GSoC'19 VideoCutTool Proposal

Profile Information:

Name: Gopa Vasanth
Email: gopavasanth1999@gmail.com
IRC Nick: Gopa on freenode #wikimedia-tech, #mediawiki
MediaWiki Username: Gopavasanth
Gerrit: gopavasanth
Github: gopavasanth
Portfolio: Gopavasanth
Blog: gopavasanth.wordpress.com
Time zone : UTC +5:30 (IST - India)
Location: Kerala, India
Phabricator: T217503
Mentors: **@Hassan.m.amin**, **@Rogueassassin123**, **@Doc_James**, **@maskaravivek**.
Working Hours (in IST): 42+hrs per week

Synopsis :

VideoCutTool will help to trim videos on-the-fly in Wikimedia Commons. Currently, a video in commons cannot be edited online. They have to be downloaded, changes made and later re-uploaded. As this process tends to take a good amount of time, the VideoCutTool makes authorized users work hopefully up-to 10x faster. This tool will be deployed on Wikimedia Toolforge, a hosting environment that provides services for the Wikimedia movement.

Main Deliverables :

- Simple user interface for the VideoCutTool.
- Video trimming feature using ffmpeg.
- Integrating the Express backend and React front-end with API's.
- Deployment of VideoCutTool on Wikimedia Toolforge.
- MediaWiki OAuth to validate the users with Wikimedia Commons account.
- Writing E2E and/or UI tests.
- Documenting the VideoCutTool code base.

Additional Deliverables:

- Video cropping feature.
- Integration of VideoCutTool with traslatewiki.net

Detailed Approach:

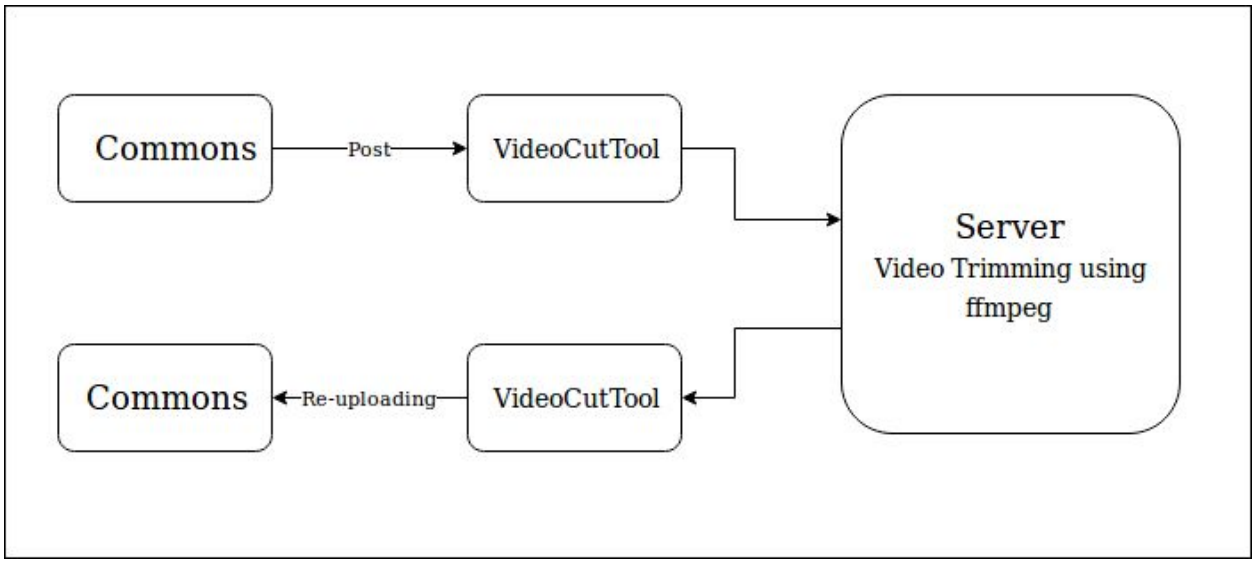

Fig: 0.1 *Flow chart of the VideoCutTool*

A simple user interface for VideoCutTool:

- The front-end will be made using the React Ant design and will be integrated with node.js as the backend by serving APIs.
- This UI mock-up is made with an inspiration of online -video-cutter and CropTool.

```
import React, { Component } from 'react';

import { Input, Typography, Layout, InputNumber, Col, Row, Button, Anchor } from
'antd';

import { Player } from 'video-react';

class App extends Component {
  render() {
    return (
        <Layout className="layout">
          <Header>
            <Typography.Title > VideoCutTool</Typography.Title>
          </Header>
        </Layout>
    );
  }
}
export default App;
```

