CS5543 Real-Time Big Data Analytics
Project Report 2
Project Title: VideoCeption

Team 2

Sri Praneeth Iyyapu - 8

Naresh Goud Pogakula - 20

Vihari Gorripati - 6

Mounika Yalamanchili - 25

Project Objectives

Significance:

The application that we proposed for this project is a video searching web-application. Our web-application is unique because it takes the video-search to another level. Then we will display the smart search results based on the actual content of video instead of names or tags associated with video. It is significant because currently there is no video-search application available that performs similar operation. There are few paid services, however are planning to make the project freely available with the open sourcing option as well.

Features:

1. **Smart Search:** This smart-search is based on the video content. The entire video is analyzed based on the visual and the audio content of the video uploaded to the application. So this is a smart way of searching the video instead of routine name based search.
2. **Smart Tags:** The smart-tags can be user given while uploading the video or else our video-engine analyzes the video content and gives some smart-tags that pertain to the video. The video-search is dependent on the generated smart-tags that are generated.
3. **Spam Filtering:** Many spam videos are uploaded every day to YouTube and other websites. Currently there is no way to possibly identify whether a video is genuine or spam. We try to analyze the content of the video and label the video as spam.
4. **Video Recommendations:** This feature is similar to already existing video-recommendations. But we try to give smart recommendations based on our smart-tags.
5. **Parental Control:** We can also categorize the videos based on content and generate a flag called “not suitable for children”. Hence if the parents decide to turn on a feature called “parental control”, all the inappropriate videos would be removed from the search.
6. **Content based Navigation:** You can navigate to the particular content/entity in the video using this feature.

Approach:

Data Sources: Our application needed videos particular to sports field. So we are collecting the videos from YouTube channels.

Analytic Tools:

- Apache Spark
- Apache Storm
- Apache Kafka
- OpenIMAJ

Analytical Tasks:

The Video that our application takes as input is analysed using OpenIMAJ. This tool will analyse the each frame of the video and select the main frames which are completely different to other frames.

The main frames are analysed using deep machine algorithms which will be executed on Apache Spark with the help of Storm and Kafka.

Expected Inputs/Outputs:

The application takes the search words from user and based on the search word .It displays related videos. These videos are selected based on the content of the video.

Algorithms:

The algorithm that we will use for detecting the patterns, objects and actions and store that particular information so that our application can categorize the videos and display them according to the user search.

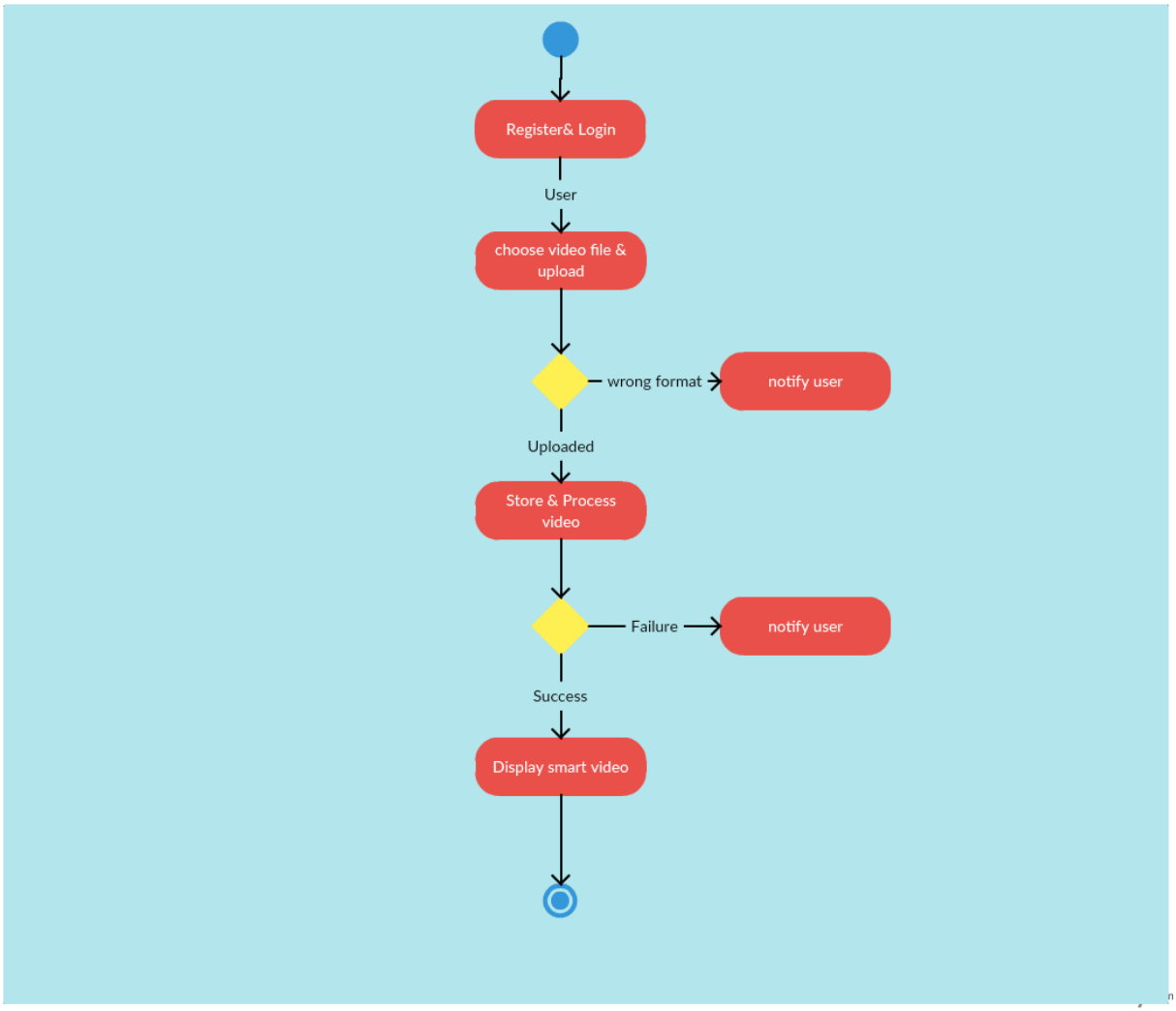
Related Work:

