Asaf Golan

Online video presentation editor

Helsinki Metropolia University of Applied Sciences

Bachelor of engineering

Media engineering

Bachelor thesis

February 13 2017

Author(s) Title Number of Pages Date	Asaf Golan Online video presentation editor 32 pages 13 February 2017
Degree	Bachelor of Engineering
Degree Programme	Media Engineering
Specialisation option	Web Development
Instructor(s)	Olli Alm, Senior 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.

The study focuses on a tool for creating educational videos and informative videos. Implementing cloud services with NodeJS and ExpressJS the goal is to build a prototype for editing and creating video presentations.

Keywords	video presentation editor, NodeJS, ExpressJS, cloud services.

Table of content

Abbreviations and Terms

API	Application Programming Interface
AVC	Audio and Video Codecs
CRUD	create read update delete, are the four basic functions of persistent storage.
DOS	denial of service. Is an attack on a service by overloading it and making it unavailable.
GNU	GNU's Not Unix, Linux OS. open source operating system
GPL	GNU’s General Public License
HD	high definition any video image with considerably more than 480 horizontal lines (North America) or 576 horizontal lines (Europe)
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
LMS	learning management system
MVC	Model View Controller
MPEG4	Motion Picture Experts Group Layer-4 Video
MMS	Multimedia Messaging Service
NoSQL	Not only SQL
NLE	Non-linear editing
S3	Simple Storage Service
SD	standard definition less than 480 horizontal lines in an image(North America) or 576 lines (Europe)

SPA	single page application
SQL	Structured Query Language
UI	user interface

1 Introduction

This thesis will discuss online video presentation editors. Such an editor prototype 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 myself and other developers. The scope of my work is to prototype the application and check if the prototype could be made into a viable product. Other team members are responsible for front end design. The scope of this work is limited mainly to architecture.

Datafisher provides its customers with an LMS (learning management system). Learning management system is a software for the 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]. Since those are not very specific requirements, the customer introduced WeVideo and Powtoon web applications as example.

WeVideo, a video editing cloud service, and Powtoon. Powtoon is a web-based animation software that allows users to create animated presentations using pre-created animation objects. Both applications are described in sub-chapters 2.3 and help in defining our software requirements. The video presentation software is a hybrid concept that combines the features from both presentation and video editing software. The aspiration 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.

Piranhas output is a video. It is a product of media assets such as audio, video and images, called clips. Clips can be trimmed and concatenated 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 can also input informative data and enable similarly to presentation software as the common MS PowerPoint [2].

Chapter 2 introduces the background for video editing and presentation editing as well an analysis of WeVideo and Powtoon in terms of software features, functionality and pricing. Subchapter 2.4 defines what the video presentation hybrid concept is and also focuses on the basic software requirements for the prototype. Chapter 3 presents the Piranha prototype, architecture and how it meets the software requirements. Subchapters 3.7 and 3.8 discuss the security and scalability of the prototype architecture. Chapter 4 will specify the operating costs for the prototype. Chapter 5 will show performance for manual testing of the prototype and summarize the results of the test case in terms of cost and performance.

2 Online video presentation editor

2.1 Video editing background

The following chapter introduces the basic concepts of video editing or more specifically the basics of digital video processing. It will define video editing and briefly describe the standard requirements for a modern video editing software as well as terms needed to understand the practical work done.

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 are Video editing software 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 colour manipulation, titling and visual effects, as well as tools for editing and mixing audio synchronized with the video image sequence [5].

Due to the size of raw video data, there is a need for data compression. Data compression is done by devices or software, codec, that utilizes some compression algorithms. Compressing the file is called encoding and decompression decoding [6]. One of the most popular codecs is x264: A free library under GNU GPL-licensed for encoding video streams into the H.264/MPEG-4 AVC compression format [7].

The compression format defines the type of the container holding video or audio file. The container contains the data and metadata for the media file. Popular video formats are MP4, AVI, and WMV [6] .

Transcoding is the conversion of one encoding to another, like movie raw data files. The most popular use of transcoding in the smart mobile era is MMS. A protocol for exchanging messages between mobile devices [8]. For example, a digital photo taken with mobile device produces a high-quality image of usually at least 640x480 pixels. Sent to another device, this high-resolution image might be transcoded to a lower resolution image with fewer colors in order to better fit the target device's screen size and color limitations. Reduction of size and color improves the user experience on the target device. MMS is relevant to the project for the user should not be constrained by the frame size of the assets, clips, desired for editing. FFmpeg is a popular Open source framework for video and audio transcoding and handling. It includes libraries like libavcodec, libavformat, libavfilter and functionalities such encode, decode, mux, stream and play media content. FFmpeg provides a simple command line interface to access functionalities. It is able to compile on many different platforms such as Linux, Windows, Mac OSX, Android, and iOS [9]. FFmpeg is important to this study since Transloadit API which is integrated into our application rely on FFmpeg for video editing. Transloadit is described in detail in chapter 4.

2.2 Presentation software background

This chapter defines presentation software and how in a presentation text and graphic elements are integrated into the slides. It is important to the logical structure since one of the inspiration sources provided by Datafisher, Powtoon, is designed as a presentation editor.

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.

2.3 Requirements for Piranha application

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

The following section is an analysis WeVideo and Powtoon in terms of software features and pricing. The analysis is then used to refine the software requirements and the define the hybrid approach described in subchapter 2.4, and to provide a pricing benchmark for this practical work.

2.3.1 WeVideo

WeVideo is an easy-to-use cloud-based video editor. Desktop native video editing applications such as Adobe Premiere are the basis for video editing software, hence it is designed according to the NLE approach. Built as SPA (single page application) WeVideo has an intuitive user interface consists of 4 components:

- Assets - user can view all images, audio and video files available for editing.
- Timeline - video objects container. The container's video order determines the chronological order of the final export file. Timeline object is a single video.
- Preview - generate a preview for user view prior to export.
- Single video edit dialog - additional manipulation options set for specific timeline object.

Videos that are in the timeline container, such as clips in the sequence, are subjected to selection and combining into a sequence and the addition of accompanying sound effects and audio to ultimately create a finished commercial, promo, or snipe.

Single video manipulations:

- Text and watermark overlay - set by end user. The text is overlaid given, indicating start and end times and location parameters. However, text location is limited to two possibilities.
- Transformations - rotating and scaling current frame by direction and multiplier.
- Adjusting frame size - pad a video with black bars to fit export resolution.
- Audio/video - fade in fade out effect.

Pricing, Monthly plans [10]:

Table 1. Wevideo monthly payment plans

Plan name	Cost /$
power	19.98 / hour of video duration
unlimited	15.99 /month unlimited video duration

2.3.2 Powtoon

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 [11]. 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. 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 [12].