Sample Code Snippet: 0.1 *Expected (Header) from React ant design Code for VideoCutTool front end.*

```
<Content className='Content' >
            <Row gutter>
              <Col span>
                <div className="upolad">
                  <Player playsInline poster="#"
                  src="<Wikimedia Commons Video URL>"
                />
                </div>
              </Col>
              <div className='trim'>
              <Col span>
                <h2>Video Trim Settings</h2>
                <Row gutter>
                  <Col span>
                    <Typography.Text strong >From</Typography.Text>
                    <Input placeholder="00:00:00" />
                  </Col>
                  <Col span>
                    <Typography.Text strong >To</Typography.Text>
                    <Input placeholder="00:00:00" />
                  </Col>
                </Row>
                <Button type="primary">Trim</Button>
              </Col>
              </div>
            </Row>
</Content>
```

Sample Code Snippet: 0.2 *Expected (Content) React ant design Code for VideoCutTool front-end.*

```
 <Footer >
             (c) 2018 <a href="#><span> Gopa Vasanth </span></a> |
            <a href="#"><span> Github </span></a>
 </Footer>
```

Sample Code Snippet: 0.3 *Expected (Footer) React ant design Code for VideoCutTool front end.*

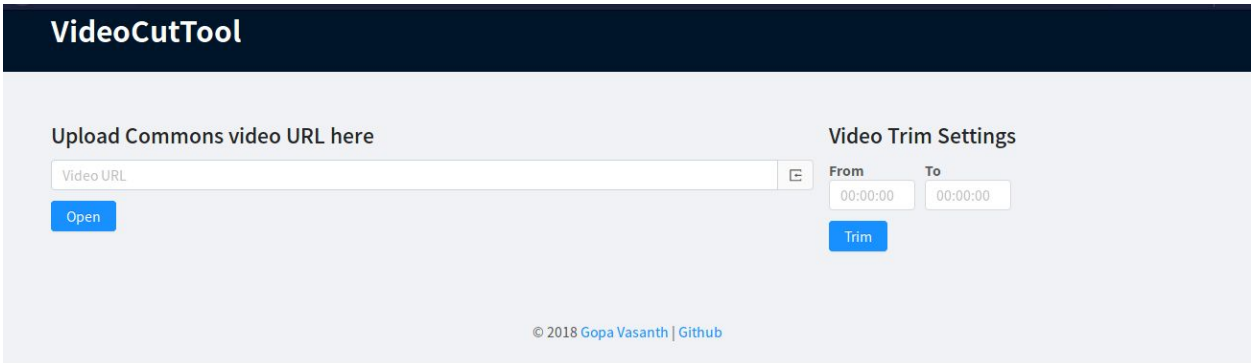

Fig: 0.2 *VideoCutTool takes the URL of the Wikimedia Commons Video.*

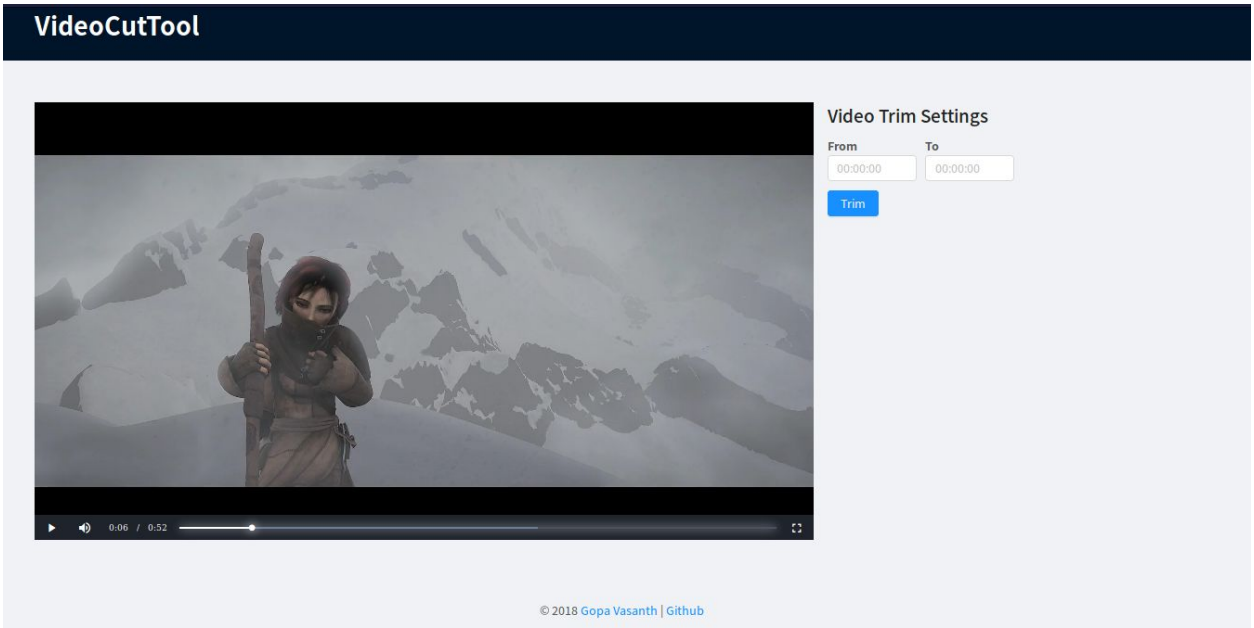

Fig: 0.3 *VideoCutTool after fetching the Video.*

Video Trimming using ffmpeg:

Videos are trimmed using ffmpeg by taking fromTime and toTime as user inputs. Here -ss represents start time and -t represents the end time of the video.

```
fromTime ← Input()
toTime ← Input()

ffmpeg -i in.webm -ss $fromTime -t $toTime -async 1 out.webm
```

Sample Code Snippet: 0.4 *Video Trimming using ffmpeg*

Video Cropping using ffmpeg:

Videos are cropped using ffmpeg by taking the out_width, out_height, x and y as the user inputs. or by enabling the user to drag the layer on video to crop and auto-generating these fields similar to CropTool for Images.

```
out_width is the required width of the video.
out_height is the required height of the video.
x and y are to specify the top left corner of the video.

ffmpeg -i in.webm -filter:v "crop=out_width:out_height:x:y" out.webm
```

Sample Code Snippet: 0.5 *Video Cropping using ffmpeg*

OAuth login with Wikimedia account.

- Register/Login using OAuth with Wikimedia account.
- Once logged in, check if the user is already registered in the database by querying the database using their wikimedia id which is returned from the OAuth login.
- If the user is already existing user update the token or save the users information into a new record with username, wikimedia-id, token secret.

```
