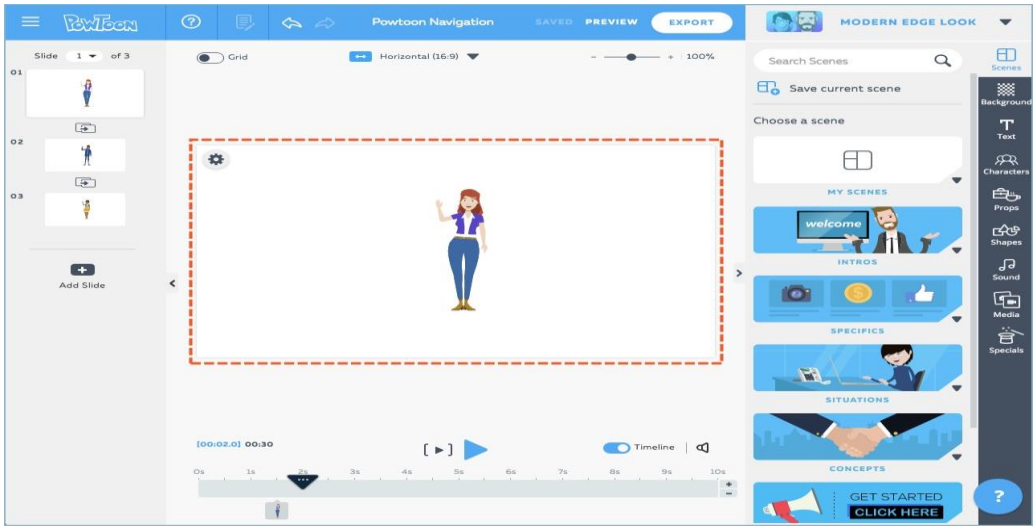

Figure 1: Powtoon interface

The result is a professional-looking demo presentation. The service also includes templates to get started (e.g a product demo teaser, event invitation, etc.). The resulting presentations can be uploaded to YouTube or shared on social networks like Facebook and Twitter or downloaded as a video file [6].

3 Software overview

This section describes the components of the technology stack for implementing the software solution, with emphasis on the client side since the scope of my work is mainly front end. As mentioned, Piranha is an **online** tool.

3.1 Back end technologies

Node.js - A software platform for developing event-driven applications. Node.js is based on the JavaScript-runtime engine, V8, which was built for the Google Chrome browser. [7]. The Node Package Manager (NPM) is managing all node modules and packages which have been made publicly available by many different users, typically consist of JavaScript files and a manifest called package.json.

Express.JS - Express is a minimal and flexible framework for Node.js. Built on top of Node.js, making all of Node.js features available, it provides developers robust simplified functionality. The reliability and simplicity of Express.JS have made it the choice of companies such as MySpace, PayPal and Persona [8].

3.2 Front end technologies

JavaScript was originally developed as a simple scripting language. However, nowadays JavaScript is one of the most popular languages world-wide. Since JavaScript supports dynamic features, it allows adding object properties dynamically and even delete them during program execution. Since JavaScript became so popular, many JavaScript users have created pre-written JavaScript files which are known as JavaScript libraries. Where complex manipulations of the DOM are needed, it gets complicated using plain JavaScript, which is where JavaScript libraries and frameworks shine. A library is a collection of functions that are useful when developing web applications [9]. The developer's code is in charge and simply calls upon the functions provided by the library in order to accomplish some repeated common behavior. e.g., jQuery. A JavaScript framework takes charge of the overall functionality of the application, and the framework, in turn, calls into the developer's specific code in order to customize the behavior to accomplish the specific application functionality. e.g., Angular.js. The main difference between a library and

a framework is that a framework clearly defines how the application should be implemented which constrain the developer by the dictates of the specific framework [10].

jQuery is an open source JavaScript library the goal of which is to simplify interactions between JavaScript and HTML. As mentioned in jQuery official website, make things such as events handling, animation and Ajax much simpler [11]. To create the simplest user experience for Piranha users, jQuery Draggable and Droppable widgets are used across the whole application, those widgets are making the usage of drag and drop interaction in the application much simpler and richer.

Angular.js is a JavaScript library that is classified as framework since it exhibit full-stack capabilities and properties that are not found in typical JavaScript libraries. Angular is mostly used for building modern web applications. Angular.js main functionality is to display data provided by the client on web pages. That helps with better dynamic page updating. Angular.js is designed to build an application using MVC (Model View Controller) approach [10].

MVC architecture is a software design pattern for developing web applications. The MVC approach enables to isolate domain logic from the UI, permits independent development, testing and maintenance. The separation of concerns is what allows independent development of each part [12]. The Model contains the data and logic, the View contains the visual layout and presentation, while the Controller connects the two as shown in figure 2.

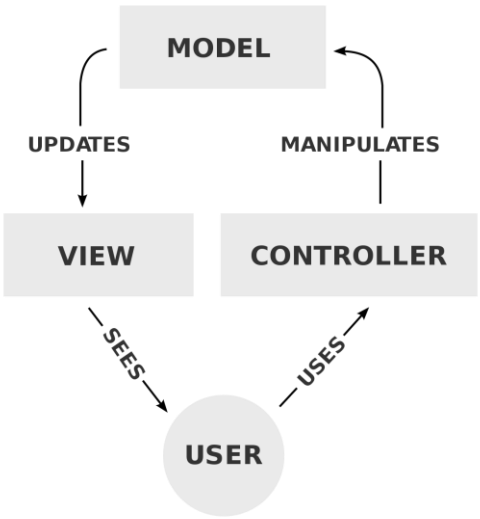

Figure 2: MVC architecture

In Angular JS the MVC approach is implemented by JavaScript and HTML. The Model and Controller are implemented in JavaScript while and View is using HTML.

View - The view is responsible for displaying the layout of the application using the model data. Interactions with the view will cause the controller to manipulate the model data which will result in updating the view.

Model - The model component is responsible for all data related logic. Can represent data transferred between the Controller and the View components or any other business logic data. Models may contain functions that are invoked by user actions. In Angular JS there are two ways of configuring model variables, $scope which creates an object that is accessible from current component and $rootScope which acts as a global scope variable and is accessible from anywhere in the application. Listing 1 will define $scope and $rootScope Models:

```
//Defining a new model variable
$scope.message = "Hello World"

//Defining a new global Model variable
$rootScope.message = "Hello World"
```

Listing 1. Defining Model variables in Angular JS.

Controller - The Controller is responsible for use the model used by the view and to set up the dependencies required for rendering the view or handling user interaction with the view. One way of defining a Controller is by manually adding it to the JavaScript variable representing the Angular JS application as seen in Listing 2 below.

```
var indexController = app.controller("indexController", function
($scope) {
// controller logic goes here

});
```

Listing 2. Creating a Controller by adding it to a JavaScript variable.

4 Piranha

Piranha is a video editing software for post-production video editing of multimedia sequence on a computer called Non-Linear Editing system (NLE). It has replaced linear editing, where a traditional flat cellulose film was cut and paste to form a sequence [13]. Piranha is based on a timeline interface paradigm where multimedia items, known as clips, can be trimmed, rearranged and have overlaid text and images. The clips form a sequence, which will be available to share and download.

Piranha software package includes presentation editor functionalities that were inspired by traditional presentation editors. An editor for inserting, formatting and manipulating text and images. Displaying content in a slide-show system [14]. A single clip is behaving as a single slide in a presentation editor. The user can add, modify or position images and text by a simple drag and drop functionality. As most presentation editors, Piranha's future development will also include transitioning functionality for text, images, clips and other objects.

4.1 Requirements

As mentioned in the introduction, the application is built for Datafisher. The customer did not list the requirements in a traditional way. Instead, two applications were shown the developers, WeVideo and Powtoon, and the customer required some sort of application that will combine features from both, as in a hybrid application

The designated application aims to provide educational instructors or students with a tool for creating educational videos in the simplest way possible, no prior experience in video editing is needed. Piranha can be referred to as a hybrid application since it combines both slideshow and NLE approaches. In an NLE application video clips are laid in a sequence as in a presentation editor where slides are creating a sequence called a slideshow, ergo the timeline concept is common to both but being used differently.

Multimedia assets such as videos, images and audio inside a sequence could be cropped, trimmed and manipulated as a slide where it could be overlaid with clipart, text and other pre-defined animations. The animation, clipart library is unique to the presentation software. As an amateur tool for non-professional users, Piranha is designated to

be used by subscribers to produce educational videos that are, according to the client, usually less than 15 minutes long.