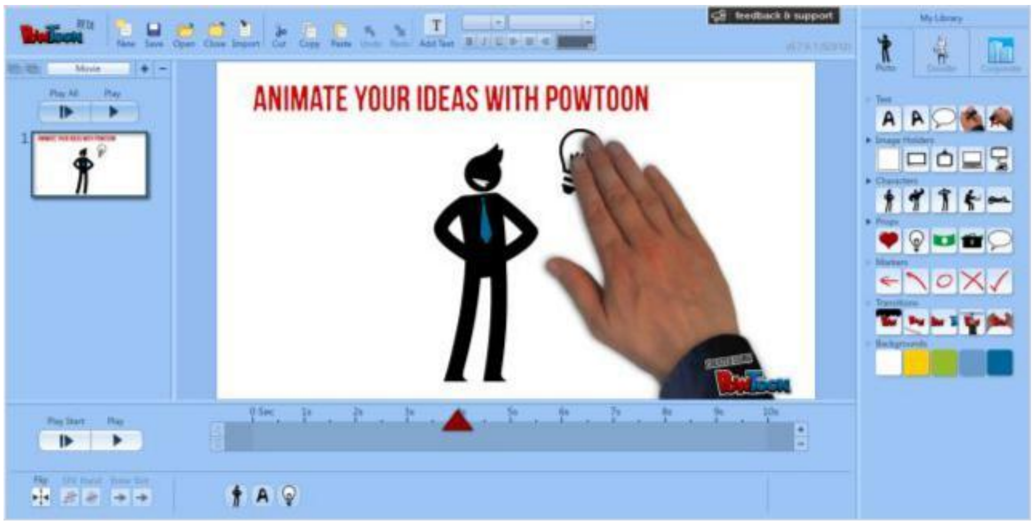


Figure 1 . Powtoon interface

Pricing, monthly plans [10]:

Table 2 . Powtoon monthly payment plans

	Monthly cost /$	Presentation time limit	Total presentation limit
Business	179	00:01:00	unlimited
Pro	89	00:00:15	unlimited

2.4 Video presentation hybrid concept

The designated application aims to provide educational instructors with a tool for creating educational videos. Piranha can be referred to as a hybrid application since it combines the slideshow and the NLE approaches. In an NLE application a sequence is made from video clips as in presentation editor where slides are creating a sequence called a slideshow, ergo the timeline concept is common to both.

Clips inside of a sequence could be treated as another video sequence and they could be cropped, trimmed and manipulated and as a slide where it could be overlaid with clipart, text and other pre-defined animations and allow transition between slides from an animation or a clipart library of pre-configured objects. The animation, clipart library is unique to the presentation software. As an amateur tool for non-professional users, Piranha is designated to use by subscribers to produce educational videos that are, according to the client, usually less than 15 minutes long.

Video manipulation required features:

The following requirements are the building blocks for piranha and are derived from the analysis of WeVideo and Powtoon.

1. Transcoding different video assets uploaded by user video to the same format - assuming an end user could upload assets for editing. Users should not be limited to a specific video format and should be able to edit multiple types of formats.
2. Transcoding all videos to same frame size - no constraints for assets frame size so when creating a new sequence there are 2 strategies to chose from:
    a. Padding - letterboxing the frame with black bars to fit the monitor/preview is described in more detail in subchapter 4.5.1.
    b. Resizing - stretch the frame width and size to meet monitor resolution.
3. Cropping video files by time - trimming a video sequence into a shorter sequence by defining endpoints.
4. Concatenating videos - attaching clips creating a new video sequence.
5. Overlaying text caption and images - by time and x, y coordinates and opacity. overlaying animation is the product of overlaying images or text by time, x, y coordinates, and opacity. In a linear manner, similar to creating a cartoon, text and images are placed in different coordinates and times. Time intervals between the different positioning of an object should be short to create motion like movement. The animation is preconfigured, however, the text content, transition, duration, start, end times and location should be dynamic.
6. Replace/place an audio track for a video file - a user should be able to overlay sound, audio file, over a video clip in the sequence.
7. Preview sequence to end user is outside the scope of this study and will not be discussed.

3 Piranha

Piranha is an online presentation video editor. The user can choose video clips from a storage and edit them to a sequence. Clips could be, trimmed by time, padded to adapt frame size, concatenated and overlaid with audio. The video file could also be overlaid with animations from an animation library that contains preconfigured objects. The end user for piranha is an amateur video editor. Each user has his own files stored on the cloud those files are available for editing when the user logs into the application.

After manipulating the files in the preview user could export a video and the result is a video file made from the clips chosen by the user and overlaid with animations from the animation library.

The application overview tells about the basic use case and presents UI mockup that explains how a new video sequence is edited and the external APIs, once a user chose to export the sequence, are used to process the video clips in the new sequence that results in an edited video file.

3.1 Application overview

Assuming a subscriber to the application logs in and chooses to create a new project user is routed to the main edit view. The main edit view is divided into 3 UI components as shown in figure 2:

Assets - each user has media assets stored in Amazon S3. Currently, the application only supports audio and video with no constraints on formats or frame size. Assets URLs are fetched to the client-side from the server upon getting a response from amazon API. Thumbnails of the assets are loaded to the Assets UI component.

Timeline - timeline contains the current sequence of clips subjected to editing. Assets from the Assets UI component can be dragged, dropped and rearranged into the Timeline UI component.

Preview - is the top right corner. The preview is basically an HTML5 video Player loaded with the first video in the timeline. The preview will play the sequence of clips to the user according to the order of assets in the timeline.

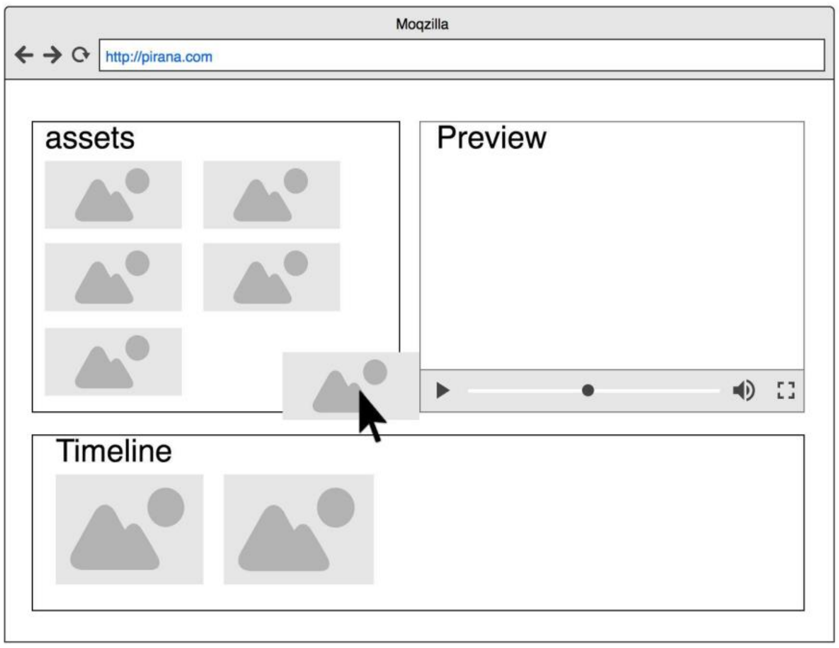


Figure 2 . drag and drop main edit view

Mouse pointer hovering over a timeline asset as shown in figure 3 introduces the user with the option to delete an item from the timeline or extended edit functionality for a specific item marked by the garbage icon and the pen icon in the figure. Clicking on the pen button will open a single item edit view as shown in figure 4.

