Cartooning of an Image Using Python

Fundamentals of Artificial intelligence

CSCE 5210

Group 22:

Team Members:

Anvesh Chinta

Sai Krishna Reddy Beluri

Srikanth Reddy Guntaka

Abstract:

This project paper describes the process of converting the image which is received as input, to a cartoon-like image with the help of reference images. Cartoon images have a few colour variants with a dark border for the edges. The main aim is to develop a system that gives a Cartoon view of the real input image. It takes a lot of time and space to create a cartoon effect. Currently, the cartoon-effect solutions available are complicated. Other solutions involve performing certain tasks by the user, while others require installing complex software like Photoshop. Toony-Photos is an existing website to perform such a task, but it is difficult to use as the user has to mark down points & lines on the image to apply effects, which isn't very user-friendly. Also, the options are limited. In this regard,

there is a great need for a site that is user-friendly and which is capable of imposing visual effects on images. Bilateral filter and edge detection are the two machines we have used in the process of the real image, in which the edges of the real image is identified by the system and the real image is enhanced to a cartoon view image.

Acknowledgement:

The paper proposed a method for cartooning using reference images. By deforming the target image into the same shape as the reference image and adding cartoons, it produces the resulting image. This program is very straightforward to use. Various result images can be generated by adjusting the deformation intensity of the target and the cartooning intensity.

An algorithm limitation in this paper is that the feature point model of the reference image must be predefined and provided. Additionally, the deformation for the exaggeration of the target image may only be applied to the frontal face. The improving method for these problems is being researched now.

If the system of this paper adopts the model extracting feature points from the face of diverse angles through the expansion The deformation for exaggeration of the target image can also be applied only when a non-target image is used and a target image by using several reference images is also in the plan.

Introduction:

The process of converting the image into a cartoon-like view is similar to the method of image processing. Exaggerating the real image as the cartoon image is really difficult, so only a cartoon specialist does that. Usually, only cartoon specialists make cartoons. with current sophisticated technology and more libraries in python, we can easily create it using our computers. In the currently available model, the cartoon exaggerates the target. Cartoons have this feature, and that is what makes their cartoons. It is very difficult to convey to every user the exaggeration in the cartoon. A study that creates cartoon-like images with a computer has been developed to help novices easily create cartoons.

Related Work (with Linked References):

Toonyphotos (http://www.toonyphotos.com/) is a well-known site used for cartoonify imaging but because of its not user-friendly interface and difficult to use as the user has to mark down points & lines on the image to apply effects. Since it is an established site we take this as a reference and developed a proposed version on it. And we have linked all reference papers and websites we reviewed in the last of this document.

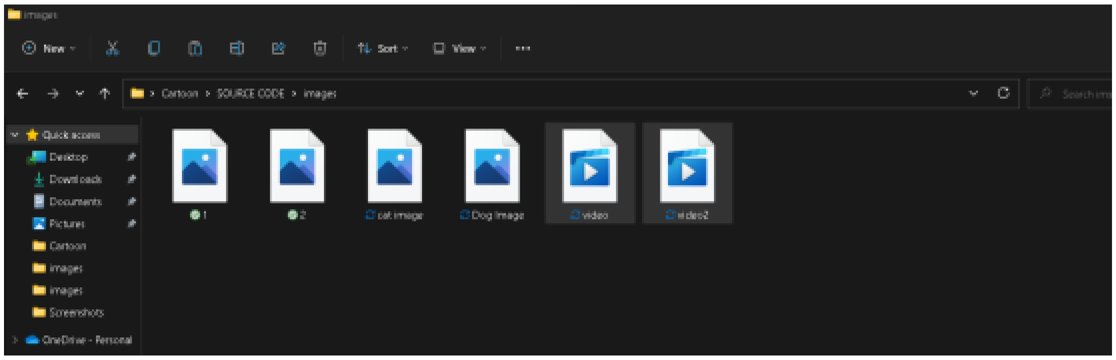
1. Problem Specification:

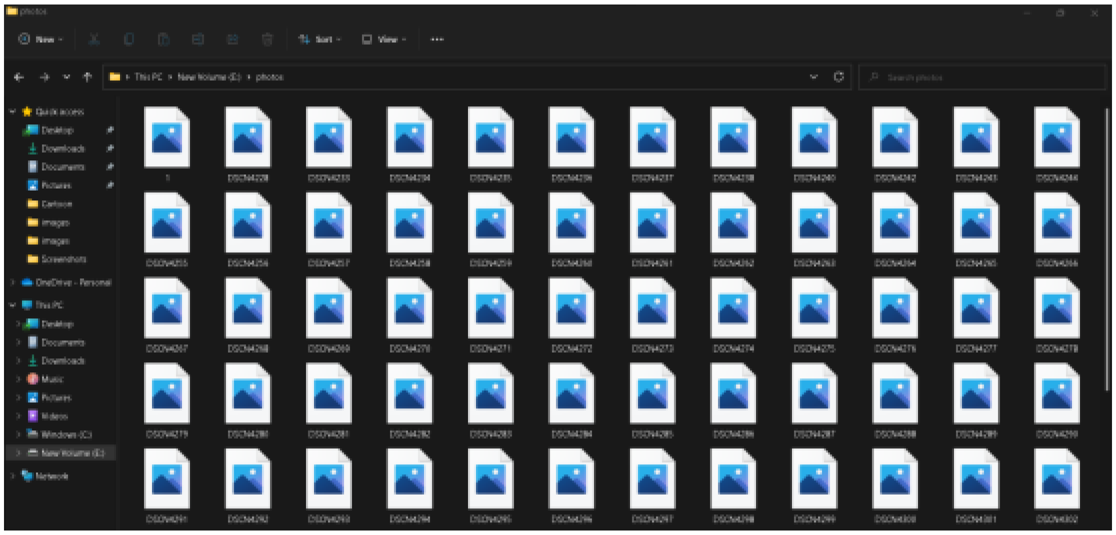
With the current latest and greatest technology, we can use machines to create cartoon-like images in a very short time this is the main advantage of this program but However, most cartoon rendering systems cannot give their results in various ways, since their results are generated according to a fixed algorithm. There are also other cartoon rendering methods that use textures or use user interaction to produce different results. Unfortunately, these methods are not intuitive. Few image processing techniques are performed on the real image, to eliminate functional data and traffic in the image. The processed image is then converted to cartoon-like images by extracting the edges of the image and then applying the median blur of the bilateral filtration method. Hence they turn out to be not that effective. For example below is an image that is not processed well and it doesn't feel like cartooning.