The following requirements are the building blocks for Piranha and are derived from the analysis of WeVideo and Powtoon. The list of requirements are for the client side only since the scope of this work is limited only to client side development.

1. Upload - the user should be able to upload multimedia and overlay assets.

2. Timeline - a user should be able to add or rearrange multimedia assets in an NLE timeline in order to form a sequence.

3. Audio - a user should be able to overlay sound over a video clip in the sequence.

4. Trimming - a multimedia asset duration or an overlaid asset appearance duration should be available for trimming by defining starting and ending time.

5. Overlaying textboxes and images - Overlaying animation is the product of overlaying images or textboxes by time and x, y coordinates. Text and images are placed in different coordinates and times using drag and drop. The images are preconfigured and uploaded by the user, however, the text content, duration, start, end time and location should be dynamic.

6. Preview - the user should be able to play the full sequence including audio and watch the outcome in the preview component.

7. Export - the user should be able to export the sequence in order to download or share the exported video.

4.2 UI overview

Assuming a subscriber to the application logs in, the user will be routed to the dashboard view. In the dashboard view the user can choose to create a new project or edit an existing project. As seen in figure 3**,** Mouse pointer hovering over an existing project introduces the user with the option to continue editing the project, or delete the existing project. Those options are marked by the pen icon and the trash bin icon. Each existing project has a title and by hovering with the mouse over an existing project more details such as date and length are reviled. On the left side of figure 3 there is a donut that indicates how much storage is available for the Piranha account subscriber based on their subscription plan.

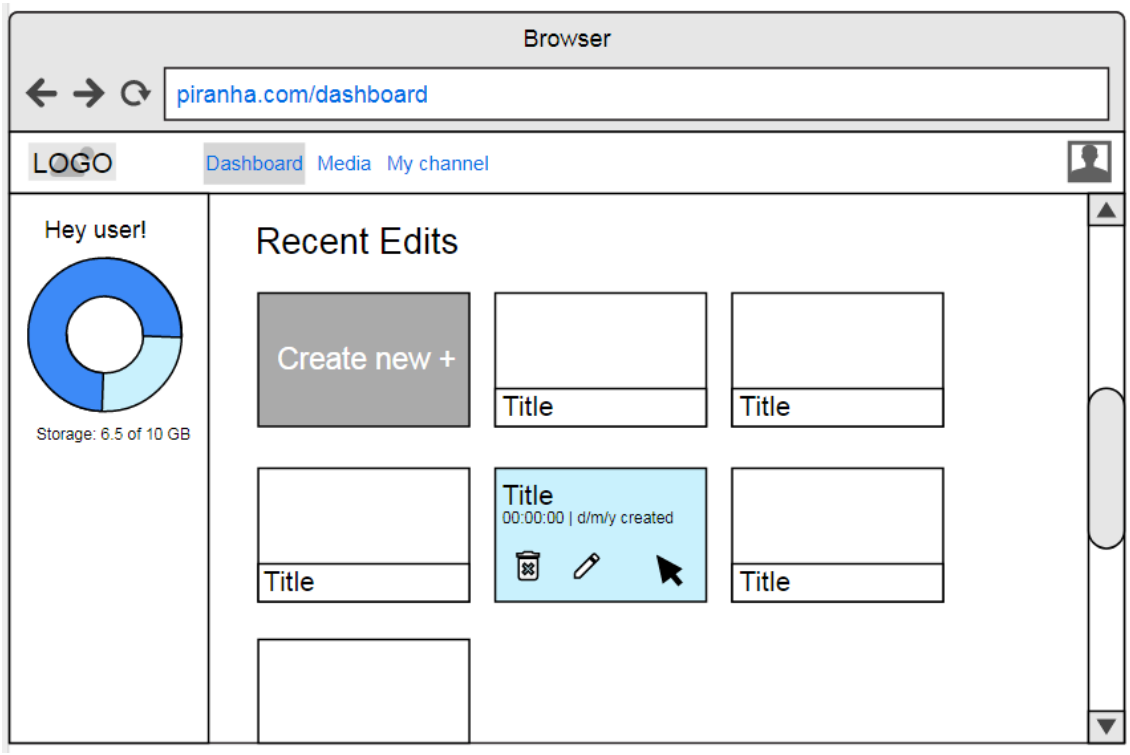

Figure 3: Dashboard view

Clicking on the pen icon or create new project button will route the user to the main view of the application, the Edit view. As seen in figure 4, the edit view is divided into three main UI components. Since the scope of this work is limited, the study will focus mostly on those three UI components.

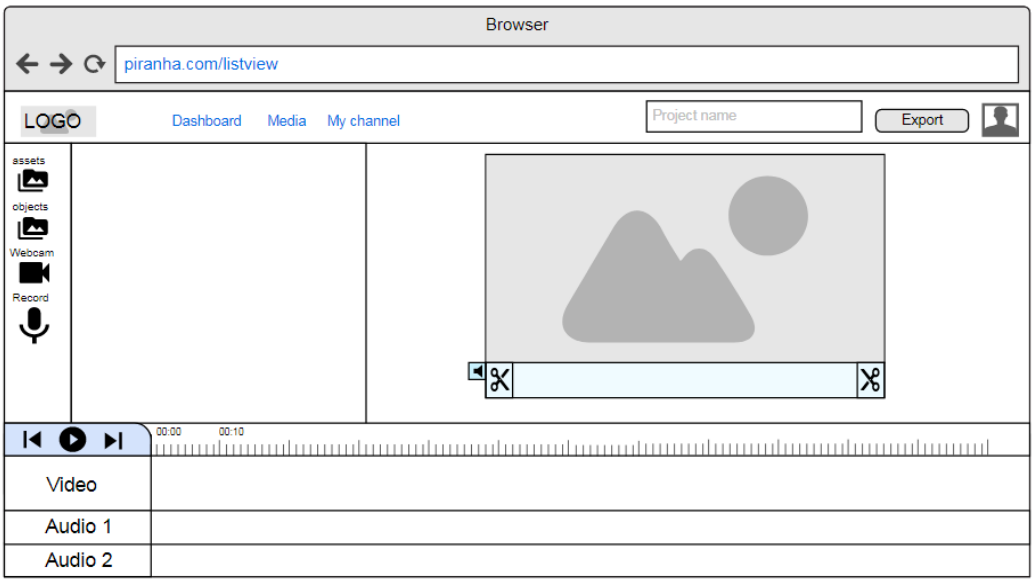

Figure 4: Edit view

The export button is located in the right top corner of figure 4. Once pressed, the screen will become blur and a loading icon will appear in the center of the screen indicating the user that the exporting of the sequence is loading. When completed, a link to download the exported sequence will appear and the exported sequence will also be saved in the "My channel" view. This view stores all of the exported sequences the user has created.

Side vertical menu - the vertical menu is located on the left side of figure 4. The vertical menu allows the user to navigate between different assets libraries such as multimedia assets and overlay assets. The vertical menu contains deferent features as well, such as Webcam recorder, audio recorder and an upload feature.

Each user has multimedia and overlay assets (video, image and audio files) stored in Amazon S3. This will be described in greater detail in subchapter 4.2. The application supports all types of multimedia assets (image, audio and video) with no constraints on formats or frame size. When the multimedia assets are fetched to the client-side from the server, thumbnails of the multimedia and overlay assets are loaded to the corresponding assets folder as seen in figure 5. The user can drag and drop assets from the multimedia library to the timeline and therefore create a sequence.