- Open Source Projects:
    - Dextro: https://www.dextro.co/
    - Clarifai: https://www.clarifai.com/

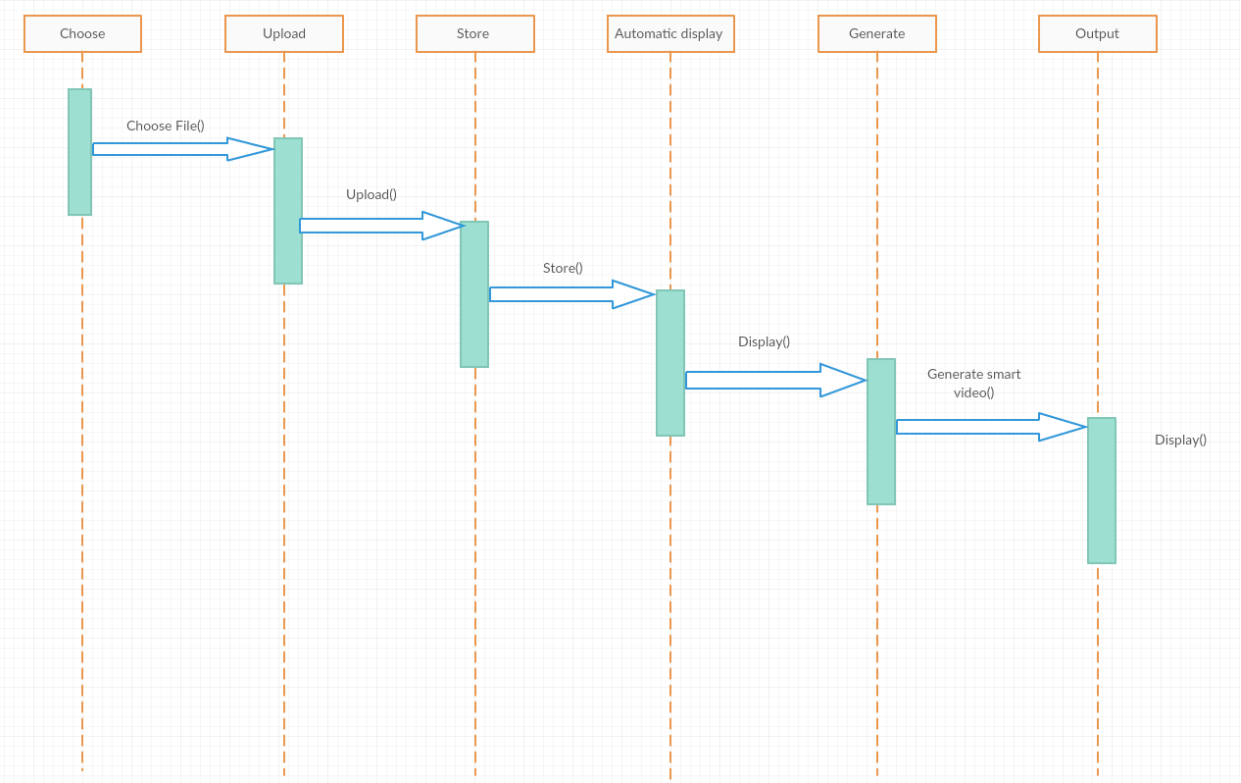
Application Specification:

- **System Specification**
  - **Software Architecture**

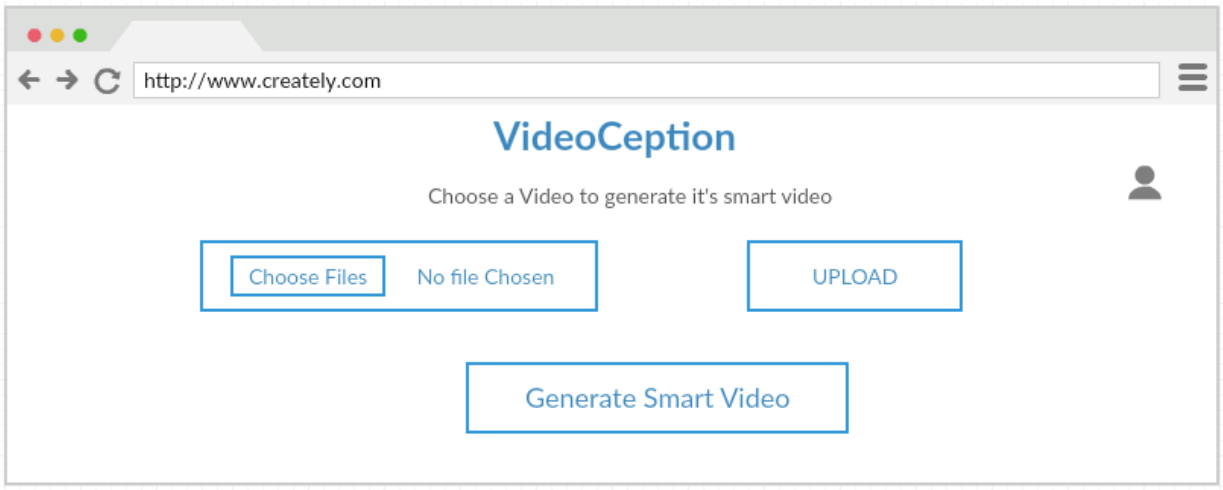
- **Design of Mobile Client (smartphone/web)**
  - **Features, GUI, Technologies**
    - **Activity Diagram (workflow, data, task)**