1.1 Data-Set:

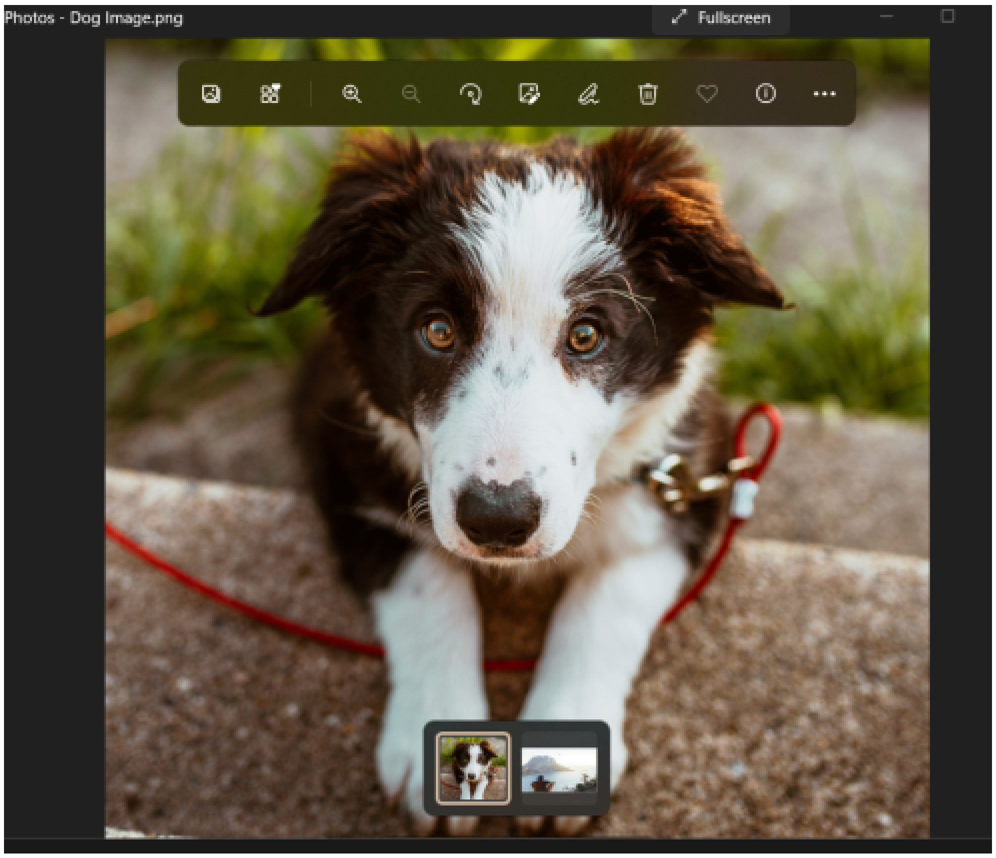
In this model, the data which is passed to the process is an image or video. The main theme of the project is to bring the cartoon effects to pictures or the video which is uploaded by the user.

The images which are passed while executing are around 1 MB and the video size is around 10MB.





The image which is going to upload to the model can be a .jpg or .png format as these both are accepted by or model. The below screenshot is the real image of the dog, the cartoon image of the same dog picture is mentioned in the results in this document.



It is not like the data-set is fixed for this model, whatever the image or the video is uploaded from the user side, the model brings the cartoon effects to it.

1.2 Problem Analysis:

As we already identified the problem we need to analyse the root cause and try to improve the system from there. The idea was to create cartoons using reference images. By deforming the target to the form of the reference image and adding cartoons to it, the resulting image is obtained. Every user can use it easily. A user can generate a variety of result images by adjusting the intensity of the deformation and cartooning. One of its limitations is that the reference image has to be pre-defined and provided with the feature point model.

2. Design and Milestone:

We know what to design in this model from performing the problem analysis. We first need a few essential parameters to perform this task like with min requirement system is listed below

Minimum Hardware Requirements:

System	: Pentium IV 2.4 GHz and above
Hard Disk	: 40 GB and above
Ram	: 512 Mb and above

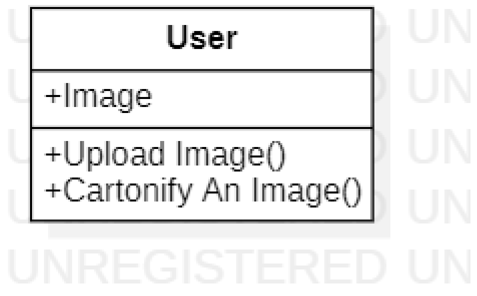
Minimum Software Requirements:

Operating system	: Windows 7 and above
Coding Language	: python 3.0 and above

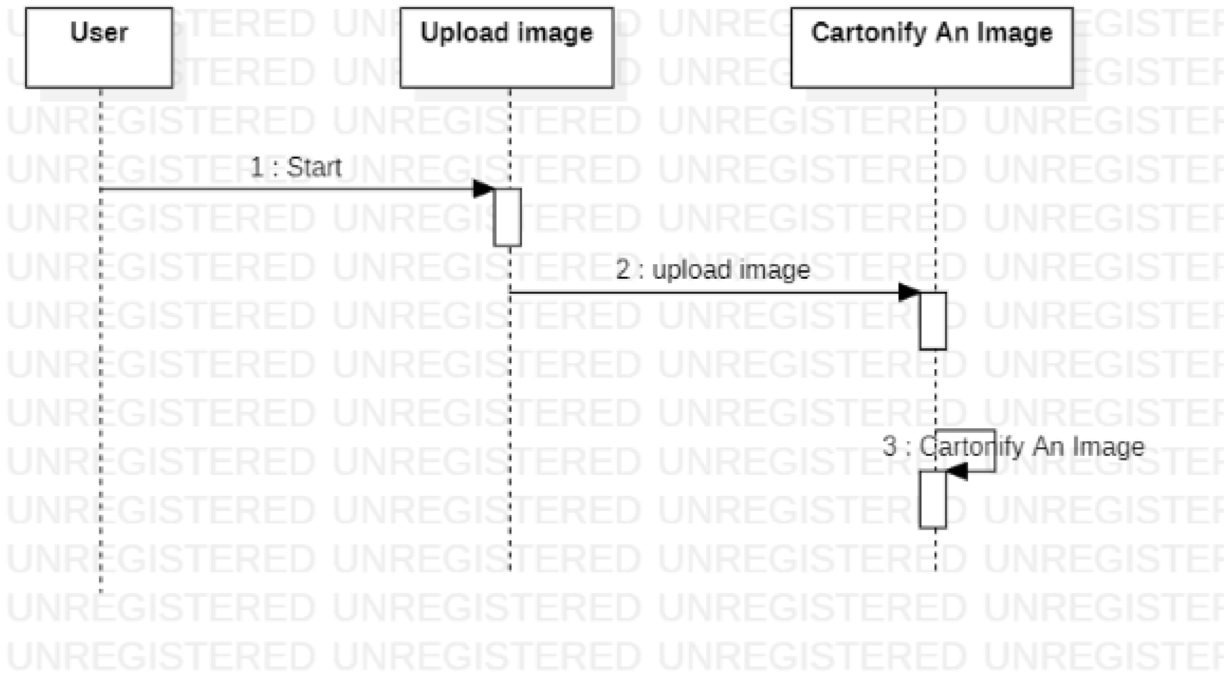
As per our model we just need very few modules to perform the operation of cartoon-like images. Below listed are the required modules.

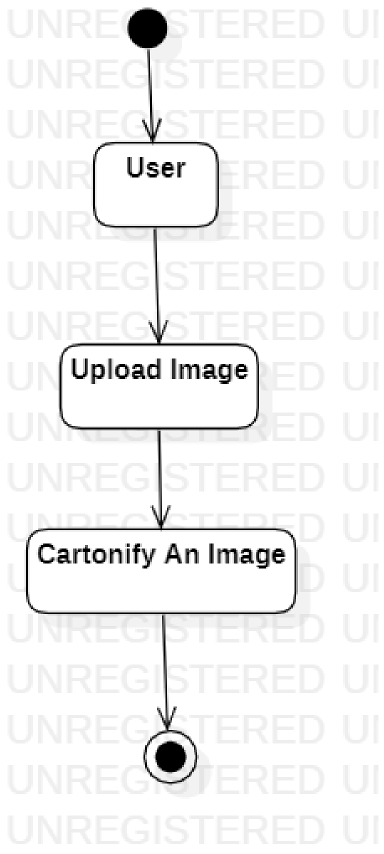
Modules:

- Upload : In this module user uploads image for the folder.
- Cartonify An Image: In this module, the user converts images into cartoon Images.



Below is the sequence and activity diagram designed for this model





As shown in the above sequence and activity diagram the model is simple, flexible and user friendly with fewer components. To define the sequence user starts the process by uploading the image of his choice for the task and by clicking cartooning the image option model will act on it to give output.

2.1 Proposed System:

This model at first downscales the image by applying a bilateral filter to get a cartoon flavour to the real image. A bilateral filter is a filter that processes the image by homogenizing the colour of the image by preserving the edges.

This technique returns the blurred image of the real image which is later converted to greyscale.

The next step is to identify the edges of the images using a technique called Edge Detection. Edge Detection is the main asset of the model which analysis the whole image and identifies the areas which have high contrasts.

2.2 Data Processing:

In this project model, the model accepts the picture and processes the picture by downscaling and extracting the edges, converting it to a grey image and then applying the bilateral filtering technique image to cartooning image.