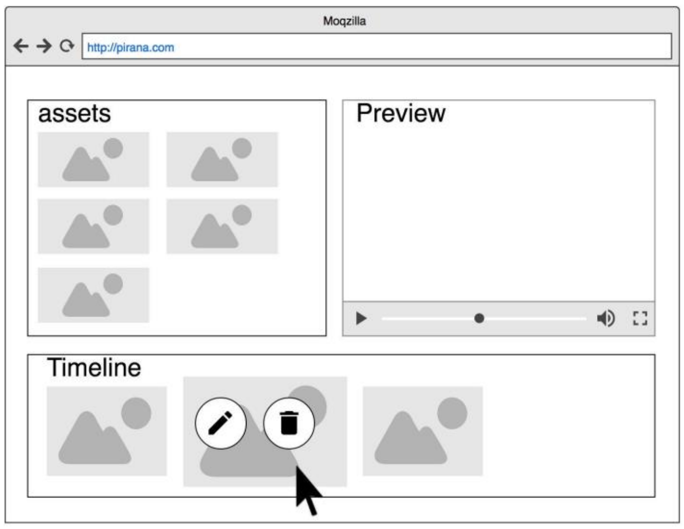


Figure 3 : edit/delete button main edit view

The single edit view is divided into 3 UI components:

Animation library UI component - animation library is a feature of presentation software and is similar to Powtoon. The animation library is outside the scope of this study but it is the reason for implementing requirement 5 from section 2.4, overlay text and images by location time and opacity parameters. The animation library is a collection of pre-configured animation objects which could be dragged into the preview where the drop location is the location of the animation when previewing to the user and in the final video file.

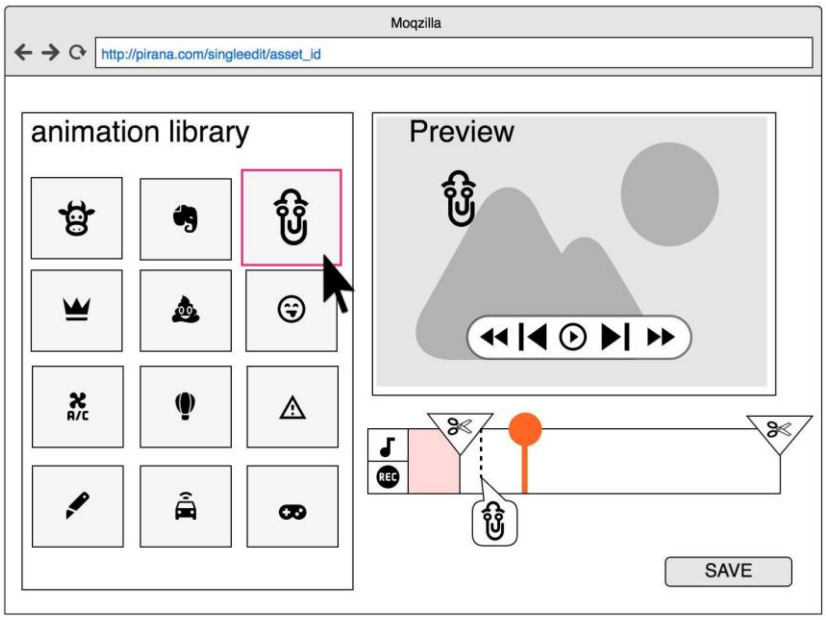


Figure 4 . single edit view

Toolbar UI component - consists of several components. The toolbar is fundamentally a progress bar for the item previewed. Shown in table 3 is the legend for the toolbar. The toolbar allows the user cropping the video clip by time endpoints, replacing the audio track of a video with a custom one, audio overlay and also shows the user the current progress of the video clip played on the preview UI component and shows the appearance time for the animation object.

Table 3 . Toolbar legend

📍	This pointer is aligned with the asset, item or clip, previewed in the edit preview and progresses along the bar with time.
✂	Handlebars for setting video start and end time. Dragging those across the progress bar will trim the video according to the end point the bar is dragged from.
🟥	The colored area covering the progress bar adjacent to the scissors handlebar represents the amount of time trimmed from video and is aligned with the progress bar.
💬	An example of how a dragged animation object would be placed on the toolbar after placing it on the preview screen. The dotted line above the bubble shown in figure 4 is the time the object was placed and will preview as the start time for the object appearance.
♩	Voice over button allows the user to select an alternative track to replace the current video audio.
REC	Voice over button allows the user to record an alternative track to replace the current video audio.

Preview UI component means that the preview shows the video clip chosen for editing, all the manipulation applied by the toolbar and the animation overlaid from the animation library. After a user has edited a new sequence, it needs to be exported to result in a new video. A data model configured on the client side for each timeline clip in the export sequence. The data model for the clips is used to format HTTP request for the external APIs actually doing the processing. The first API called is Zencoder.

Zencoder is a cloud-based video transcoding company that provides comprehensive transcoding solution [23]. A simple API for fast, scalable, high-quality video encoding.

Zencoder functionalities

- Formatting - transcode all files to the same format.
- Set resolution -
  - Width - max width of a video in pixels.
  - Height - max height of a video in pixels.
  - Pad aspect ratio mode - manipulate a video if video size does not match the requested output aspect ratio. As an example, assume that an input file is 1280×720 (widescreen 16:9), but the target output is 640×480 (standard 4:3). Zencoder supports 4 basic approaches: preserve, crop, stretch, pad. Pad is the selected mode for the application. Pad option letterboxes the video with black bars so it will fit desired output resolution.
- Output URL - determines the output URL. The output is a unique URL and the file is stored in Zencoder temporary storage.
- Media files trimming by time parameters - crop a video or an audio by start and
  - end times.

Referring to the video manipulation required features listed in chapter 3
Zencoder meets requirements 1 2a and 3.

Zencoder imports the files which are relevant to be stored on Amazon S3 and crops the files by start and end times. Then it pads the clips with black bars if video frame size does not match the desired frame size, enabling adaptive frame size without image quality loss. Zencoder also transcodes the video files to the same format. Zencoder output is stored in Zencoder temporary storage. The next API call is made Transloadit.

Transloadit is a file uploading & encoding service for web and mobile apps. Founded in 2009 with a wide clientele including Mercedes Benz and khan academy. Transloadit API call is referred as an assembly. The API calls are built as distributed system where robots are performing different steps in the process. Each robot can perform multiple tasks called steps. Steps are asynchronous by default. However, it is possible to use the output of one step as the input of another step without special configuration [24].

Transloadit functionalities:

- Import robot - import all files relevant to an assembly, audio/video.
- Merge robot - merge audio and video files, create a video from an image with an overlaid audio file.
- Concat robot - file concatenation.

Referring to the video manipulation required features listed in chapter 3 Transloadit meets requirements 4 and 6. Transloadit first imports the files from Zencoder then, if defined, overlay audio on top of the separate files and it concatenates all videos to a single file which is stored on Transloadit temporary storage.

The final output from Transloadit URL is the file input in an API call to Cloudinary. Cloudinary provides a cloud-based image and video management solution. It enables users to upload, store, manage, manipulate and deliver images and video for websites and apps [25][26]. Cloudinary is used by more than 120,000 web and mobile application developers at more than 3,000 companies including Condé Nastand Dropbox. Inc. Magazine crowned Cloudinary API as the "gold standard" of multimedia management on the web.[27][28]

Cloudinary functionalities:

- Overlay text by time, x and y coordinates, and opacity.
- Overlay images by time, x and y coordinates, and opacity.

Referring to the video manipulation required features listed in chapter 3 Cloudinary meets requirement 5. Cloudinary imports the output file from Transloadit, overlays animation and text according to the request parameters and stores the output in Cloudinary temporary storage. This is the final output and it is returned to the user as a downloadable link.