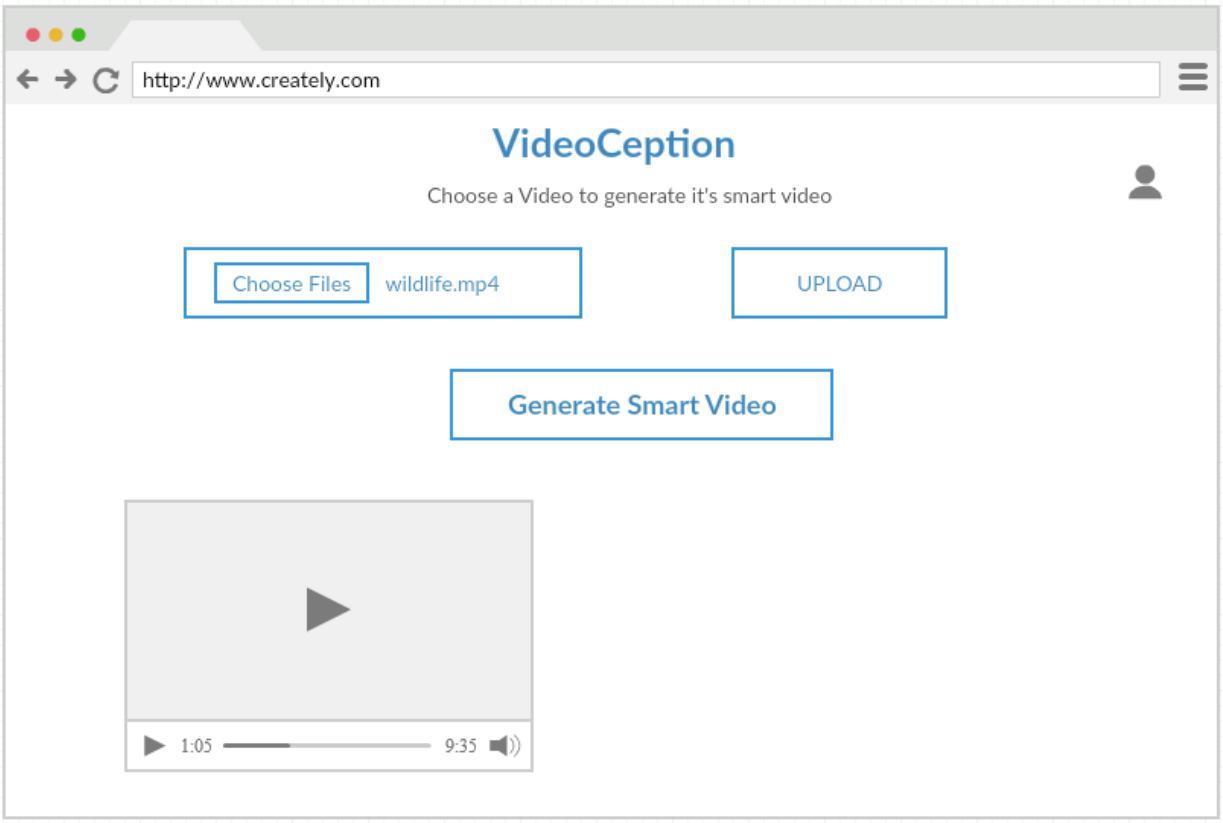


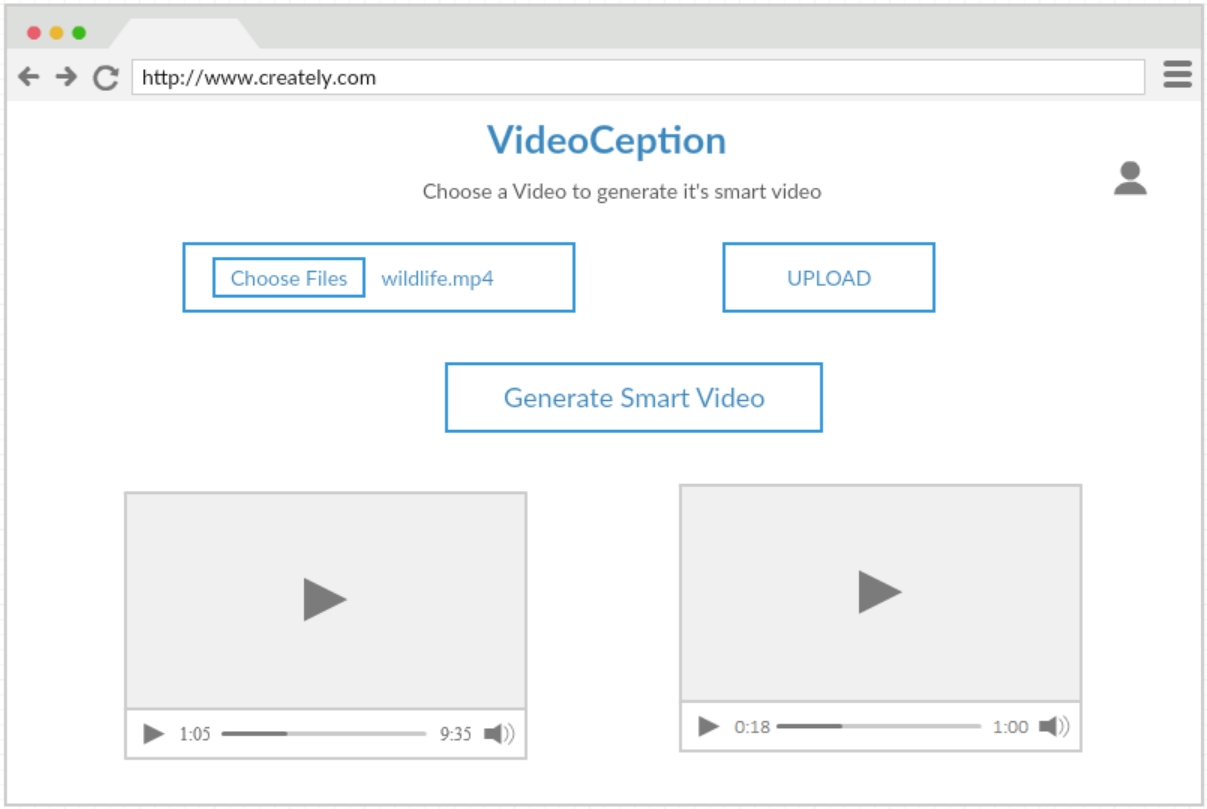
- **Sequence Diagram (interaction/collaboration)**



- **Wireframes**

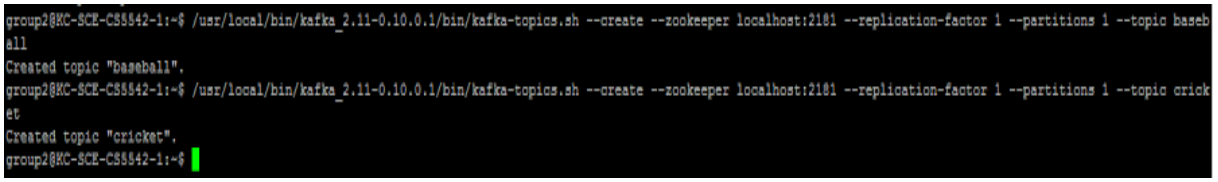


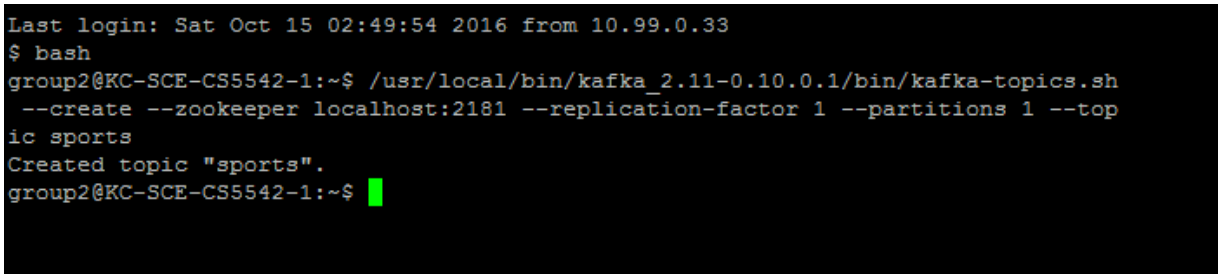

http://www.creately.com
VideoCeption
Choose a Video to generate it's smart video
Choose Files
wildlife.mp4
UPLOAD
Generate Smart Video
1:05
9:35