Since we are implementing this model in python All below-listed modules such as

pip install NumPy

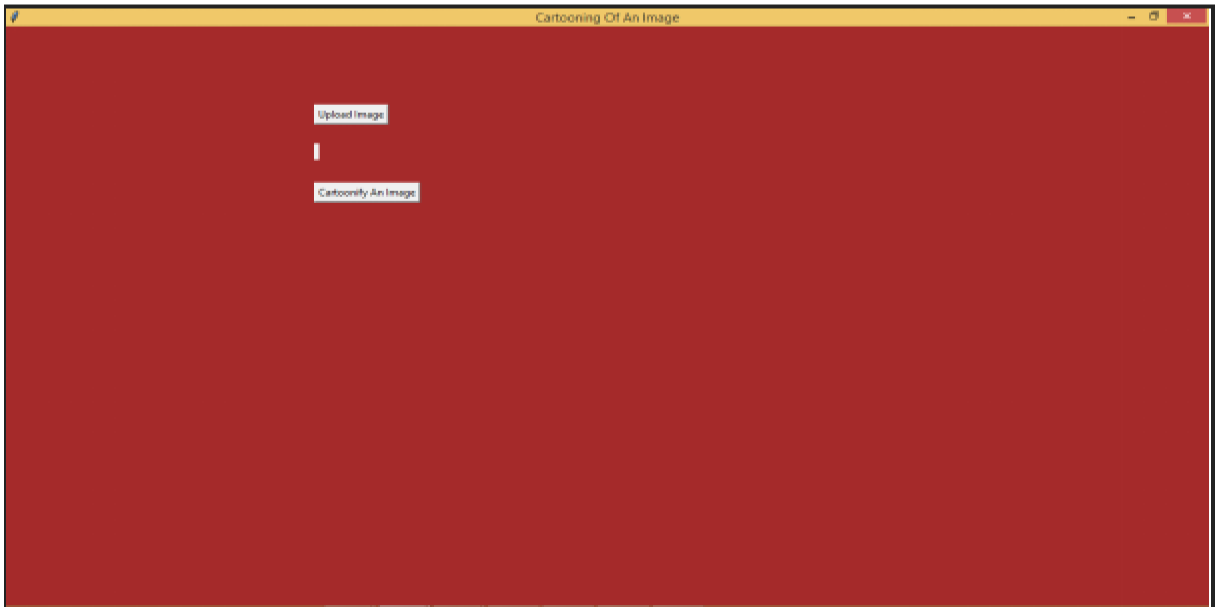
pip install scipy

pip install OpenCV-python

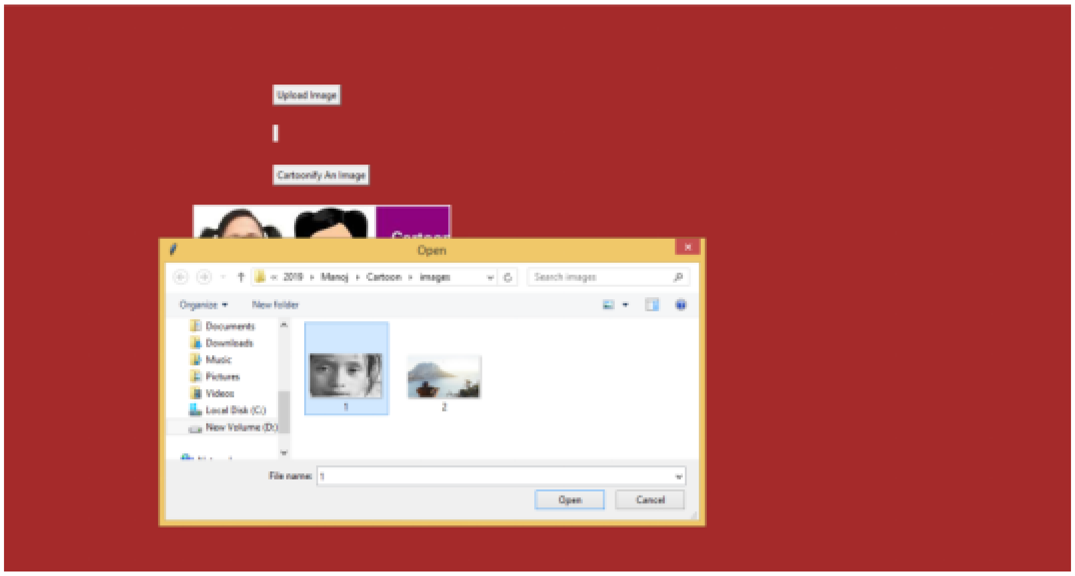
pip install pillow

Are imported for the program to execute.

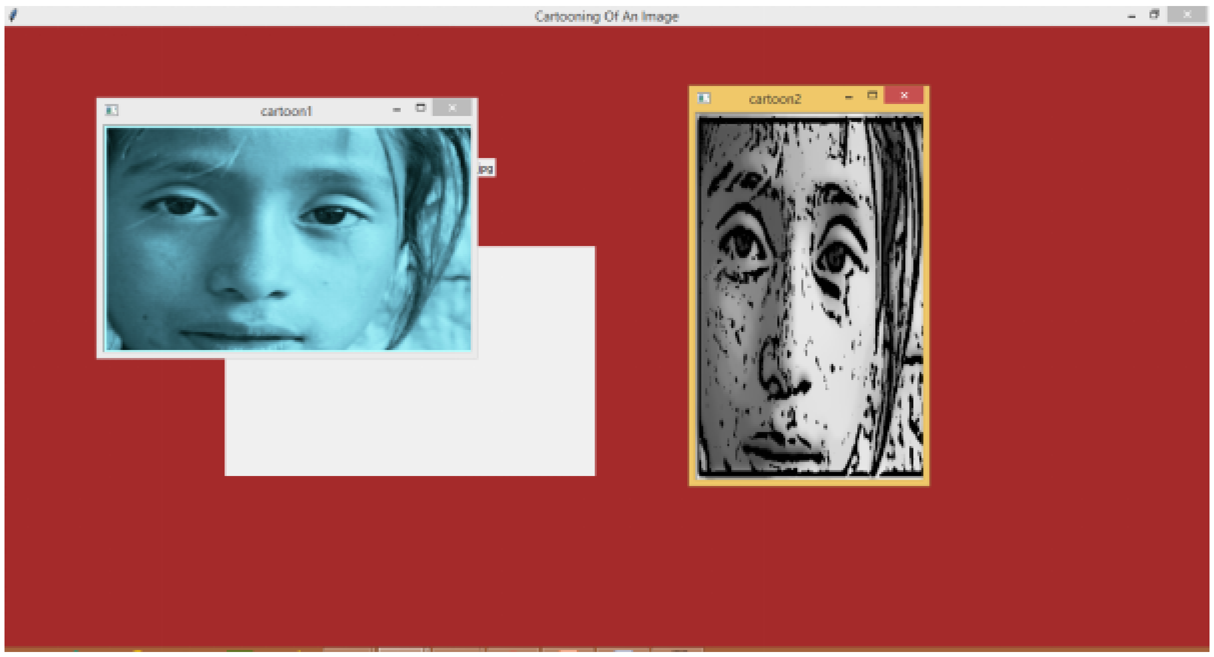
When the model is launched, this kind of interface can be observed



The image which has to be converted into a cartoon should be uploaded first by clicking the upload image button on the screen.