$scope.openFile = function(updateHistory) {
  if ( updateHistory === false) {
       $scope.currentURLParms =  {
              username: getParameterByName(username),
              userID: getParameterByName('userId'),
              tokenSecret:getParameterByName('tokenSecret'),
     };
  }
}
```

Sample Code Snippet: 0.6 *Sample code if the users updateHistory is false.*

  - Either storing the user data in the session or generate a JWT token and return to the user in JSON format as

    {
    “Username”: “Gopavasanth”,
    “Token”: “JwtToken”
    }

```
User.update({ mediawikiGlobalId: profile.id }, {
$set: { username: profile.username, token, tokenSecret } },
     { upsert: true }, function(err, user) {
    return done(err, user);
})
```

Sample Code Snippet: 0.7 *The strategy requires a verify callback, which accepts these credentials and calls done providing a user, as well as options specifying a user name, tokenSecret.*

```
app.get('/auth/mediawiki,
  passport.authenticate(mediawiki, { scope: MEDIAWIKI_AUTH_SCOPE }));

app.get('/auth/mediawiki/callback',
  passport.authenticate(mediawiki, { failureRedirect: '/login' }),

function(req, res) {
    // Successful authentication, redirect home.
    res.redirect('/');
  });
```

Sample Code Snippet: 0.8 *Using passport.authenticate(), specifying the 'MediaWiki' strategy, to authenticate requests. For example route middleware in an Express application.*

API Routes:

- GET '/login'
    - Enabling the users to login via MediaWiki OAuth.
- POST '/cut'
  This enables the VideoCutTool to trim the videos using ffmpeg and the new video file title should be uploaded with the proper file description and the title is to be verified to avoid the duplication in Commons.

```
$scope.$watch('titleInput', function() {

        if (!$scope.titleInput) {
            return;
        }

        var params = parseVideoUrlOrTitle({title: $scope.titleInput}),
            key = params.site + ':' + params.title;

        if (params.title && $scope.exists[key] === undefined) {
            if ($scope.title !== params.title) {
                $scope.error = '';
                fileExists( params.site, params.title );
            }
        }
        $scope.currentUrlParams = params;
    });
```

Sample Code Snippet: 0.9 *Sample code to verify the file existence in Commons as per the user input.*

Create a new record in the database with the video information and author information. A new thread is generated to track the progress of the video from ffmpeg stdout and (piped through) WebSockets.

- Inside the new thread
  - Download the video from commons at the server for video processing.
  - Start the video cutting operation using ffmpeg.
  - Once the video is trimmed and the new video is to be re-uploaded on Commons with the few parameters (Tittle, Description, Categories) and the logged in users token.