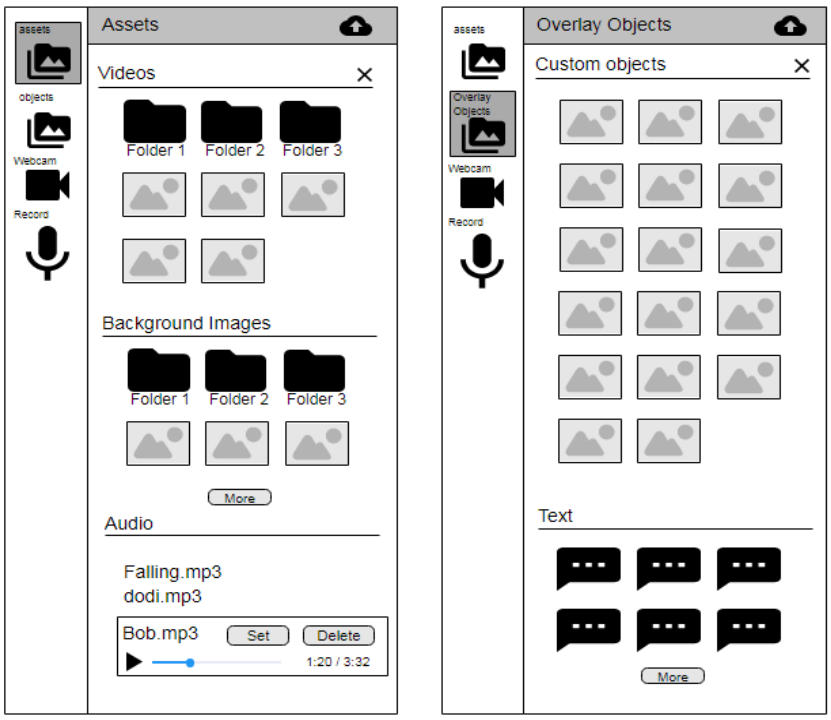

Figure 5: Vertical menu -multimedia and overlay assets

The multimedia assets are sorted by type and the user is able to create folders to store his assets in a more organized way, but since it is not part of the project requirements it will not be deeply described.

The Timeline as seen in figure 6, is where the user can create and manipulate his sequence. The Timeline is built as a traditional video editor timeline. The Timeline contains the current sequence of clips subjected to editing. Assets from the multimedia assets folder can be dragged, dropped and rearranged into the Timeline UI component.

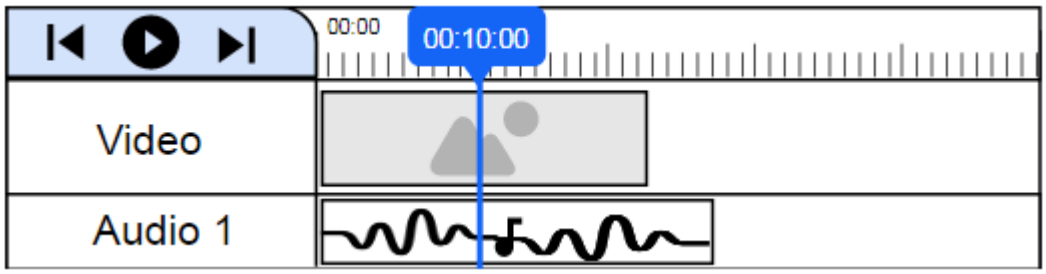

Figure 6: NLE Timeline

There are currently only two tracks in Piranha's timeline, one for visual media and another for audio, more will be added after the beta release. Once the user starts dragging an asset from the multimedia folder the appropriate track will highlight to signal the user where this asset should be dropped. If the user start dragging an audio asset, then the

audio track will highlight. This gives indication for users that are not experienced with video editing and help guiding the users on where they should place the selected media and therefore also enrich the UX. By hovering with the mouse over a clip in the Timeline sequence, a trash bin icon will appear and by clicking it the user can remove the clip from the Timeline sequence.

The Playhead, the blue indicator, is connecting the Timeline to the Preview by letting the application know what frame to display in the Preview at any given position on the Timeline or where the application should start playing the sequence from. The Playhead is draggable and as seen in figure 6, shows the current time of the sequence as most video editors Timeline Playhead does. When a user plays his sequence the Playhead is moving along the Timeline to indicate the user the current playing time and informing the application when to start playing to the next clip. Since image does not have a duration the Timeline Playhead play crucial part when it comes for concatenating the sequence and forming a smooth transition between the clips.

As seen in figure 6, the play button starts the preview of the sequence, created by the user, based on the location of the Playhead. If the Playhead is positioned at the fifth second then the play button will start playing the sequence from the fifth second. There is an HTML audio tag which is not visible. In case there is an audio asset in the Timeline's audio track, the audio will start playing when the Playhead reach the starting time of the audio. By clicking on the audio clip the Preview will load a preconfigured image of audio waves to indicate the user that the audio selected has loaded and can be modified in the same way as any other media asset. Audio clips cannot have an overlay asset.

The preview is built using three HTML5 layers overlapping each other, video tag overlaid with image tag and canvas tag. While playing the sequence from the Timeline, the Preview is where the user can see the outcome of his sequence or of a selected clip. Since the Timeline sequence might contain images and videos, the Preview is built in a way that once an image is supposed to start playing the video tag becomes hidden and the image tag is shown and vice versa. The HTML canvas tag is responsible for displaying the overlaid textboxes and images on top of the video or image tag, doing that gives the user the best way to experience how the outcome of the export will look like. The canvas tag is the first layer of the three because it needs to show on top of the playing visual media (image/video).

The Preview will play the sequence based on the Timeline Playhead which is controlled by the user. If the user is simply dragging the Timeline Playhead over a clip than the clip will be displayed in the Preview. For example if the user dragged the Playhead to the fifth second of a video clip then the Preview will display the frame of the video at that specific time. If the user play the sequence, the Preview will display the appropriate HTML tag based on the current clip the Playhead is on. The Preview's HTML video tag, assuming a video clip is to play, will just play the video. However if an image is to be displayed, the Preview will display the HTML image tag loaded with the image URL and wait for the timeline Playhead to move on to the next clip in the sequence and act accordingly. Each clip in the sequence is acting like a slide in a presentation editor, meaning that each clip can have overlay images or textboxes. In case the Preview displays a clip which the user added an overlay item, the HTML canvas tag will display the overlaid item on top of the playing clip.

As seen in figure 4 the Preview is located on the upper right side. Below the Preview is a Slider that specify the current time of the clip playing and not the whole sequence time. The use of the Slider is to set the starting and ending time of a selected clip. Dragging the left handle of the Slider will determine the starting time of the clip while the left handle will determine the ending time of the clip. The Slider also controls the timing of the overlay assets, in subchapters 4.3.3 and 4.3.2 the overlay assets and the use of the Slider functionality will be detailed and described.

4.3 Source – Multimedia assets and Overlay assets

Piranha's source is divided into two, multimedia assets and overlay assets. Multimedia assets contain three different types of media - Video, Image and Audio. Multimedia assets use is to form a sequence by simply dragging an asset and dropping it in the highlighted Timeline track. Overlay assets has two types, Images and Textboxes. Each Piranha user has his own bucket in Amazon Simple Storage Service (Amazon S3) that stores all multimedia and overlay assets, there are some preconfigured overlay assets that are available for every user. The user can upload assets to Amazon S3 via Piranha's upload feature.

When the edit view is loaded, an HTTP request is sent to Amazon S3 to fetch the data. The data which consist of multimedia and overlay assets comes in JSON format with two properties for each asset, URL and Title. When the data is fetched, a function for identifying the type of each media asset is called. The function is using HTMLVideoElement and HTMLImageElement object with deferred object to validate the type of the multimedia asset using the URL of each asset. Deferred object is a jQuery chainable utility object that can register multiple callbacks into callback queues [15]. It allows an organized way of testing multiple URLs at the same time. By the time the function ends, all multimedia assets obtain a new property called Type with the appropriate value. The type property value is an integer that symbolizes the multimedia type, one is Video, two is Image and three is Audio.

As described above, the source data is stored in Amazon S3, some preconfigured assets are stored in the user's bucket in Amazon S3. However the user can upload assets to his Amazon bucket via Piranha's upload feature. There are no constraints on the media format or frame size.

Figure 7: Upload popup feature

The user can either choose a file or simply drag it on top of the upload popup, as seen in figure 7, in order to start uploading the selected file to his private Amazon S3 bucket. Once the upload is completed the appropriate assets folder will refresh and update to include the newly added file and the corresponding thumbnail will appear. The user can upload an asset either to the multimedia folder or to the overlay folder. Only images are allowed to be uploaded to the overlay assets folder.

```
$scope.upload = function(file) {

     $http.post('/api/uploads', {
          filename: file.name,
          type: file.type,
          bucketName: $localStorage.currentUser.bucketName
     })
     .success(function(resp) {
          $.ajax({
               type: 'PUT',
               url: resp.url,
               ACL: 'public-read',
               headers: {
                    'Content-Type': 'text/plain;charset=UTF-8'
               },
               // this flag is important, if not set, it will try to
               send data as a form
               processData: false,
               // the actual file is sent raw
               data: file
     })
     .success(function() {
          getVideos({"id": $localStorage.currentUser.bucket-
          Name,"folder": "videos"});
     })
```

```
    .error(function(resp) {
        alert("An Error Occurred Attaching Your File");
    });
}
```

Listing 3. Upload function

The upload function is using HTTP POST request to send the uploaded files straight to the user's Amazon bucket, as seen in listing 3.