After uploading the real image, by clicking the cartoonify the image button. The image will be converted into a cartoon.



The Above image can be considered a preliminary result.

2.3 Experimental Settings:

In the implementation of this model, we deform the target to match the reference image and add cartooning effect to it, and the resulting image is obtained. A user can generate a variety of result images by adjusting the intensity of the deformation and cartooning. By increasing and decreasing contrast, brightness, shadows, exposure and tint we can create so many variations and gets a perfect cartoon effect we need to experiment with a couple of combinations with those settings and choose a few variations of it to get as output. This might give us a better idea of how to interpret the different types of images and give the best outputs.

2.4 Validation methods:

Since this is a user driver model there is no specific validation method in order to verify the output. We are checking the best-fit settings but manually uploading and checking the outputs and improving the results from the samples.

We are following below exit criteria to validate the best results.

- The operation time should be small and the throughput should be high.
- It should produce timely and accurate results.

Below are the test cases for user requirements:

In Application Home Screen:

Use case ID	Cartooning Of An Image
Use case Name	Home button
Description	Display home page of application
Primary actor	User
Precondition	User must open application
Post condition	Display the Home Page of an application
Frequency of Use case	Many times
Alternative use case	N/A
Use case Diagrams	
Attachments	N/A
Pass criteria	Successfully giving desired output image

Limitations:

- The reference image has to be pre-defined and provided with the feature point model.
- It's hard to pick the perfect settings to produce output since every image is different and has different factors.
- Since it is a computer-generated model for now it is still not perfect when compared to the work of a cartoonist.

Future Work:

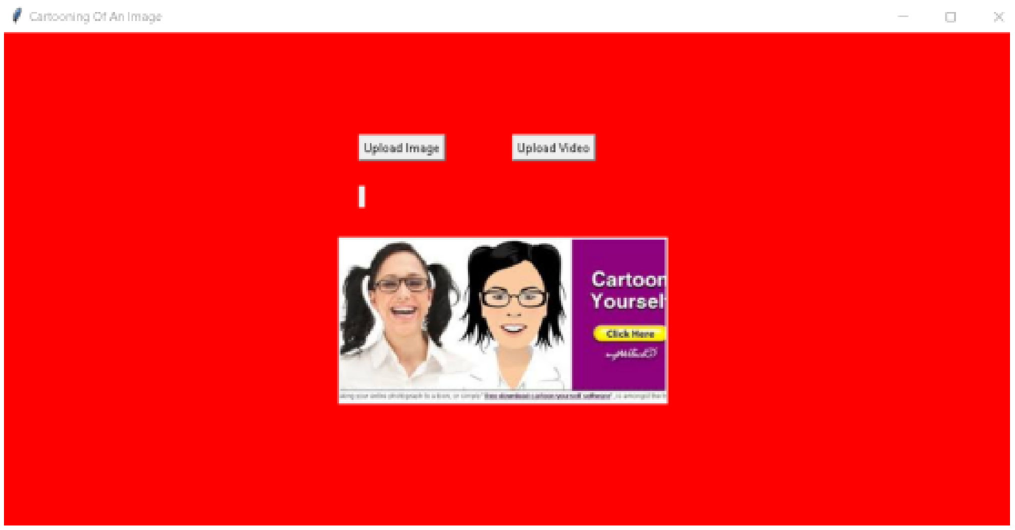
The pictures which are returned are cartoon views but are low in quality with fewer colour variants. In future this model can be developed to return the high-quality picture with the 3D feature, even 3D motions can also be added as the extended version for this model.

3. Project Management System:

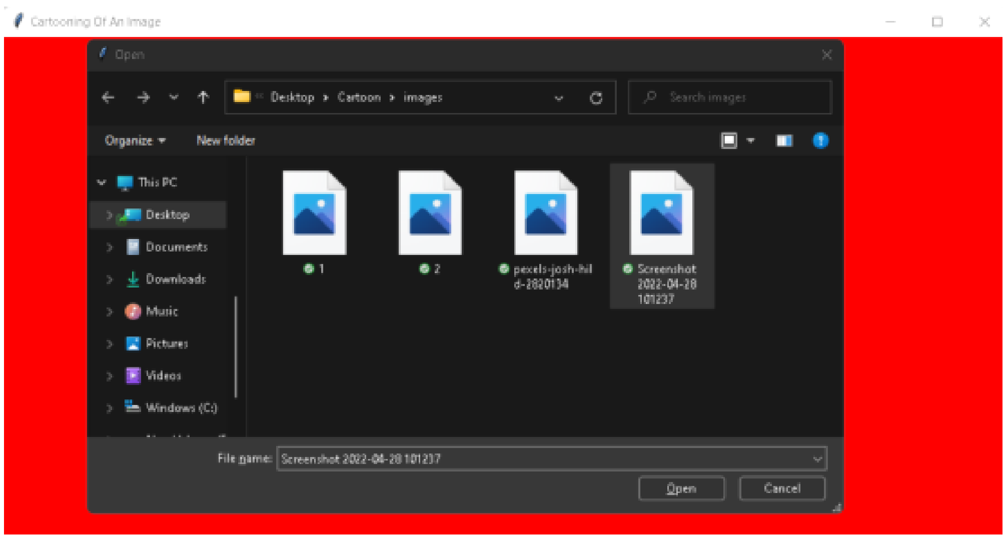
3.1 Implementation Status Report:

To Cartoonfying the image, the image has to be selected which has to be converted.