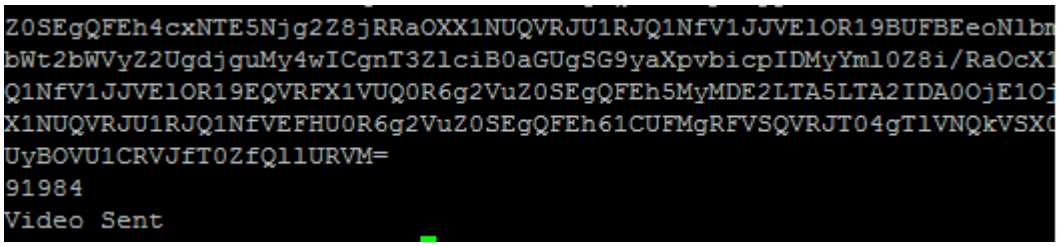

http://www.creately.com
VideoCeption
Choose a Video to generate it's smart video
Choose Files
wildlife.mp4
UPLOAD
Generate Smart Video
1:05
9:35
0:18
1:00

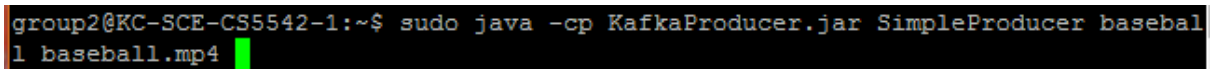
- **Existing Applications/Services Used:**
  - Dextro, It makes videos discoverable, searchable, and curatable using machine learning, https://www.dextro.co/
  - Clarifai, Powerful, secure, and affordable visual recognition API for developers and businesses, https://www.clarifai.com/
  - OpenIMAJ, Intelligent Multimedia Analysis, http://openimaj.org/

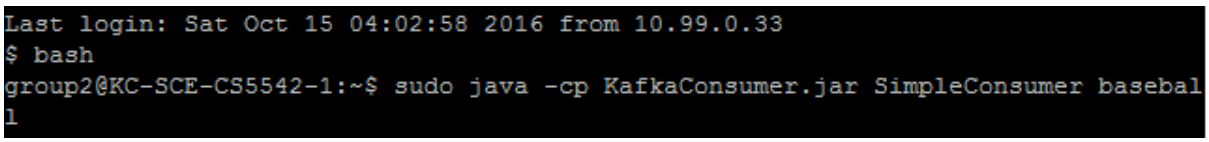
Implementation:

- Step 1: Send the video to **Kafka Producer** and get the tags for each frame.
  - For this increment we are sending the video from client side UI to server side.
  - Now various topics are created like baseball, cricket, ball, sports etc.
  - 

  - 

  - So the producer produces the video.
  - 

  - 

  - The consumers are subscribed to the respective topics.
  - 

  - We also create various bolts for the logic part. Each bolt pertains to handling the data pertaining to the different topics.
  - For example, for a topic called baseball we develop the logic in a bolt such that the bolt recognizes all the baseballs in the input based on say training data that it has.

- In the similar way we can develop logics for different topics in different bolts.
- All these data is consumed by the consumer based on the topics that it subscribed to.
- So this is our basic topology for this increment.
- We have used the [Official Client Java library](https://github.com/Clarifai/clarifai-java) to access the API.
- Code for Tags

```
File path = new File("output/mainframes");
File[] files = path.listFiles();
//request for tags
List<RecognitionResult> results =
        clarifai.recognize(new RecognitionRequest(files));
```

- The code above gets the tags for all the main frames and stores in results list. For each frame, we'll get **20 tags** and their probability associated with the frame.
- Sample Tags for a Main Frame:

```
cavalry: 0.9959737062454224
mammal: 0.9941948056221008
horse: 0.9887654781341553
motion: 0.9825373888015747
outdoors: 0.9734552502632141
water: 0.9731769561767578
no person: 0.9723465442657471
sitting: 0.9648486375808716
evening: 0.9642931222915649
equestrian: 0.9608144760131836
side view: 0.9573180675506592
sunset: 0.9562845230102539
```

- Step 3: Produce a new video by adding tags to the respective frames.
  - We have added **6 out of 20** tags (based on the highest probability) for each main frame and constructed the new video. The following is the code snippet for the Video Reconstruction.
  - Video Reconstruction

```
for(int i=0;i<results.size();i++){
    MBFImage image = ImageUtilities.readMBF(files[i]);
    int j=1;
    for (Tag tag : results.get(i).getTags()) {
        //System.out.println("result : "+i+" "+ tag.getName() + ": " + tag.getProbability());
        image.drawText(tag.getName(), (j*30)+40, (j*30)+50, HersheyFont.ASTROLOGY, 30, RGBColour.BLACK);
        if(++j>6){
            break;
        }
        //break;
    }
    for(int k=1;k<200;k++) {
        DisplayUtilities.displayName(image, "videoFrames");
    }
}
```