Shown in figure 2 is an overview of the architecture. The database stores the users and the relevant storage details on Amazon S3. Each user has his own storage defined and can view only files belonging to him. On the edit view of the application client-side functionality defines a data model with required video manipulations.

When an end user sends a video to export the video URLs are sent to Zencoder for importing formatting and padding. The result from Zencoder is then sent to Transloadit for audio overlay and concatenation and the output from Transloadit is imported to Cloudinary for text and image overlay. The result is eventually returned to the user as a downloadable link.

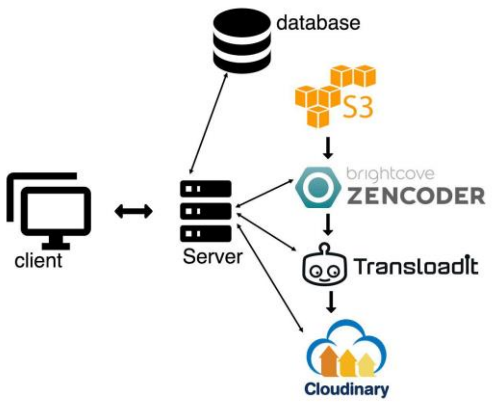


Figure 5 . Architecture overview

Subchapter 3.2 describes the Angular front end functionality. Sub-chapters from 3.3 to 3.5 describe the storage database and the server technologies. Subchapter 3.6 describes in detail the external APIs and their integration as part of the exporting video process.

3.2 Client side

Client-side implements AngularJS as a front end framework. AngularJS is a front end Javascript framework for building modern web applications. Angular Core's functionality is to fill page templates with data from the client Which results in better dynamic page updating. Angular is designed for building MVC (Model View Controller) architecture applications, as shown in figure 6 is MVC diagram [13].

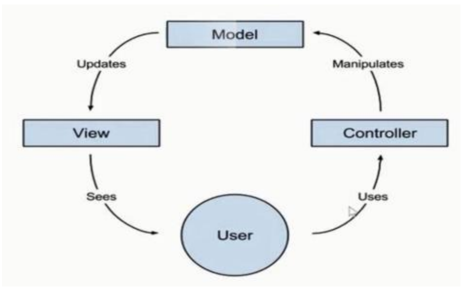


Figure 6 . Angular MVC diagram

The application is designed so that the user configures an angular data model object on the client side for the media assets. Media assets URLs are fetched from Amazon S3 when the page is loaded to an angular scope variable which makes it available to HTML. The end user populates a model referred as an export model because it results in an exported video file. Figure 7 is a screenshot of the main edit view after the page is loaded. Media assets are available on the red, upper left corner. The preview is the top right corner and the timeline is the blue vertical bottom.

Implementing a drag and drop functionality the user can:

- Move assets to the timeline
- Order assets inside of the timeline

Timeline items are assigned to the export model and ordered just as a plain array by indexes. Export model indexes indicate the order which in files are played in the preview and the order of video concatenation.

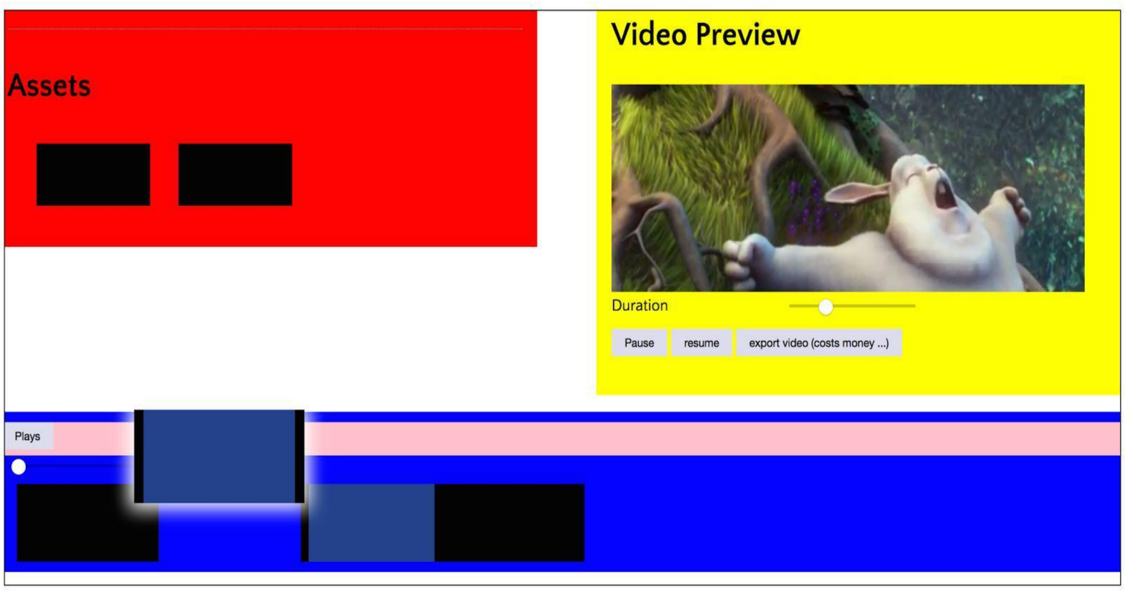


Figure 7 . main edit view drag and drop new item to list

Figure 8 shows a timeline item hovered. When hovered, an edit symbol appears under the thumbnail of the item. Clicking on it will open an extended edit functionality view.

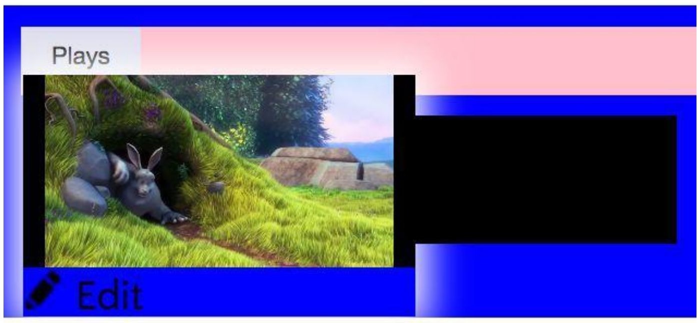


Figure 8 . hovering a timeline item

Figure 9 is the video preview component when a video is paused. There are two range bars for the user setting a start and end time to the clip selected and a duration bar which is aligned with video progress. The start and end time input are dynamically updating the value of the model which in this case is the export model. The item attributes start and end times. Pause and play buttons are straightforward. The export video button is for executing the code that actually manipulates the videos in the back end and outputs the final video.