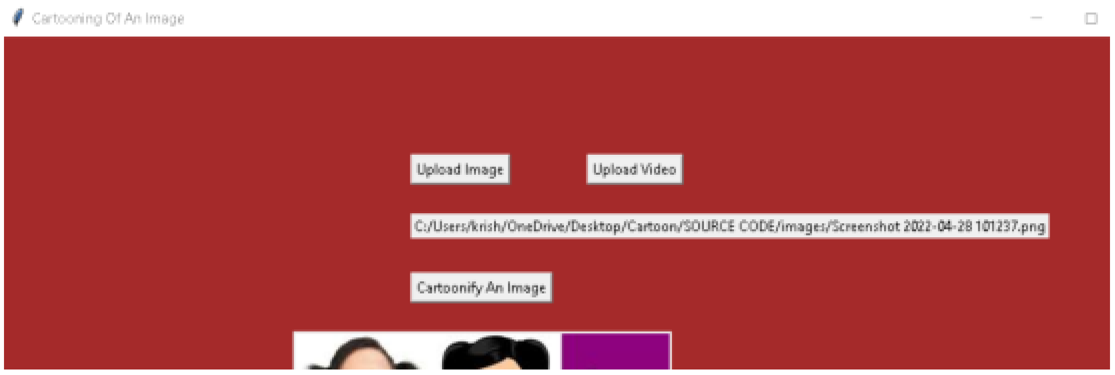
After hitting the run folder in the project folder, the window pops out with which users can interact and can upload the image or the video to cartoon it.



Hit upload image or upload video, on user requirement

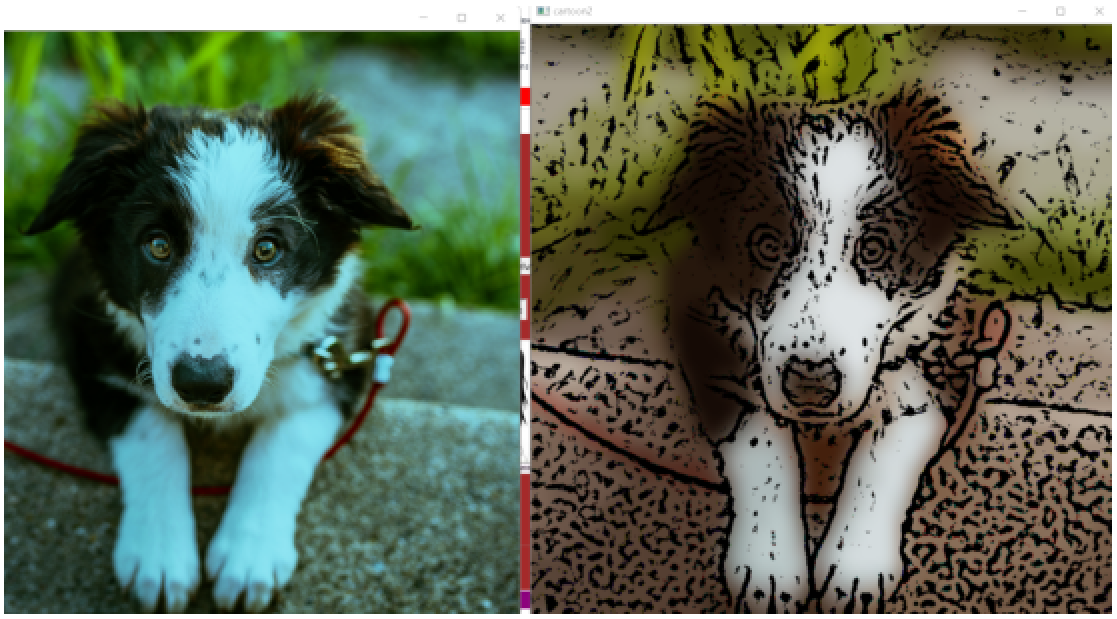


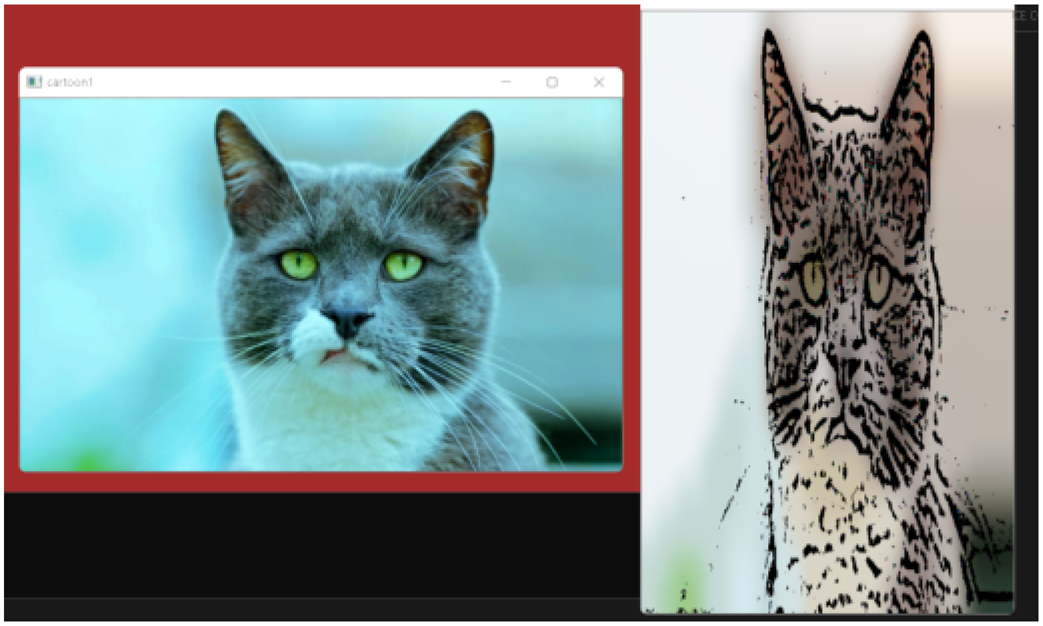
Then it asks to select an image or video from the system which has to be converted.