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

var params= {
	Title: $scope.currentURLParams.title,
	Site: $scope.currentURLParams.site,
	Page: $scope.overwrite,
	Overwrite: $scope.overwrite,
	Comment: $scope.uploadComment,
	Filename: $scope.newTitle,
	Elems: $scope.cutresults.page.elems,
	Store: true
};

$http.post('./api/file/publish', params).then(function(res) {
	var response = res.data;

	if (response.result === 'Success') {
		$scope.uploadresults = response;
	}
}
```

Sample Code Snippet: 0.10 *Sample code to upload the new video back to Commons as per the user input.*

- GET '/videos/:id'
  - This returns the video recorded by id to track the progress on the visual front-end using WebSockets.

Testing and Documentation:

- Writing the Unit tests and integration tests (if possible E2E tests) to test the flow of application performance as per the designed way.

- Documenting the VideoCutTool codebase and prepare materials for further development for the tool.

VideoCutTool deployment on Toolforge:

- Wikimedia Toolforge supports node.js, MySQL in deploying tools. VideoCutTool is to be compatible with the Wikimedia Toolforge.

About Me :

I am Gopa Vasanth, a sophomore in Computer Science and Engineering from Amrita Vishwa Vidyapeetham, India. I am an active member of an Open Source student club at our university, FOSS@Amrita. I have been contributing to various projects in the Wikimedia sphere since 2017, starting with "Season of RevisionSlider" with Wikimedia DE. As part of the program, I have fixed a couple of bugs and implemented new features [1]. Also, I was a Google Code-in 2018 mentor with WikiMedia Foundation. I organized a Wikimedia Contribution drive at our club [2] in 2018 with around 30 participants. And delivered a talk on WikiMedia Foundation at our university [3].

I will be hopefully (visa application under process) attending the Wikimedia Hackathon 2019 with a full-scholarship at Prague as well.

I plan to publish blog posts during or after my GSoC. I also plan to regularly discuss with my mentors and update my work using weekly status updates. Post GSoC, I plan to be an active maintainer of this tool (as with any Open Source projects).

MicroTask: VideoCutTool

Previous Contributions to Wikimedia:

Gerrit: https://gerrit.wikimedia.org/r/#/q/gopavasanth
Github: https://github.com/gopavasanth
[1] Highlight revisions from the same user:
https://gerrit.wikimedia.org/r/#/c/mediawiki/extensions/RevisionSlider/+/454526/
[2] Contribution drive: https://www.mediawiki.org/wiki/Wikimedia_Contribution_Drive_amFOSS
[3] Talk: https://twitter.com/akhilam512/status/1106836610321018880?s=19

Project Timeline:

Weeks	TimeLine	Milestones
Community Bonding Period		
1	May 6th - May 27th	• Installing and configuring the required

2 3	May 16th - May 25th May 20th - May 27th	software in my system. ● Planning and discussing the frontend UI and other project related queries with mentors. ● Wikimedia Hackathon 2019 at Prague and discuss in depth of the project with mentors. ● Familiarising with the CropTool codebase.
Coding Period Starts		
4	May 28th - June 8th	● Designing and Implement the front end UI of VideoCutTool. ● Testing and building the UI of VideoCutTool with React Ant Design.
5	June 8th - June 15th	● Validating the video from Commons. ○ Checking for supported video formats (webm). ○ Implementing error handling techniques. ● Fetching the video to display in the front end.
6	June 15th - June 27th	● Trimming the video using ffmpeg from the user input fields.
Phase-1 Evaluation		
7	June 28th - July 10th	● Tracking the video trimming progress from ffmpeg by using the information from stdout.
8	July 10th - July 17th	● Verification procedure to make sure that the title of the new video isn't duplicated on Commons. ● Updating the database with the Title, Description, Categories, and Author.
9	July 18th - July 25th	● Tracking derivatives for the cropped videos with the original video file
10	July 25th - July 26th	● Testing and Bug fixing.

Phase-2 Evaluation		
11	July 27th - Aug 10th	• Re-Uploading the video back to the Commons. • Testing and bug fixing.
12	Aug 10th - Aug 22nd	• video cropping feature for additional benefits.
13	Aug 22nd - Aug 26th	• Writing Unit tests/integrated tests and Bug fixing. • Documenting the VideoCutTool codebase.
Phase-3/Final Evaluation		

References:

- https://www.npmjs.com/package/passport-mediawiki-oauth
- https://ffmpeg.org
- https://phabricator.wikimedia.org/T217503
- https://stackoverflow.com/questions/18444194/cutting-the-videos-based-on-start-and-end-time-using-ffmpeg