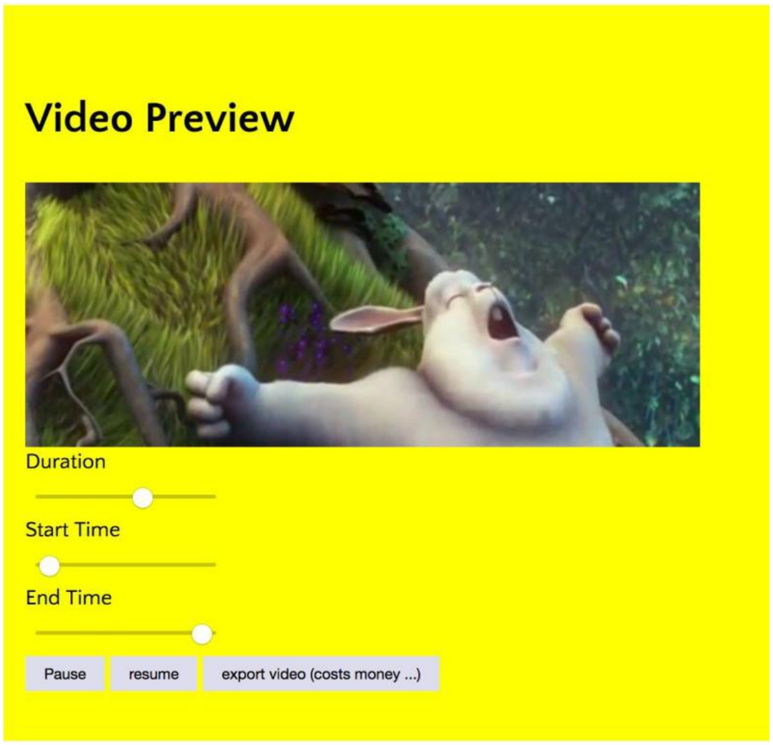
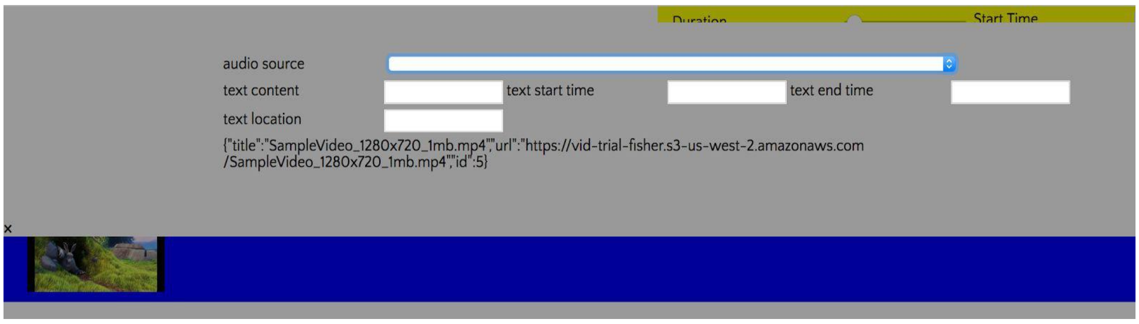


Figure 9 . preview UI component when video is paused

As shown in figure 10, the user interface for editing a single view is far from completed and the implementation includes the basic end-point through which the export model item is configured. UI layout for a single video edit dialog will change substantially with development. User interface functionality will engage the user differently in a similar manner to Powtoon by implementing the option to drag and drop predefined animation objects onto the video from an animation library. For example, there is a text transition from the bottom center of the frame to the center of the frame where a user sets only the appearance time and location but not the animation. All form fields are bound to an export item model.



audio source

text content | text start time | text end time

text location

{"title":"SampleVideo_1280x720_1mb.mp4"","url":"https://vid-trial-fisher.s3-us-west-2.amazonaws.com
/SampleVideo_1280x720_1mb.mp4"","id":5}

Figure 10 . single edit view, add text endpoint

The audio source is a list of audio files also stored on an Amazon bucket. A selected option from the list will set an alternative track option that will replace the current audio track for the video. The audio file, if longer than video duration, will be cropped to fit. Text content is the string value meant to be overlaid on the video. Text start and end time fields take integers and decimals and text location takes x,y values which are decimals as well. The list below shows the configured export model item before executing an export video functionality. UI configurable attributes are noted. The model attributes are:

- `$$hashkey`: a unique identifier, string.
- `Title`: the file title from amazon S3 bucket, string.
- `Url`: the file URL stored in an Amazon S3 bucket, string.
- `startTime`: video start time, decimal value, UI configurable.
- `endTime`: video end time, decimal value, UI configurable.
- `altrTrack`: URL, string, UI configurable.
- `Text`: JSON object, UI configurable:
    - `Location`: the location of the text, an object with x and y properties both are decimals.
    - `Start`: the text start time, decimal.
    - `End`: the text end time, decimal.
    - `String`: the content of a text, string.

3.2 Storage

Amazon simple storage service (Amazon S3) is a secure simple and scalable cloud storage solution. Files are stored and assigned to a key in file repository called a bucket. Size limitations on a file are 5 terabytes and there is no limit on the number of files stored. An authorized user could upload, access and delete files from a bucket. Amazon S3 is a highly reliable service. By duplication of files and storing redundant files in their data centers, Amazon could reach 99.99% availability meaning the possibility of data loss is minimal. [14]

The user’s storage schema is not designed yet as the outside the scope of the prototype. In the current implementation a user owns an Amazon bucket with personal video, audio and image files. Each file type is stored in a compatible directory. Those will be retrieved by the server and passed to the client side where they will be available for the end user. The Current implementation does not include an animation library, that is accessible by all users. When such a library will be implemented, the animation assets, images and SVG images, will be stored on Amazon S3.

3.4 Database

Background

JSON (JavaScript Object Notation) is a lightweight data-interchange format. humans could read and write it while machines can parse and generate it. based on a subset of JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999 [15].

MongoDB is a NoSQL document database. Unlike relational databases, it stores documents that do not require a schema. Document sets are called collection. A document is a JSON object with given attributes. The are no constraints regarding data types as a document could contain even complex data objects as arrays, JSON objects or another document [16].

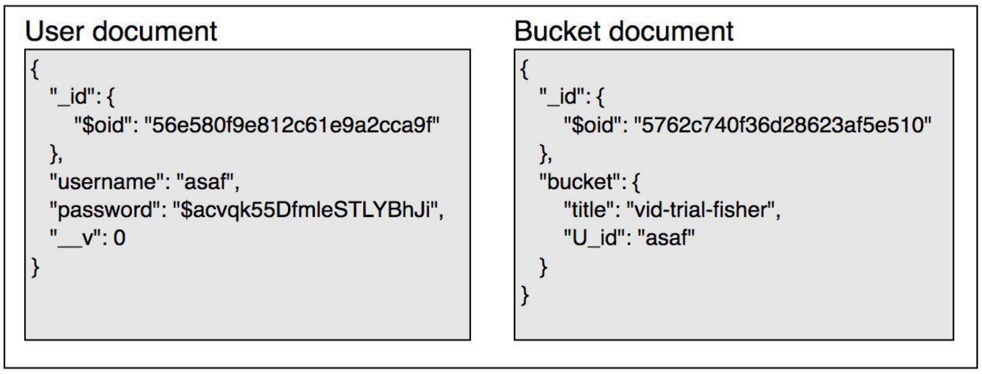
Mongoose is a Node.JS package provides many robust features. Used to define a data model schema, it enables developers to define document data and data types. Practically it provides schema data validation and business logic for queries.

Mongoose can also read, write, update, save and delete from document/collection database [17].

MongoLab is a database-as-a-service (DBaaS) provider for MongoDB, 10gen's scalable, open-source, document-oriented database currently being used by companies like Foursquare, bit.ly, and Etsy. [18]. MongoDB is a part of the MEAN (Mongo Express Angular NodeJS) and it is comfortable to work with as it uses JSON objects and, additionally, mongoose provides schema options.