C:/Users/krish/OneDrive/Desktop/Cartoon/SOURCE CODE/images/Screenshot 2022-04-28 101237.png

After successfully uploading it, it shows the path and file name which is uploaded. If it is correct, proceed to hit the "Cartoonify An Image" button then the model converts the uploaded file to cartoon patterned.





The above screenshot is the final result, that model has converted.

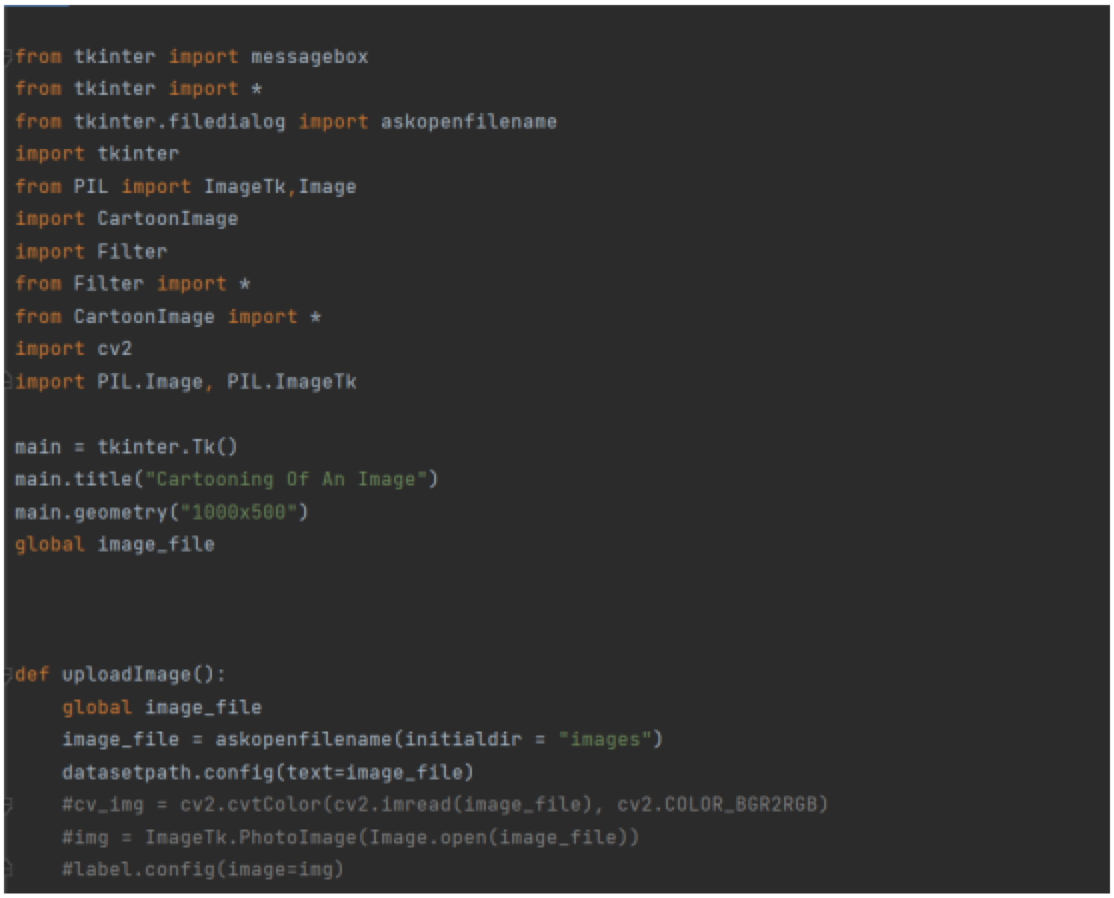
3.1.1 Work Completed:

Successfully implemented the proposed model and with results as expected and sample results are listed above in implementation. The final code will be available in the below-given GitHub link.

GitHub Link: https://github.com/Krishreddy11/Cartooning-the-Image

3.1.1.1 Description:

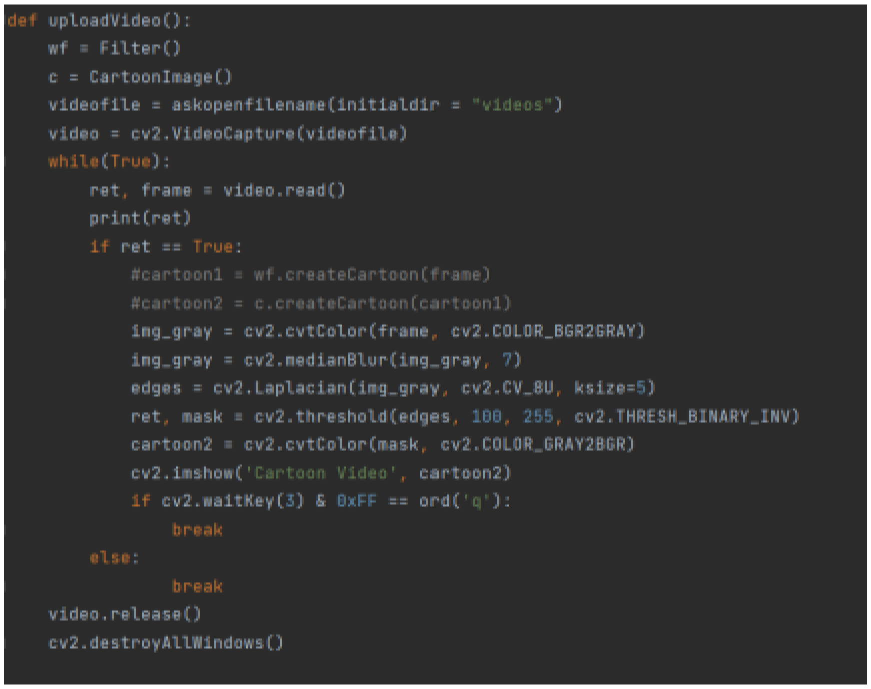
As per the proposed model we have implemented the model using python with lib like open CV and numpy. Below are a few code screenshots and their working explination.