4.4 Data model

Piranha's data model is an array of JavaScript objects, each object represent a clip from the Timeline's sequence. The array is indexed exactly as the clips are indexed in the sequence, if a clip was moved to a different index then the model will rearrange accordingly. The data model is acting as the Model in the MVC architecture, responsible for maintaining data. Changes made to the model will manipulate the view.

4.4.1 Populating the model

In order to populate the model it is necessary for a multimedia asset to be dragged to the Timeline. The drag and drop functionality is built using jQuery UI Draggable and Droppable widgets. JQuery UI is built on top of the jQuery JavaScript Library and enable developers to customize both behavioral and presentational aspects of their application [16]. jQuery UI Droppable widget includes very useful triggered events and options. The class option is triggered when the user started the dragging process and stops when the drag has ended as shown in listing 4. This allows a very simple way for adding and removing the appropriate classes in order to highlight the approved droppable location, in this case the Timeline's appropriate track will highlight depending on the asset dragged.

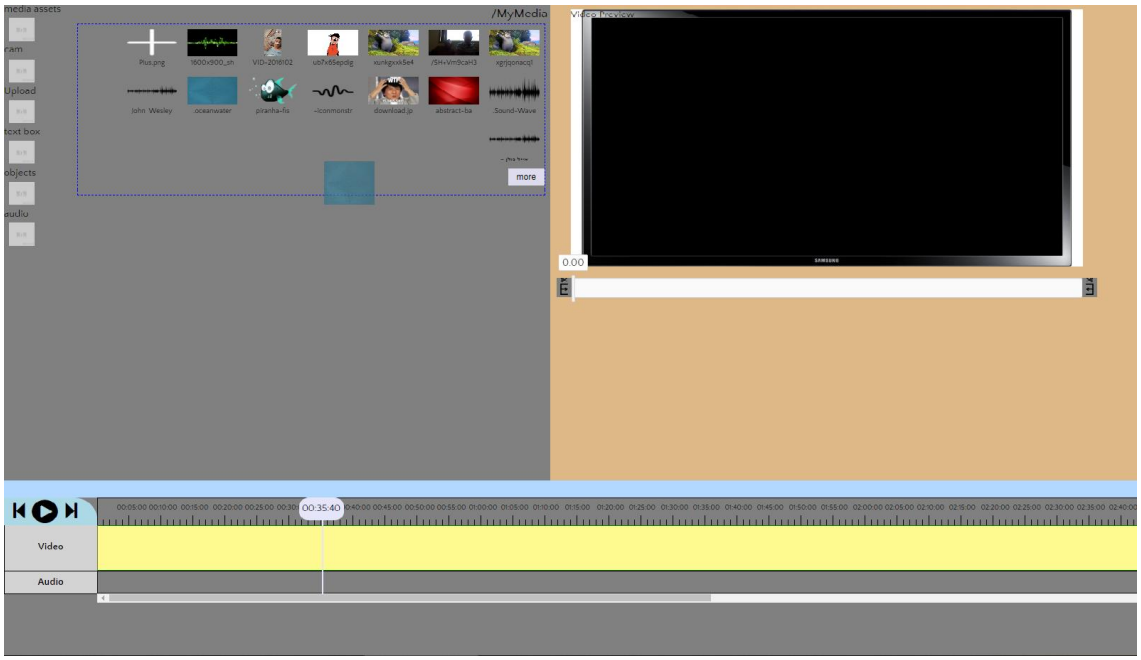

Figure 7: Edit view - Timeline track is highlighted while a multimedia asset is being dragged

The drop event is triggered when the dragged item is dropped in the allowed location. The drop event is triggering the creation of a new JavaScript object that will be added to the model. When a multimedia asset is being dragged the activate event is triggered and checks whether the sequence is being played or not. If it is playing then a call to pause the sequence from playing will execute, as shown in listing 4.

```
$(".droppableMedia").droppable({
     accept: ".draggableMedia",
     scroll: true,
     classes: {
          "ui-droppable-active": "ui-state-highlight"
     },
      activate: function(e, ui){
          //Activate event triggered when dragging starts.
          //Call to pause the playing sequence in case it was play-
          ing.

          if($rootScope.model.playing == true){
               $scope.pauseVideos();
          }
      },

      drop: function(e, ui){
          //Once the dragged item was dropped a function to add a
          new model item is triggered.

          sortableService.sourceDropped(ui, $scope.rootItem);
      },
});
```

Listing 4. jQuery UI Droppable.

As described in chapter 4.2 the source assets contain three properties, URL, Title and Type. Those properties pass on to the newly added JavaScript object and five new properties are added as well. When a multimedia asset is dropped in the allowed location, the drop event is triggered and call the "sourceDropped" function to execute. This function contain a set of functions. These functions are responsible for adding new properties to the newly added JavaScript object. The following part will describe how the functions obtain the values for the new properties.

URL - using the URL property, a function responsible for buffering is executed. Buffering a video is based on packet transmission and is influenced by delays, data losses and bandwidth limitations which causes devastating influence on the perceived quality of the media played [17]. Since modern browsers buffer "only what is needed", which means that the browser buffers only a part of the media and then stops, can cause errors and

bad UX. As example if a video has paused to buffer while the Timeline Playhead and the added audio keep on playing, it will cause the sequence to be out of sync and will not represent the exported version, that can cause confusion for the user. Hence to prevent unwanted buffering time and to enrich the UX, once the user drags an audio or video asset to the timeline, a buffering function is called. The buffering function force the browser to buffer the media item fully and store the fully loaded data in a Blob URL. Playing the media asset using a Blob URL prevents any buffering time and ensure smooth play of the sequence. When the buffering function is called, as seen in figure 8 below**,** a progress bar appears to inform the user that the wanted media is loading. This will only happen for each media item used, if the same media item is being dropped to the Timeline track again, the corresponding Blob URL will be used so the user will not have to wait.

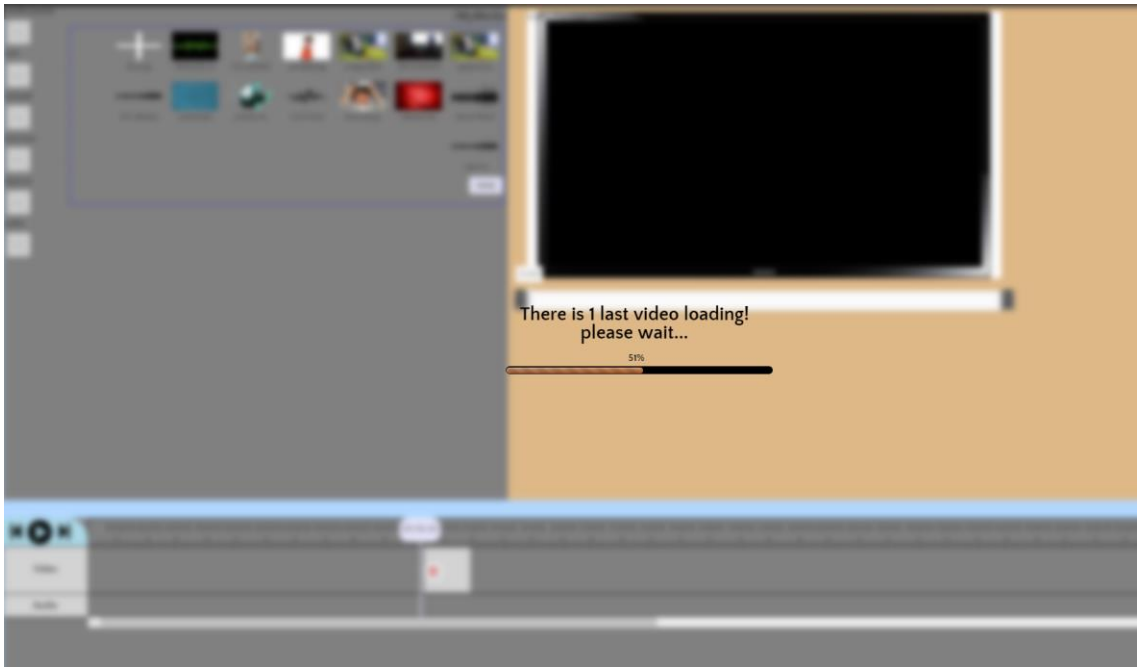

Figure 8: Edit view - Buffering a video that was added by the user to the Timeline

Duration - The duration property is crucial for calculating the width of the clip in the Timeline sequence. Each clip on the Timeline has a width based on the duration of the clip, changing the starting or ending time of a clip will cause the clip's width to change. The duration is acquired using a function that creates the appropriate media HTML tag in order to load the asset's blob URL and obtain the duration of the media asset via loadedmetadata EventListener. The loadedmetadata event occurs when the metadata of the media asset has been loaded [18]. When the metadata is loaded the HTML Audio or

Video DOM duration property receives an integer value and assigned to the model item's duration attribute.