Schema

For modeling, a simple scenario is described. Assuming Piranha is sold as a service, an authorized user could access his files and media assets. In order to do so two mongoose schemas are used as described in figure 11. The bucket document is an example of the Amazon S3 bucket Object and it is a part of the Bucket collection. The user document is a part of the users collection which lists authorized users, or subscribers. When a user has been authenticated, the bucket of assets should be available from Amazon S3 storage. Using the username in a filtered query to the Database bucket collection, it returns the relevant bucket ID. Querying Amazon S3 API so that the relevant bucket assets are available as shown below.


User document

```
{
   "_id": {
      "$oid": "56e580f9e812c61e9a2cca9f"
   },
   "username": "asaf",
   "password": "$acvqk55DfmleSTLYBhJi",
   "__v": 0
}
```

Bucket document

```
{
   "_id": {
      "$oid": "5762c740f36d28623af5e510"
   },
   "bucket": {
      "title": "vid-trial-fisher",
      "U_id": "asaf"
   }
}
```

Figure 11 . User, Bucket schemas

3.5 Server

Node.JS is a platform for building an event driven network program or server. Developers are provided with JavaScript runtime environment and basic libraries to write an application. The JavaScript-runtime engine, V8, is the same engine being used by Google in the Chrome browser. V8 dynamically compiles machine code that results in high performance [19]. Node.js is more of a standard library to JavaScript than a web framework and should be compared to PHP1 or Ruby2 rather than CakePHP3 or RubyOnRails4. Node.js event-loop executes events while there is an event to listen to. NodeJS demonstrates low latency compared to other runtime environments [20].

ExpressJS

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 ExpressJS have made it the choice of companies such as MySpace, PayPal and Persona [21]. The Express framework maintains a clear code structure and application structure [22,142].

The basic components of Express are router, routes and middleware. A route is essentially an HTTP verb and a path. HTTP methods as GET, POST, PUT and DELETE and path references to the resource (URI) [22, 143-145]. Middleware is an Express patterned function: `function(req, res, next)`. `req` is the incoming request and the `res` is the server response. `next` is a callback function. Middleware functions handle request and response objects, end request-response cycle and call the next middleware function. A middleware function must always call the next middleware function, even in the case of an incomplete request-response cycle to avoid request hanging [21]. Node and Express are part of the MEAN stack. It is an open source JavaScript stack so the entire application is built in Javascript.

3.6 Piranha export model processing chain

Piranha video processing is done utilizing the external APIs in the following order, Zencoder, Transloadit and Cloudinary. The processing is designed so the export model object array is sent by the end user from the client-side to the application API routes. The routes call an external APIs. A response from the external API is returned to the Piranha server parsed and passed to the client. On the client-side, a watcher function is triggered to check the process status. Once it is completed the next stage of processing begins.

3.6.1 Zencoder API integration

The Zencoder request body is a JSON object dynamically set on the client side. `zencoderManipulations()` function iterates the export model. For each video clip in the export model the function formats a JSON object and sends HTTP POST request to the Piranha server with the JSON object as the request body. Shown in listing 1, there is a JSON object for a single export model item. Request `"input"` will be the item URL hosted an amazon S3. `"aspect_mode":"pad"` is discussed in the Zencoder functionalities and it is not dynamically set. `"startTime"` and `"clip_length"` are set dynamically.

```
{
"input":"https://vid-trial-fisher.s3-us-west-2.amazonaws.com/SampleVideo_1280x720_1
mb.mp4",
  "outputs":[
    {
      "format":"mp4",
      "public":true,
      "aspect_mode":"pad",
      "size":"1280X720",
      "thumbnails":{
        "number":1,
        "public":false
      },
      "start_clip":"00:00:01.190",
      "clip_length":"00:00:02.770"
    }
  ]
}
```

Listing 1. Zencoder request JSON body

In this stage of the process, the API is not called; however, the data that Zencoder needs is formatted to a JSON object on the client-side and sent to the server. The Piranha server then sends an HTTP POST request to Zencoder API with the formatted JSON object as the request body creating a new job. The Zencoder response is an array. The response array contains the output file URL stored on the temporary storage of Zencoder and the job ID / on the temporary storage of and the job ID of Zencoder. The response is returned to the client-side and assigned to the designated export model item. While Zencoder is processing files the `checkZencoderJobStatus()` is triggered. This watcher function queries Zencoder API. It sends an HTTP GET request from the client side to Zencoder API in 5 seconds intervals to check if the job was completed. For security, Zencoder API credentials are read-only credentials.

3.6.2 Transloadit API integration

The transloadit API request body is a JSON object consisting of steps. Steps are JSON objects, describing the lowest level of action Transloadit could perform. `transloadit()` is a client-side function that iterates the export model and for each item sets an import step. If an item has an audio overlay, it will set an import step for the audio file. These steps are assigned to the import robot. The import files are the files Zencoder transcoded, and they are stored on the Zencoder temporary storage. The Transloadit architecture allows using the import step as the parameters for the other steps so after importing the files Transloadit uses the files as a parameter to the merge robot. The merge step is formatted with import step files as parameters. Then the concatenation step is formatted with the names for the import and merged files as parameters. It is important to mention that the order of concatenation steps for the files is determined by the order in the export model. Transloadit steps by order of execution:

- Import video files.
- Import alternative audio file, if an alternative track is assigned to export model.
- Merge audio and video file, if an alternative track is assigned to export model.
- Concatenate video files.

After formatting the request JSON object on the client-side. The object is sent via HTTP POST request, as the request body, to the Piranha server. The server then sends an HTTP POST request to Transloadit API with a/the given JSON object as a request body, and a new Transloadit assembly is created. Similarly to the way Zencoder is integrated, the API is queried from both the front and backend. The backend queries are reserved for operations that have a cost and the front end queries are reserved for read-only operations such as checking the assembly status. The

Transloadit API response is an assembly ID. The response is then sent from the Piranha server to the client side and passed as a parameter to the `checkTransloadit()` function which is triggered. The function sends an HTTP GET request from the client side to Transloadit API in 5 seconds intervals to check if the assembly was completed. The response to this request will contain the assembly status and a URL for the file hosted on the Transloadit temporary storage. The assembly results in a single concatenated video file.

3.6.3 Cloudinary API integration

The video file handled by Cloudinary is the result of a Transloadit assembly. Each one of the model export items can be assigned with a text object by the user on the client-side as shown in listing 2.

```
"text":{
  "location":"center, center",
  "start":"0.00",
  "end":"1.00",
  "string":"video 1 txt overlay"
}
```
Listing 2 . export model item JSON text object

Cloudinary API handles the JSON object but since the video file at hand is the concatenated assembly from Transloadit the `"start"` and `"end"` times are not valid as they refer to the time relative a specific export model item. `cloudinaryOverlay()` is a client-side function formatting the JSON object and calculating the overall time elapsed for each of the text elements and adding it to the start and end duration.

After the JSON object is formatted an HTTP POST request is sent from the client to the Piranha server with the JSON object as the request body. The server sends an HTTP POST request to Cloudinary creating a new job. Cloudinary unlike Zencoder and Transloadit APIs returns a response only when processing has completed. The response is a JSON object containing a URL for the final video. The video's URL is then returned from the server to the client.