Video Output and Screenshots:

- The individual main frames from our video with the annotated tags (**we have annotated the top 6 tags with most probability to each frame**) from our program
- Picture 1

- Picture 2:

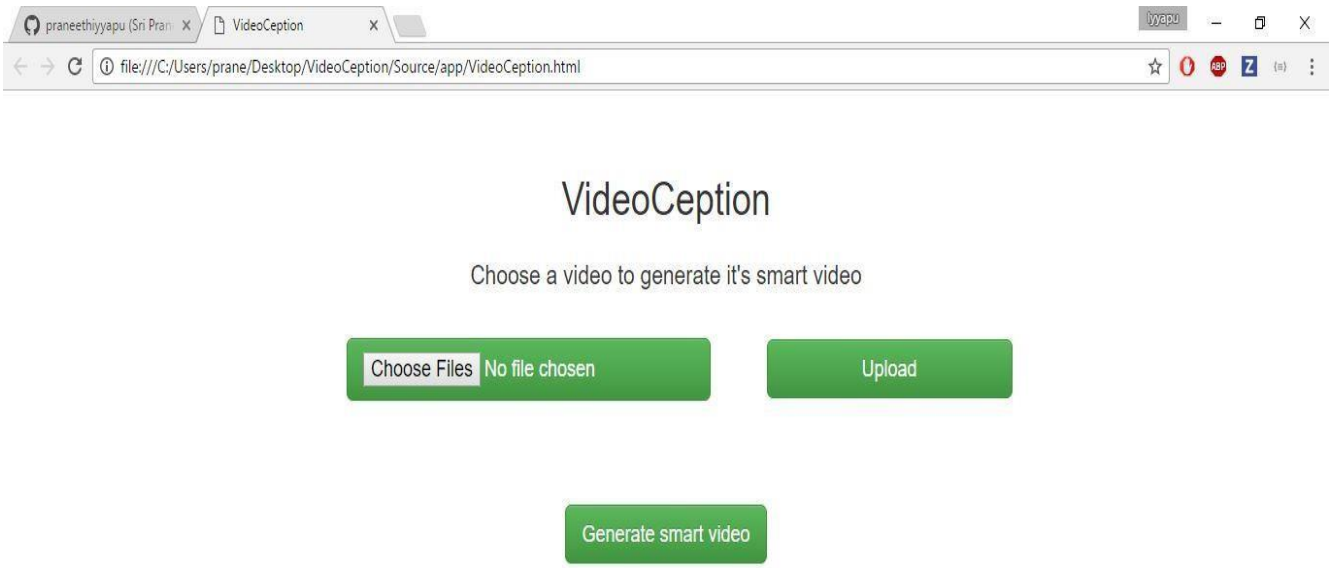
- Picture 3:

- Picture4:

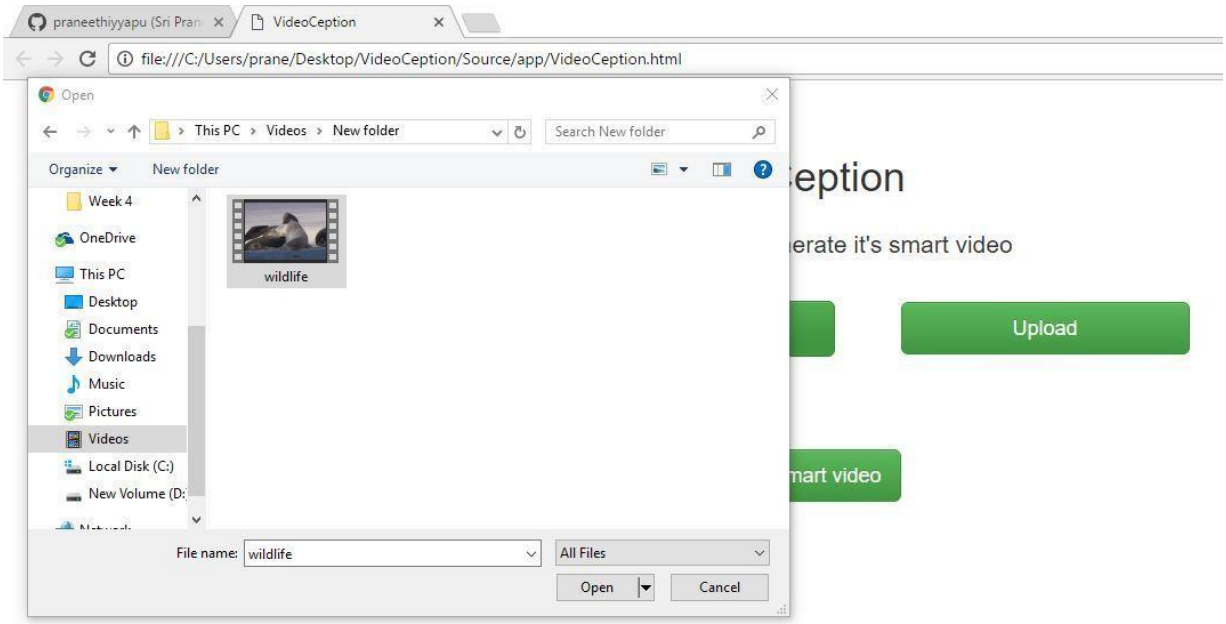
- Picture 5:

Documentation:

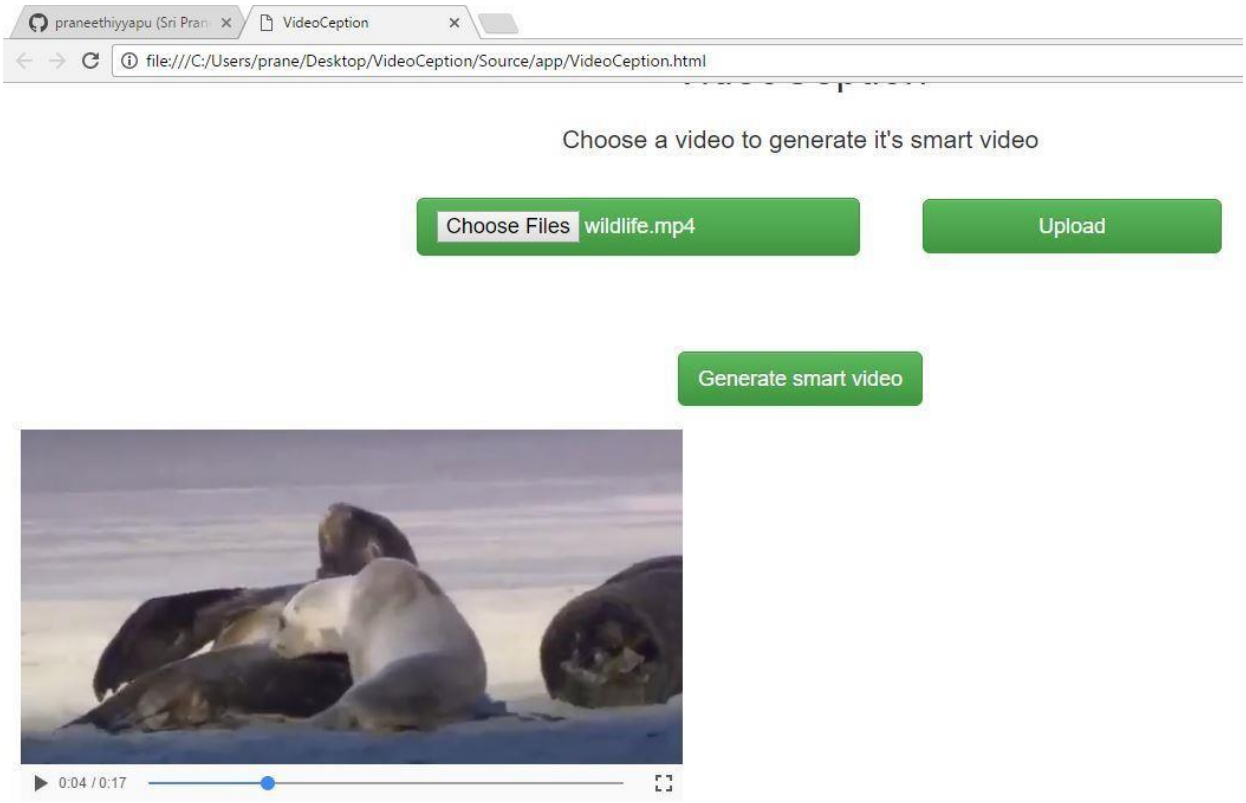
- The home screen of our application has three options.
    - **Choose files:** The user can choose a video from the local machine
    - **Upload:** Upon choosing this option the video gets uploaded to the Kafka Producer running on AWS server.
    - **Generate a smart video:** This option generates a shorter/smart video