Index and position - the position value is given through the jQuery UI drop event, which provide the position of the place the clip was dropped. The position property is the value of the left side of the clip and is taking only the x axis position value. A function then runs and compares the position of the new clip with the positions of other clips in the Timeline. The function then receives the index of the clip in the Timeline and will set it as the new value for the index property.

Start and end time - Initially when a new model item is created, by dropping a multimedia asset in the Timeline track, there is no predefined starting or ending time for the clip. Hence the value for the start time will be zero while the value for the end-time property will be the same as the duration, meaning that the clip will start playing from the beginning of the media asset and will stop playing when reached the end of the media asset. Images are preconfigured to have an end time of 5 seconds.

The overlay property is an array of JavaScript objects that represent overlaid assets. This property is not added to the model through "sourceDropped" function, the overlay property is added once an overlay asset is added to a clip in the Timeline sequence. Therefore this property will be described in subchapter 4.3.3 Overlay assets.

4.4.2 Model manipulation

The model in Piranha is a global variable that stores an array of JavaScript objects representing clips in the Timeline sequence. The model item contain properties that affect the outcome of each individual clip, hence it affects the whole sequence. The user can modify these properties to create the desired outcome. The Model item properties are listed in table 1.

Table 1. Model item properties

Model item properties	**Value**
Index	Integer marks the index of the clip in the sequence. The value is based on the location that the newly added clip was dropped at in the Timeline.
Title	String, the title of the clip.
URL	String, media asset URL or in case of video and audio assets a Blob URL is used.
Type	Integer, represent the clip's media type (Video, Image or Audio).
Duration	Integer, The clip's duration.
Start-time	Integer, The time the clip is set to start from (default value is 00:00).
End-time	Integer, The time the clip is set to end (default value is the duration of the clip).
Position	Integer, the x position of the clip on the timeline. This attribute is used to keep all clips in the location defined by the user. Important especially when leaving and returning to an existing project.
Overlay Available only if an overlay asset is added Not available for audio clips	Array, stores JavaScript objects that represent the overlaid text and images.

Assuming a user would like to trim the starting or ending time of a clip, there are two options in doing so. First option is to simply resize the clip's width in the timeline by dragging the left side of the clip to modify the starting time or the right side to modify the

ending time, this method is available by using jQuery UI Resizable widget. jQuery Resizable enables DOM elements to be resized by giving the element draggable resize handles that allows to change the width and height of the element to the desired size. It is possible to specify one or more handles as well as min and max width [19]. In Piranha the max width will be the duration of the clip and the height resize option will be disabled. The second option is available by clicking on the wanted clip, which will trigger the Preview to load the clip and the Slider, which is below the Preview as seen earlier in figure 4, will be adjusted by the clip's duration. By dragging the Slider's left handle the user can select the starting time and by dragging the Slider's right handle the user can select the ending time of the clip. The Slider is created using a JavaScript library called noUiSlider. noUiSlider is a responsive lightweight range slider with many features and events [20]. Each time a clip is loaded to the Preview, the Slider will also receive the duration, starting time, and ending time properties from the selected model item and will adjust accordingly. Once a user has dragged one of the handles, an onChange event is triggered and through that model item property is changed.

```
slider.noUiSlider.on('change', function(values, handle){
     if (handle == 0) {
     //Left handle was moved
     $rootScope.model[$rootScope.index].startTime = values[0];
     }
     else if (handle == 1) {
     //Middle handle was moved
     myVideo.currentTime = values[1];

     }
     else if (handle == 2) {
     //Right handle was moved
     $rootScope.model[$scope.index].endTime = values[2];
     }
});
```

Listing 5. noUiSlider, onChange event.

As seen in listing 5. When the onChange event is triggered there are three possible outcomes. By moving the left handle the starting time value will be set to the same value the left handle is set to. If the middle handle was moved than the current time of the selected clip will change. If the right handle moved than the ending time value will be set to the same value the right handle is set to. $scope.index is a variable representing the index of the selected or current playing clip.

Assuming a user would like to rearrange the order of the clips in the Timeline, it can be done by a simple drag and drop using jQuery Draggable and Droppable. While a clip is being dragged, there will be a green border around the clip, indicating the user that it is possible to place the clip in that position, but if the dragged clip will be over another clip then the border will change color to red, indicating that this position in the Timeline is not allowed as seen in figure 9. If the clip will be dropped while the border color is red then the action will revert and the clip will move back to its previous position.

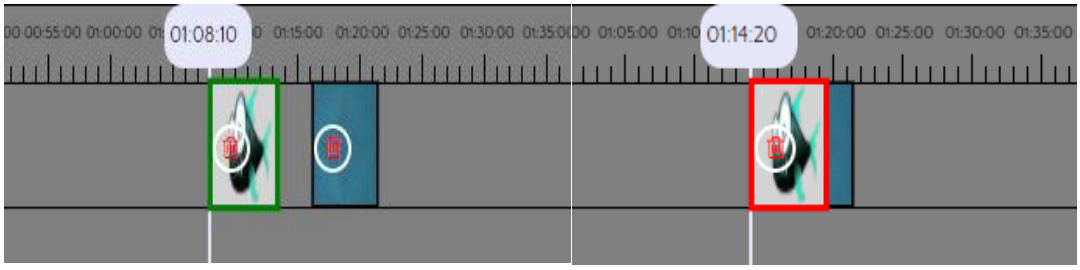

Figure 9: Edit view - Buffering a video that was added by the user to the Timeline

This was achieved using jQuery Draggable event called drag, the drag event is triggered when the user starts dragging a clip. Once the drag event is triggered, an if statement calls the "itemsOverlapCheck" function that checks whether the dragged clip is overlapping another clip or not. As long as the user is dragging the clip, the function will keep executing until the dragging will end. The "itemsOverlapCheck" function returns a Boolean, based on the result the appropriate classes are applied. The code snippet in listing 6 shows how this is done.

```
$(". draggableMedia "). draggable ({
     axis: "x",
     containment: "#visualTrack",
     scroll: true,
     scrollSensitivity: 100,
     snap: ".mediaItem",
     snapMode: "outer"

(...)

drag: function(e, ui){
// if statement calls a function that returns a Boolean if a clip
//is allowed to be dropped or not.

if (TimelineFuncService.itemsOverlapCheck(time, itemEndTime,id,
$rootScope.model)) {
     //if not allowed than revert option will change to true and
     //change class to redAlert.

     $( ".draggableMedia" ).draggable( "option", "revert", true );
```

```
        $('#VisualModel-'+id).addClass('redAlret').re-
        moveClass('greenAlret');
}else {
        //if allowed than cancel the revert option and change class to
        //greenAlert.

        $( ".draggable2" ).draggable( "option", "revert", false );
        $('#VisualModel-'+id).addClass('greenAlret').re-
        moveClass('redAlret');
}
},

stop: function(e, ui){
(...)
        //When dropped. The dragged model item's position property
        //receives new value.

        var index = ui.helper[0].children[0].id.replace(/\D/g, '');
        $rootScope.model[index].position = ui.position.left;

(...)
}
});
```

Listing 6. jQuery UI Draggable. drag event

Since one of the requirements for this project is to give the user full control over the clip's positioning in the Timeline sequence, therefore allows the user to choose the exact time that the clip should start playing. The position property allows the data model to store the position of each clip in the Timeline so when leaving the project or refreshing the page, the position of the clips will be saved and rendered when the project is reloaded. This is done by saving the model data in localStorage. That allows the user to have full control over his Timeline clips in regard to their arrangement. Once the user rearranged the sequence and dropped a clip in the desired position in the sequence, a jQuery UI Draggable triggered event called stop is executed and the position the clip was dropped at is available. As seen in listing 6, the new position received from the stop event replace the current position value of the model item.

The position property refers to the left side of the clip in the Timeline, the left side of a clip in the Timeline also represent the time the clip should start playing. The "itemsOverlapCheck" function is using the position, start and end time properties to calculate the width and location of each clip in the Timeline sequence and compare it with the current position of the clip that is being dragged. As shown in listing 7 below, there are two if statements inside a forEach function, the forEach function compares every model item in the Timeline sequence with the dragged clip to see if they are overlapping each other.