3.6.4 Export video function

This section describes the export video process. Zencoder, Transloadit and Cloudinary APIs are called in an `exoprtVideo()` function execution. The client-side is the pipeline connecting the components. As mentioned, the export model is configured by an end user on the client side. The `exoprtVideo()` function is built synchronously and triggers different functions:

- `zencoderManipulations()`
    - `zencoderManipulations()` iterates the export model for each video and sets JSON object parameters.
    - The client sends a POST request with the JSON object as a request body to `api/zencoder.`
        - The route queries Zencoder API.
        - The router sends a response from Zencoder API to the client.
    - The client assigns the response to the export model as the zencoder_manipulation attribute.
- `checkZencoderJobStatus()`- watcher function. Queries Zencoder API.
    - `checkZencoderJobStatus()` sends a GET request from client-side to Zencoder API in 5 second intervals to check the job status.
- `cropAudioTrackForVideo()`
    - `cropAudioTrackForVideo()` iterates the export model for each video's alternative track and sets JSON object parameters.
    - The client sends a POST request with a JSON object as the request body to `api/zencoder.`
        - The route queries Zencoder API.
        - The route sends a response from Zencoder API to the client.
    - The client assigns the response to the export model as the zencoder_alt_track_audio_crop attribute.
- `transloadit()`
    - It iterates the export model for each video and sets JSON object parameters.
    - The client sends a POST request with a JSON object as the request body to `api/transloadit.`

        - The route queries Zencoder API.
        - The route sends a response from Zencoder API to the client.
    - The client assigns the response to the export model as the trans_assembly.id attribute.
    - `checkTransloadit()` is a watcher function.
        - The client sends a GET request to Transloadit API in 5 second intervals to check the assembly status.
    - `cloudinaryOverlay()`
        - It iterates the export model. for each video in the export model sets the JSON object parameters for Cloudinary request.
        - The client sends a POST with the JSON object as the request body to request to `api/cloudinary.`
            - The route queries Cloudinary API.
            - The route sends a response with the final video URL to the client.

3.6 Security

External APIs require some kind of authorization credentials, i.e. keys, to create a job or an assembly, and additionally, to check a status of a job sent to execution by Transloadit and Zencoder APIs the ID of that job/ assembly is required. Video processing costs money; therefore, all video processing requests are sent from the backend using proper routes. The routes import, require, in Node.js terminology, the API keys from the configuration file in the root directory of the application. All calls to external APIs made from the client-side contain no sensitive information such user ID. API keys are read-only. Optimally the keys will be auto-generated when a user has successfully logged in [29].

The Piranha server and the client act as a pipeline between different external APIs. The server deals with eight HTTP POST requests for a video export. The POST request bodies are in JSON format. No file uploading or video processing is done on the server; therefore, the chances for DOS attacks, which occur by overloading the server, decrease [30].

3.8 Scalability

The restful API of the Piranha server handles eight HTTP requests per video export and two more requests for S3 storage to fetch the URLs of the media assets on the page load. The watcher functions, which query the external APIs, are HTTP requests sent from the client side in five second intervals checking if the job/assembly finished / the job/assembly is finished. The logic is that the Piranha server should do as little as possible, not serving any media files while the client and external APIs do most of the heavy lifting.

4 Video production costs

This chapter describes the charging method for each of the external APIs. The costs are described to give the customer an estimation of the application operating costs. Hosting costs and storage costs are not described since they are neglectable for the prototyping purposes.

Zencoder charges \$0.02 -0.05 per a minute of output. One HD video will count as 2 minutes and an audio only file is charged as a ¼ of a minute. Zencoder offers custom plans for bigger volume customers [31].

Transloadit charges by volume. Datafisher ,The client, is already a subscriber for a custom plan, \$1.3 per GB of volume.

Cloudinary charges by volume. Video processing is measured by transformations. Each second of SD video input is counted as two transformations. A second of HD video counts as four transformations. \$224 would validates a user for 400,000 transformations [32]. Hence $224 \div 400,000 = \$0.00056$ is the price for a single transformation. However in a custom plan, negotiated with Clouduínary's sales team, \$1100 would validate a user for 12,800,000 transformations. Hence - $1100 \div 12,800,000 = \$0.0000859375$ is the price for a single transformation.

5 Test case

This chapter describes manual testing for the export video process. Three files were concatenated, overlaid with audio and text. text appearing for one second for each video. The manual test includes processing the same files for three times to check that processing times are somehow consistent for Zencoder Transloadit and Cloudinary external APIs. Besides performance testing, the tests also describe the cost of processing, as suggested in chapter 4. Videos are taken from divx.com video samples [33]. The input files properties are described in table 4. Table 5 describes Zencoder as this is the first API to import the input file for processing.

Table 4. Test case original files metadata

title	Size /MB	duration	resolution	format
BigBuckBunny	78	00:10:17	1280x720	mp4
SampleVideo	1	00:00:05	1280x720	mp4
ASAF	4.1	00:00:10	1280x720	mp4

Table 5 . Test case zencoder output files metadata, cost and processing times

title	size /MB	resolution	Video codec	format	cost in time	processing time. 3 requests	average Processing time
BigBuckBunny	78	1280x720	H264	mp4	00:20:00	00:02:07, 00:01:19, 00:01:18	00:01:34.6
SampleVideo	1	1280x720	H264	mp4	00:00:20	00:00:13, 00:00:16, 00:00:09,	00:00:12.7
ASAF	1.3	1280x720	H264	mp4	00:00:30	00:00:15, 00:00:13, 00:00:15,	00:00:13.6

Zencoder output files are Transloadit input files. Transloadit processing times and costs are described in table 6 Transloadit output is a single video file:

Table 6 . Test case transloadit output filemetadata, cost and processing times

title	size/MB	resolution	format	cost in volume/MB	processing time. 3 requests	average processing time
final_cut	91.21	1820x720	mp4	42.68	00:02:04, 00:02:11, 00:02:13	00:02:09.3

Cloudinary output is the result of the final video imported from Transloadit overlaid with text and animation. Described in table are the costs and processing times:

Table 7 . Test case output file metadata, cost and processing time

title	size/MB	frame size	format	processing time. 3 requests	cost in transformations	average processing time
final_cut	94.24	1820x720	mp4	00:02:17 00:02:08 00:02:36	4x(632)= 2528	00:02:20.4

The list below described the parameters for the total average cost for test case:

- Output time from zencoder = Z_t
- Zencoder price per minute = Z_p
- Transloadit cost in volume = T_s
- Transloadit price per volume unit (GB) = T_p
- Cloudinary transformations = C_T
- Cloudinary price per transformation = C_p

The total cost equastion is the addition of the cost of the external APIs. The cost of each API is calculated according to the charging methods described in chapter 4. The processing time of the APIs are the average values shown in tables 5-7.

$$\textit{total cost}\ (\$) = Z_t * Z_p + T_s * T_p + C_T * C_p =$$

$$(20 + 0.3333 + 0.5) * 0.02 + (0.04268) * 1.3 + (635 * 4) * 0.0000859375) = \$0.69043125$$

Adding the external APIs processing times is total time Piranha took to export a video. According to table 5 Zencoder total average processing time for the three files is 120.9 seconds. According to table 6 Transloadit average processing time is 129.3 seconds and according to table 7 Cloudinary average processing time is 140.4 seconds. The total average processing time is 390.3 seconds which are 6.5 minuets.