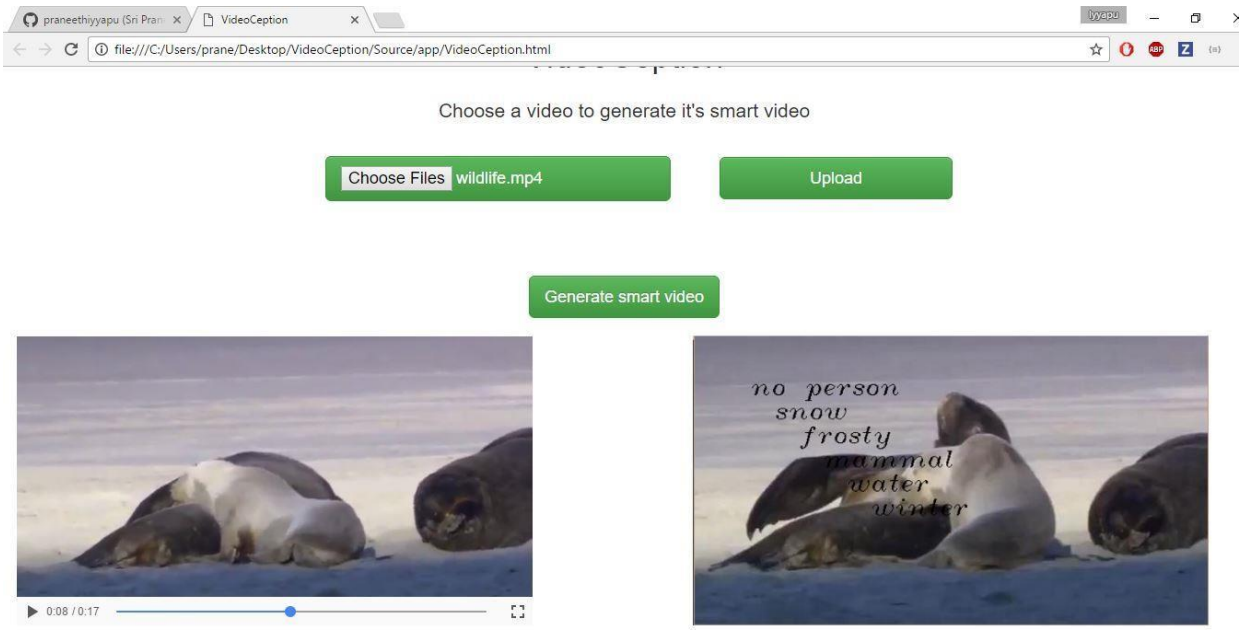
- The user chooses the video to be converted as below:



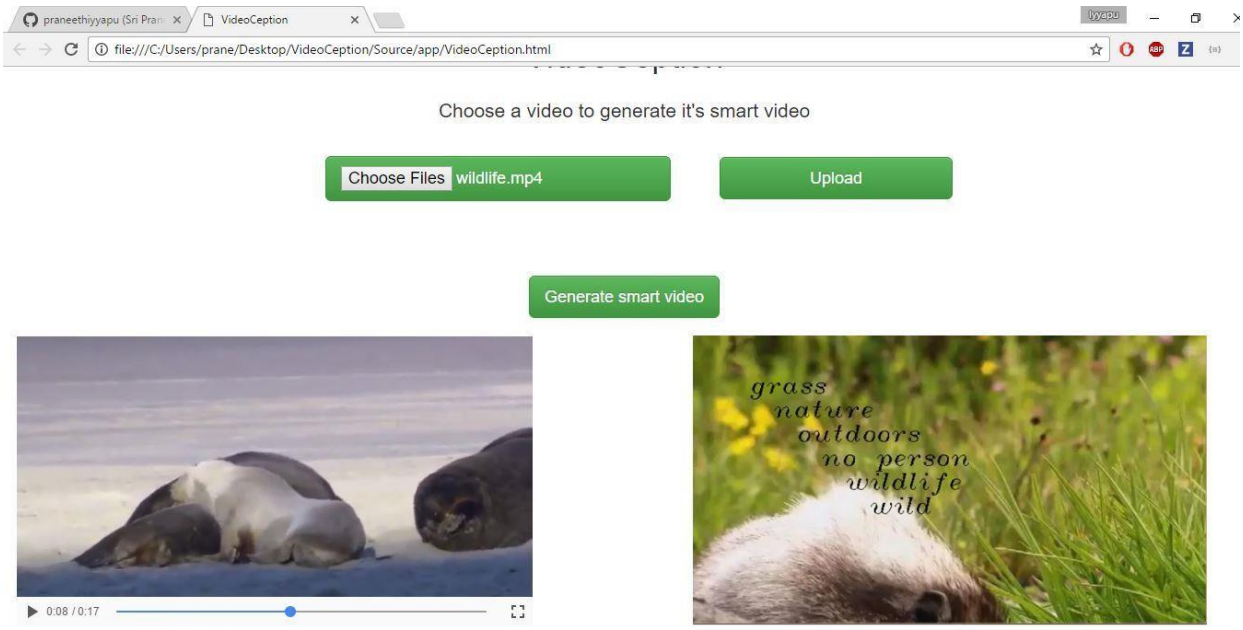
- The unprocessed/ the original video gets displayed as below



- Now when the user selects the “generate smart video” the processing happens at the back end and a short video/smart video is generated as below.



- The annotated short video is now consumed at the consumer end.



Project Management:

- **Work Completed:** We have created a web interface and integrated the java programs to the front end so the smart video could be shown to the user. The video can be uploaded to the server where processing would happen using Kafka and Storm
- **Time taken:** It nearly took around 50 hours of our time.
- **Contribution:** Everyone has contributed equally.
- **Work to be completed:** Implement our own algorithms to replace the third part services.
- **Time to be taken:** We would be needing approximately another 80 hours to finish the entire project.
- **GitHub Repo:** https://github.com/niCEnANi/RealTimeBigData/tree/master/Project%20Report%202