The if statements checks every possibility for overlapping clips. As an example it can be done by checking if the dragged clip's position, left side of the clip, is in a smaller position than the Timeline clip's position and that the right side of the clip, position plus duration (end time minus start time), is bigger than the Timeline clip's position. That means that the dragged clip is overlapping a clip in the Timeline sequence, this will cause the redAlert class to change the border color of the dragged clip to red and will revert if dropped.

```
itemsOverlapCheck: function(draggedItemStart, draggedItemEnd,
draggedItemId, model){

var pass = false;
model.forEach(function(item, index) {
  if (draggedItemId != index) {

     var start = item.position;
     var end = start + (item.endTime - item.startTime);

     //Dragged item need to end before another media item starts
     //and need to start after another item ends

     if (draggedItemStart <= start && start <= draggedItemEnd ||
     draggedItemStart <= end && end <= draggedItemEnd) {

          pass = true;
     }

     if (start <= draggedItemStart && draggedItemStart <= end  ||
     start <= draggedItemEnd && draggedItemEnd <= end) {
          pass = true;
     }
  }
})
//If true, clips are overlapping
return pass;
}
```

Listing 7. itemsOverlapCheck function.

The index property will change automatically when the user change the order of the sequence by rearranging clips in the Timeline using drag and drop. There are some properties that the user cannot modify, those are the URL, title and duration properties.

4.4.3 Overlay assets

The overlay assets are divided into two different overlay items, Images and textboxes. The user, as mentioned in chapter 4.2, has a private repository in Amazon S3 and in his repository there is a bucket that stores his multimedia assets and another bucket for his overlay assets. Each clip in the Timeline also acts as a slide in a presentation editor where the user is able to drag images and textboxes on top of it. Once a user dropped an overlay asset on the Preview component, the model item will receive a new property called Overlay. That property, as mention in subchapter 4.3.2 in table 1, is an array of JavaScript objects representing the overlaid items added. The following list shows the properties of the overlaid items.

- Icon - object, represent the icon below the Slider, see figure 11
  - URL - string, same image URL as the overlaid asset
  - SliderLocation - integer, the position of the icon relative to the Slider left handle represent the appear time property, see figure 11
- Image - object, represent the overlaid asset
  - Width - integer, asset's width
  - Height - integer, asset's height
  - URL - string, overlaid asset image URL, in case of a textbox the image opacity is 0.
  - Text - object, only if textbox was used as an overlay asset
    - Font - string
    - Font size - integer
    - Bold - boolean
    - Italic - boolean
    - Underline - boolean
    - ParsedValue - string, stores the text in a parsed way to be read by the draw canvas function.
- Configuration
  - Appear-time - integer, by default set to the clip's time when the overlaid asset was dropped
  - Disappear-time - integer, by default set to three seconds after the clip's time when the overlaid asset was dropped
  - Position_x - integer, position of the overlaid asset on the x axis
  - Position_y - integer, position of the overlaid asset on the y axis

Most of the overlaid asset properties can be modified by the user, the only property that cannot be changed is the URL.

For the user to add or adjust the location of an overlay asset, a simple drag and drop is required. By clicking on the overlay assets tab in the vertical menu the overlay asset folder will open. Once the user start dragging an overlay asset, using jQuery UI Draggable widget, the Preview will highlight to signal the user what are the borders where he can drop or move around the overlaid asset as seen in figure 10. By moving around an overlay asset the position_x and position_y properties are being updated using jQuery UI Draggable and Droppable events. Unlike WeVideo there are no preconfigured positions for placing the asset which gives the user full control of his creation.

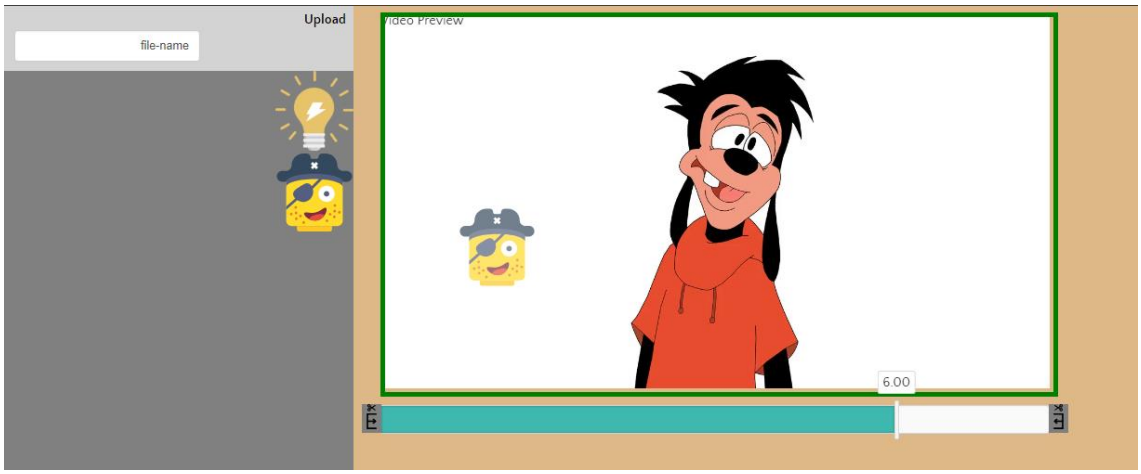

Figure 10: Edit view - overlay assets folder is opened, an asset is being dragged and the Preview border is marked as the container for the allowed drop location

Once an overlay asset was dropped, as seen below in figure 11, the overlaid asset can be resized using each of the marked four corners of the asset. The resizing functionality was built using jQuery UI Resizable. The Resizable widget, just like the Draggable widget, has a stop event which is triggered when the resize action has stopped. Through this event the new width and height properties, set by the user, are being set as the new values for the overlaid asset width and height properties. Mouse hover over the overlaid asset will present the user with the option of removing the asset by clicking on the delete button.

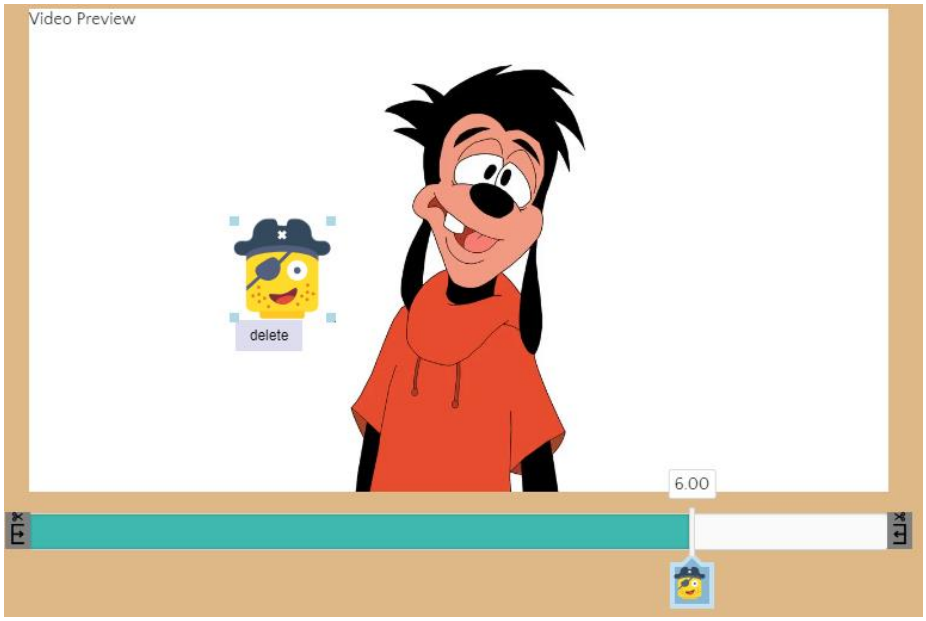

Figure 11: Preview - mouse hover over an overlaid asset

Once the user dropped an overlay asset on to the preview, the overlaid asset is set to appear starting from the time it was dropped. Assuming that the current clip is at the sixth second when the user added an overlay asset, than the overlay asset is set by default to appear from the sixth second of the clip and a new icon will appear below the Slider in order to indicate when the overlaid asset should appear. The overlaid asset has a preconfigured ending time of three seconds after it appears. If the user clicks on the Slider's overlay icon, as seen in figure 12, the Slider will change color and will now control the timing of the overlay asset giving the user full control on when the icon should appear and when it should disappear.