Total processing time =

(***total zencoder jobs duration***) + (***total transloadit assembly duration***) + (***cloudinary processing duration***)

Total average cost of test case is \$0.69. The length of total output is 10 minuets and 35 secondes shown in equation below is the cost for a single seconed of processing.

$$\textit{cost per second} = \frac{\textit{total cost}}{\textit{total time}} = \frac{0.69}{635} = \$0.0011$$

5.1 Test case result

This subchapter provides an analysis of the test case in terms of costs and performance. It is important to Datafisher because based on it a decision is made whether the prototype is viable in terms of cost and performance. Multiplying the price per second in by the number of seconds in an hour results in \$3.96/ hour of HD video output.

Despite the fact that the application is different and is a hybrid between WeVideo and Powtoon, costs are compared related to those applications. The most expensive subscriptions are \$179/month for Powtoon \$16/month for WeVideo both provide the user with unlimited HD video export. If Piranha were to break even \$179 would cover the cost 44.5 hours of HD video and \$16 would cover the cost for 4 hours of HD video.

6 Conclusion

The main goal of this final year project was to build an online video presentations editor. Since the Datafisher Oy has a wide clientele intended to use Piranha, scalability was a high priority target. 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 architechture strips the developer of the tedious work and allows focusing on creative development of new features.

Relying only on the customer feedback,cost is reasonable for creating potential profit. Since most users will not export more than a few hours a month creating mainly short, up to 15-minute videos. Additionally, the cost will decrease as the volume of use increases in each of the APIs, so the video export cost is expected to go down.

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 text and animation overlays by creating a set of pre-configured animations and a text overlays library and making animations available for the user to attach to videos and previewing. A billing system that tracks volume/cost per user is out of the scope of the study but is still to be implemented.

References

1.Ellis, Ryann K. *A Field Guide To To Learning Management Systems*. American Society for Training & Development (ASTD), 2009. Web. 9 Jan. 2017.

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]. 2016; 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 2017Jan9]. Available from: https://www.google.com/patents/US6154600.

4.CreativeCOW [Internet]. CreativeCOW. [cited 2017Apr9]. Available from: https://news.creativecow.net/story/857533

5.Editing sequences and clips in Premiere Pro [Internet]. sequences and clips in Premiere Pro. [cited 2017Apr9]. Available from: https://helpx.adobe.com/premiere-pro/topics/editing-sequences-clips.html

6.Codecs And Formats -- Zencoder [Internet]. Codecs And Formats -- Zencoder. [cited 2017Apr9]. Available from: https://app.zencoder.com/docs/faq/codecs-and-formats

7. VLAN. x264 [Internet]. x264, the best H.264/AVC encoder - VideoLAN. [cited 2017Apr9]. Available from: http://www.videolan.org/developers/x264.html

8. The History of Multimedia Messaging (MMS) - MMS London [Internet]. The History of Multimedia Messaging (MMS) - MMS London. [cited 2017Apr9]. Available from: http://www.mmsworldlondon.com/history.htm

9. About FFmpeg [Internet]. About FFmpeg. [cited 2017Apr9]. Available from: https://www.ffmpeg.org/about.html

10. Choose the plan that fits your video life [Internet]. WeVideo Plans | Get a Free Online Video Maker. [cited 2017Apr9]. Available from: https://www.wevideo.com/sign-up

11.Mersand S. Product Review: PowToon [Internet]. Tech Learning. [cited 2017Apr9]. Available from: http://www.techlearning.com/news/0002/product-review-powtoon/63310
12. Perez S. Now Everyone Can Make Marketing Videos: PowToon Launches DIY Presentation Tool [Internet]. TechCrunch. TechCrunch; 2012 [cited 2017Apr9]. Available from: https://techcrunch.com/2012/06/26/now-everyone-can-make-marketing-videos-powtoon-launches-diy-presentation-tool/

13. AngularJS Introduction [Internet]. Introduction to AngularJS. [cited 2017Apr9]. Available from: http://www.w3schools.com/angular/angular_intro.asp

14. Object Storage Details - Amazon Simple Storage Service (S3) - AWS [Internet]. Amazon Web Services, Inc. [cited 2017Apr9]. Available from: https://aws.amazon.com/s3/details/

15. ECMA International, The JSON Data Interchange Format ECMA-404 1st Edition, Geneva: ECMA International, 2013

16.Introduction to MongoDB [Internet]. Introduction to MongoDB — Getting Started With MongoDB 3.0.4. [cited 2017Apr9]. Available from: https://docs.mongodb.com/getting-started/shell/introduction

17. Automattic/mongoose [Internet]. GitHub. 2017 [cited 2017Apr9]. Available from: https://github.com/Automattic/mongoose

18. Empson R. Cloud Database Provider MongoLab Raises $3 Million From Foundry, Baseline, And Others [Internet]. TechCrunch. TechCrunch; 2011 [cited 2017Apr9]. Available from: https://techcrunch.com/2011/05/18/cloud-database-provider-mongolab-raises-3-million-from-foundry-baseline-and-others/

19. Eloff E, Torstensson , D. An Investigation into the Applicability of Node.js as a Platform for Web Service. Institutionen för datavetenskap. 2012;

20. Chaniotis IK, Kyriakou K-ID, Tselikas ND. Is Node.js a viable option for building modern web applications? A performance evaluation study. Computing. 2014;97(10):1023-44

21. Express - Node.js web application framework [Internet]. Express - Node.js web application framework. [cited 2017Apr9]. Available from: https://expressjs.com/

22. Bretz A, Ihrig C J. Full Stack Development with MEAN. Cambridge, AUS: SitePoint Pty. Ltd; 2015 , 141-143.

//23. Dupliacte of 21

23. Industry News [Internet]. Contact Center Solutions Community. 2012 [cited 2017Apr9]. Available from: http://callcenterinfo.tmcnet.com/news/2012/04/13/6257366.htm

24. Transloadit video encoding [Internet]. [cited 2017Apr9]. Available from: https://transloadit.com/tour/#video

25. Cloudinary - Features [Internet]. Cloudinary . [cited 2017Apr9]. Available from: http://cloudinary.com/features

26. 4 Interesting Business Ideas Utilizing the Cloud [Internet]. Tech.Co. 2015 [cited 2017Apr9]. Available from: http://tech.co/4-interesting-business-ideas-utilizing-cloud-2015-09

27. Pozin I. 4 Breakthrough Companies Driving Innovation In The Cloud [Internet]. Inc.com. Inc.; 2015 [cited 2017Apr9]. Available from: https://www.inc.com/ilya-pozin/4-breakthrough-companies-driving-innovation-in-the-cloud.html

28. Pozin I. 3 Startups That Will Help Your Business Save Money [Internet]. Forbes. Forbes Magazine; 2015 [cited 2017Apr9]. Available from: https://www.forbes.com/sites/ilyapozin/2015/04/16/3-startups-that-will-help-your-business-save-money/#3d69193e250c

29. Shema M. Hacking web apps: detecting and preventing web application security problems. Amsterdam: Syngress; 2012.

30. Tips | US-CERT [Internet]. Tips | US-CERT. [cited 2017Apr9]. Available from: https://www.us-cert.gov/ncas/tips/ST04- 015

31. File Transcoding Pricing | Brightcove Zencoder [Internet]. [cited 2017Apr9]. Available from: https://zencoder.com/en/file-transcoding/pricing

32. Pricing & Plans [Internet]. Cloudinary - Pricing. [cited 2017Apr9]. Available from: http://cloudinary.com/pricing

33. Video Samples [Internet]. Video Samples | DivX.com. [cited 2017Apr9]. Available from: http://www.divx.com/en/devices/profiles/video