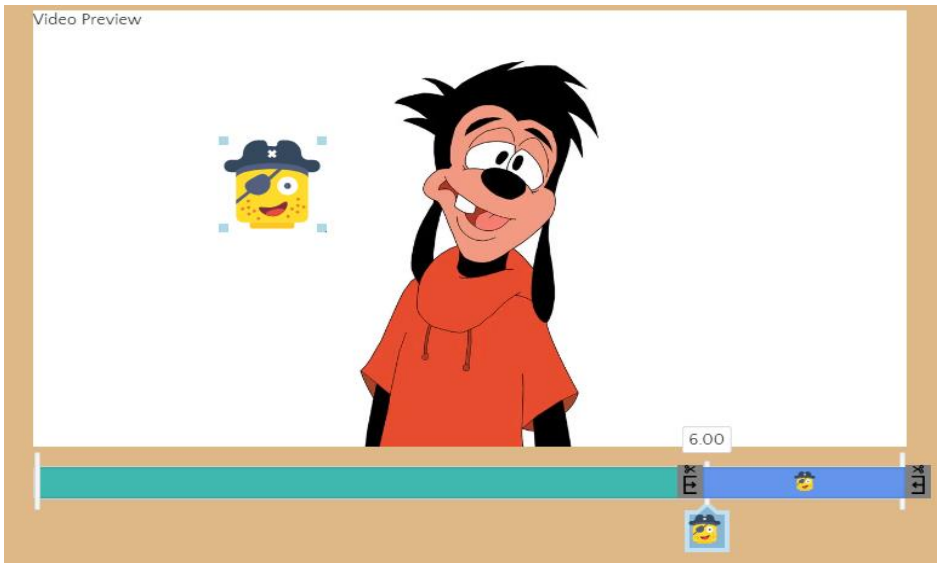

Figure 12: Preview - the Slider is controlling the appearance duration of the overlaid asset

By dragging the Slider's handles the user can define when overlay asset should appear and when it should disappear. Just like trimming a clip, the left handle defines when the overlay asset should appear and the right handle defines when it should disappear. This allows the user to modify the overlaid asset appear and disappear properties. The sliderLocation property is modified when changing the appearance starting time by dragging the Slider's left handle. In case there is more than one overlaid items that start at the same time, as seen in figure 13, the icon below the Slider will change to a number representing how many overlaid assets are there and by hovering the mouse over the icon, the list of overlaid items will appear and allow the user to choose which one he would like to edit by clicking on the corresponding icon. The user can also click on an overlaid item and by that select it and the Slider will be adjusting to control that overlaid asset appearance time.

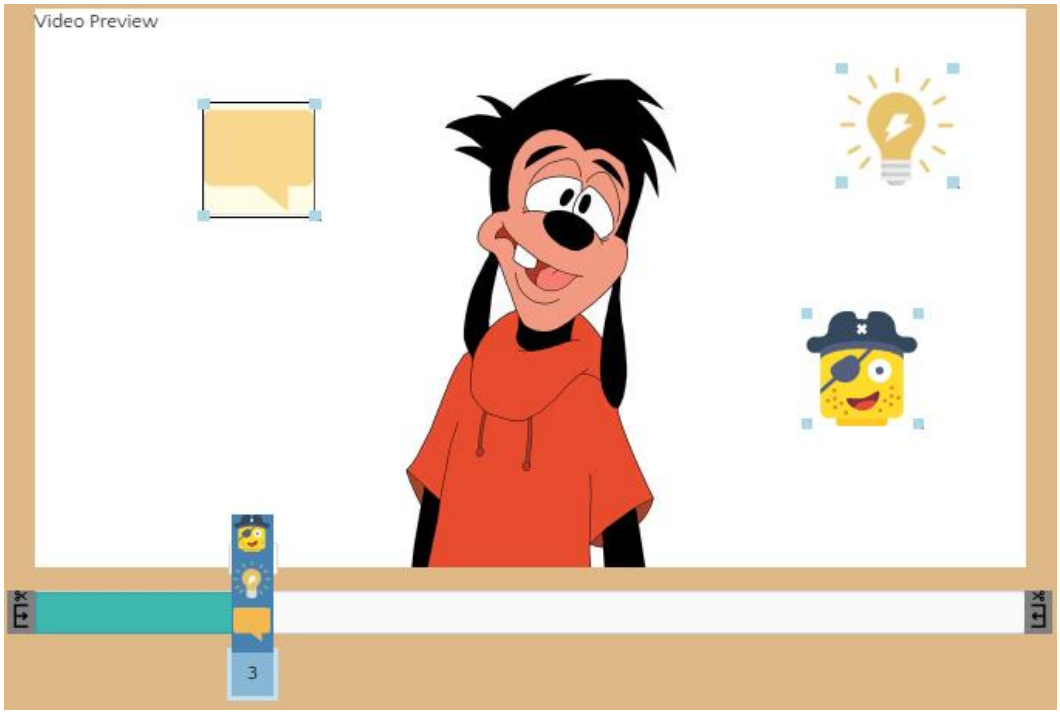

Figure 13: Preview - more than one overlay items start at the same time

When a user drags an overlay asset and drops it on the Preview component, three new elements are being created, image, button and a div. The image element receives the source and title properties from the asset along with preconfigured width and height properties. The button is the delete button shown earlier in figure 11**,** the div is appending both the image and button and then the div in being added to the DOM by appending it to the document's body. The div receives also the position that the overlay asset was dropped at using jQuery UI Draggable stop event. The div containing the overlaid asset

will appear only when the sequence is not playing and when the clip that contains the overlaid asset will be selected and shown in the Preview. When the sequence is playing, the div containing the overlaid asset will be hidden but using its properties, the canvas will draw the asset based on the appearance time and at the correct height, width and position on the Preview. The div containing the asset is created in order to allow the user to modify the way the overlaid asset will be displayed while the sequence is playing.

Since the goal of this project is to create a hybrid of video editor and a presentation editor, the overlaid textbox should allow the user to edit text with as similar features as possible to that of a presentation editor. Therefore the overlaid textbox is created using a JavaScript library called Quill.js, which is a free open source rich text editor built for the modern web. Its modular architecture allows Quill's behavior and functionality to be customized [21]. To configure Quill, it is required to have a container where the editor will be appended to, either a CSS selector or a DOM object can be configured as a container. As mentioned earlier in this subchapter the overlaid item is created with a div that contains an image and a delete button. If the overlaid item is a textbox then the image will act as the textbox visual background and the div will be configured as the container where Quill will be appended to.

```
// Add fonts to whitelist
var Font = Quill.import('formats/font');

// Listing the fonts wanted
Font.whitelist = ['ariel', 'verdana'     ];
Quill.register(Font, true);


var quill = new Quill('#editor-container', {
  modules: {
     toolbar: '#toolbar-container'
  },
  theme: 'snow'
});
```

Listing 8. Quill.js basic configuration.

Listing 8 shows the most basic way to configure the Quill editor. Since this project requires even more customization, the toolbar was created manually in HTML, and the DOM element was passed as the toolbar into Quill as seen in listing 9 below.

```
--------------------HTML---------------------
<div id="toolbar">
     <select class="ql-size">
          <option selected>Font-Size</option>
          <option value="8.0px">8</option>
          <option value="9.0px">9</option>
          <option value="10.0px">10</option>
          <option value="12.0px">12</option>
          <option value="14.0px">14</option>
          <option value="16.0px">16</option>
          <option value="18.0px">18</option>
          <option value="20.0px">20</option>
          <option value="24.0px">24</option>
     </select>
     <select class="ql-font" id="sel-font">
          <option selected>Font</option>
          <option value="verdana">Verdana</option>
          <option value="times">Times</option>
          <option value="roboto">Roboto</option>
          <option value="arial">Arial</option>
     </select>
     <button class="ql-bold"></button>
     <button class="ql-italic"></button>
     <button class="ql-underline"></button>
</div>
-----------------JavaScript-------------------
// Add fonts to whitelist
var Font = Quill.import('formats/font');
Font.whitelist = ['arial', 'roboto', 'times', 'verdana'];
Quill.register(Font, true);

var quill = new Quill('#editor', {
     modules: {
        toolbar: '#toolbar'
     },
     theme: 'snow'
});
```

Listing 9. Quill.js toolbar custom configuration.

By customizing the toolbar as seen in listing 8, the user can choose a different font, font-size and other basic options such as bold, italic and underline. For the beta version of Piranha these settings are sufficient.

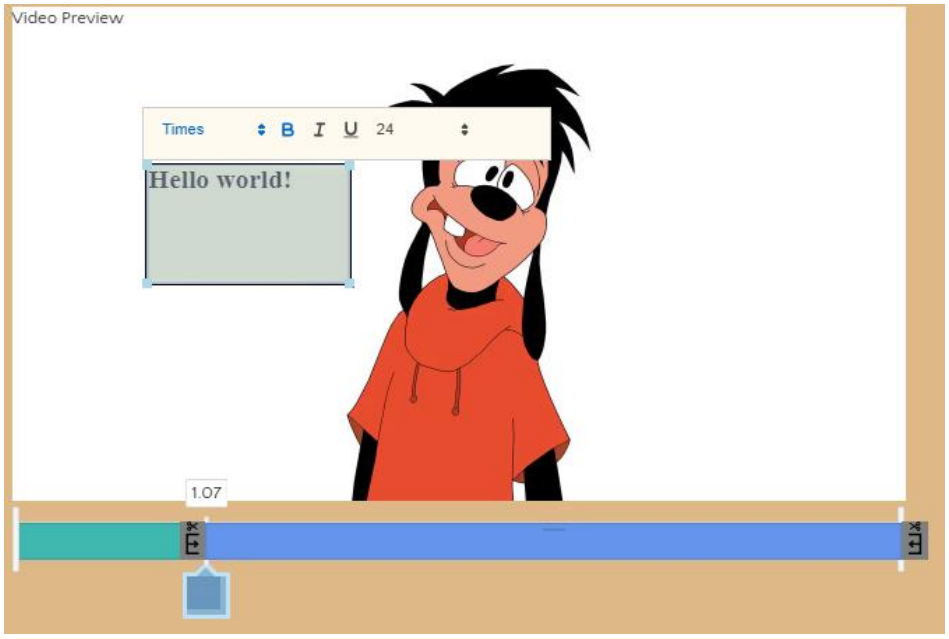

Figure 14: Preview - overlaid textbox with Quill configuration, shown in listing 9

Figure 14 shows the result of the quill configuration mentioned in listing 9, the user can resize, drag, delete, select appearance time and edit the text. When a user clicks on the toolbar an onClick event is triggered, checking what element of the toolbar was selected by checking the class of that element, each element has a unique class to be identified by. The new selected value will be set to the appropriate overlay item property. Quill has an event called text-change that monitors the text editor which allows the overlay item image.text.parsedValue property to update each time the content of the Quill editor have changed.

5 Discussion

While developing Piranha, several users were testing the application thought the development phase, giving valuable pointers on how to improve the user experience and what should be improved. This section will describe the changes made to the application regarding the comment and feedbacks received.

In an early stage of the development, there was another view in charge of editing single clips. When hovering with the mouse over a clip in the Timeline sequence two icons were introduced, trash bin icon and a pen icon representing removing a clip from the Timeline sequence and routing to the “Single edit view”. In the single edit view the user could perform actions such as changing the clip’s starting and ending time, add overlay images and textboxes and adding an audio background for the clip. At that time the Timeline only had one track for images and videos.

The feedback received after the test users tried working with the application, were that the application is too complicated and perhaps it would be easier to merge both views into one. While watching the test users struggle with how to work with the application, an idea of indicating the user by highlighting the allowed locations for the assets to be dropped came up. This valuable feedbacks led to the current version of the application. In future stages of the development an interactive popup guide will be created and will appear when the application starts and visually explain the user how to use the application.

In later stage of the development the test users were satisfied with the updated version based on their feedbacks and the only addition comment made was about the visuals. Since this is a prototype the visuals are less important at this stage, in future development modifications to the visuals will be made.

6 Conclusion

The goal of this project was to build an online video and presentation hybrid editor using modern programming languages. This was achieved using JavaScript, Angular.js, jQuery, Node.js and express.js. Using Angular.js MVC approach the creation of the model object was made, when a user exports the sequence the model object properties then pass to three API's which create the exported video that can be shared or downloaded. Nowadays cloud transcoding services provide file editing features for video, audio and images with reasonable processing times. The advantages of this architecture are that the application is highly scalable since the APIs implemented can handle a high amount of volume and the Piranha server could handle requests from thousands of concurrent users. The architecture strips the developer of the tedious work and allows focusing on creative development of new features. The APIs used, and the backend operations were not described since they are out of the scope of this study.

The solution is not at an enterprise level solution. The software requirements were met, and the application has a reliable, scalable base for further development. Future work will include improving the design of the application, adding more features such as screen recording, transitions between the clips, preconfigured animation library, an interactive guide for new users, allowing CRUD operations for the assets libraries such as creating folders, renaming files and deleting files. The above mentioned features are among those improvements added to Piranha in future development.

References

1. Ellis, Ryann K. A Field Guide To Learning Management Systems (LMS). [Internet]. [Cited 12 August 2018]. Available from: http://cgit.nutn.edu.tw:8080/cgit/PPTDL/hclin_091104025632.PDF
2. Roels R, Baeten Y, Signer B. An Interactive Data Visualisation Approach for Next Generation Presentation Tools - Towards Rich Presentation-based Data Exploration and Storytelling. Proceedings of the 8th International Conference on Computer Supported Education [Internet]. [Cited 12 August 2018]. Available from: https://wise.vub.ac.be/sites/default/files/publications/csedu2016.pdf
3. Media editor for non-linear editing system [Internet]. US patent office; [cited 12 August 2018]. Available from: https://www.google.com/patents/US6154600
4. Overview of the new features and enhancements in Premiere Pro [Internet]. [Cited 12 August 2018]. Available from: https://helpx.adobe.com/premiere-pro/using/whats-new-7-0-1.html
5. Mersand S. Product Review: PowToon, Tech Learning. [Internet]. [Cited 12 August 2018]. Available from: http://www.techlearning.com/news/0002/product-reviewpowtoon/63310
6. Perez S. Now Everyone Can Make Marketing Videos: PowToon Launches DIY Presentation Tool [Internet]. [Cited 12 August 2018]. Available from: https://techcrunch.com/2012/06/26/now-everyone-can-makemarketing-videos-powtoon-launches-diy-presentation-tool/
7. Eloff E, Torstensson, D. An Investigation into the Applicability of Node.js as a Platform for Web Service. Institutionen för datavetenskap. 2012
8. Express - Node.js web application framework [Internet]. [Cited 12 August 2018]. Available from: https://expressjs.com
9. Kwangwon Sub, Samsung Electronics, Kaist Sukyoung Ryo, Kaist. [Internet]. [Cited 12 August 2018]. Available from: https://dl.acm.org/citation.cfm?id=3106741
10. What Is AngularJS? [Internet]. [Cited 12 August 2018]. Available from: https://docs.angularjs.org/guide/intoduction
11. What is jQuery? [Internet]. [Cited 12 August 2018]. Available from: https://jquery.com/
12. MVC architecture, the theory behind Model View Controller. [Internet]. [Cited 12 August 2018]. Available from: https://developer.mozilla.org/en-

US/docs/Web/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture

13. Media editor for non-linear editing system. US patent office. [Internet]. [Cited 12 August 2018]. Available from: https://www.google.com/patents/US6154600
14. Roels R, Baeten Y, Signer B. An Interactive Data Visualization Approach for Next Generation Presentation Tools - Towards Rich Presentation-based Data Exploration and Storytelling. Proceedings of the 8th International Conference on Computer Supported Education [Internet]. [Cited 12 August 2018]. Available from: https://wise.vub.ac.be/sites/default/files/publications/csedu2016.pdf
15. jQuery, Deferred Object. [Internet]. [Cited 12 August 2018]. Available from: https://api.jquery.com/category/deferred-object/
16. jQuery UI, About jQuery UI. [Internet]. [Cited 12 August 2018]. Available from: https://jqueryui.com/about
17. Performance of HTTP video streaming under different network conditions. [Internet]. [Cited 12 August 2018]. Available from: https://core.ac.uk/download/pdf/81853112.pdf
18. HTML Audio/Video DOM loadedmetadata Event. [Internet]. [Cited 12 August 2018]. Available from: https://www.w3schools.com/tags/av_event_loadedmetadata.asp
19. jQuery UI, Resizable [Internet]. [Cited 12 August 2018]. Available from: https://jqueryui.com/resizable/
20. JavaScript Range Slider, noUiSlider. [Internet]. [Cited 12 August 2018]. Available from: https://refreshless.com/nouislider/
21. Quill.js, API Driven Design. [Internet]. [Cited 12 August 2018]. Available from: https://quilljs.com/guides/why-quill