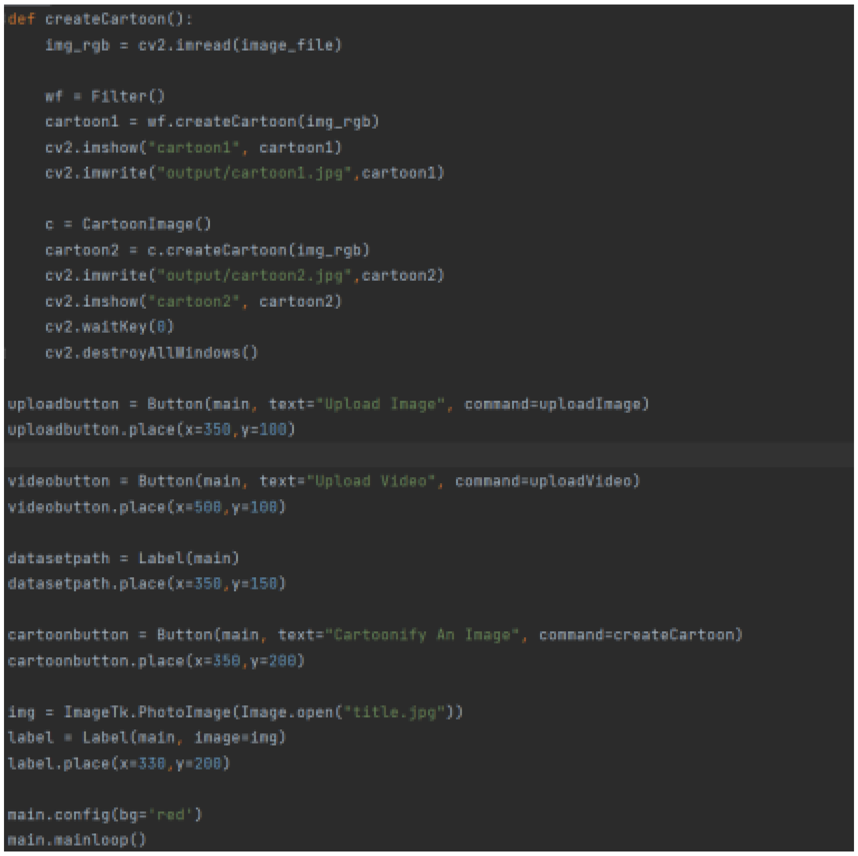
```
from tkinter import messagebox
from tkinter import *
from tkinter.filedialog import askopenfilename
import tkinter
from PIL import ImageTk,Image
import CartoonImage
import Filter
from Filter import *
from CartoonImage import *
import cv2
import PIL.Image, PIL.ImageTk

main = tkinter.Tk()
main.title("Cartooning Of An Image")
main.geometry("1000x500")
global image_file



def uploadImage():
    global image_file
    image_file = askopenfilename(initialdir = "images")
    datasetpath.config(text=image_file)
    #cv_img = cv2.cvtColor(cv2.imread(image_file), cv2.COLOR_BGR2RGB)
    #img = ImageTk.PhotoImage(Image.open(image_file))
    #label.config(image=img)
```

Imported all the modules which are required to build the model, basically the picture which is uploaded model recognizes the edges of the image, generates the grey image and then using filter and CV modules the unnecessary noises are eliminated and given colours to the grey image.

In the above lines of code, the interface is created with which users can interact and can upload images or videos to cartoon it.

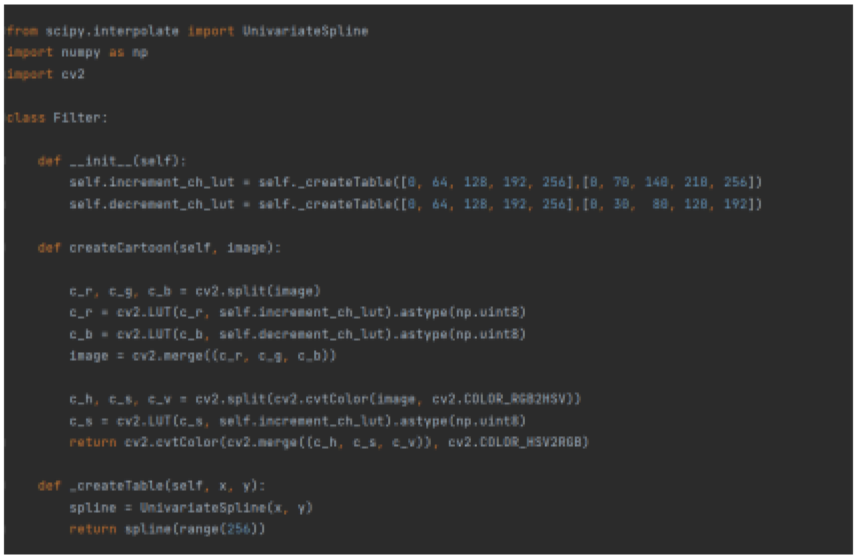


```
def uploadVideo():
    wf = Filter()
    c = CartoonImage()
    videofile = askopenfilename(initialdir = "videos")
    video = cv2.VideoCapture(videofile)
    while(True):
        ret, frame = video.read()
        print(ret)
        if ret == True:
            #cartoon1 = wf.createCartoon(frame)
            #cartoon2 = c.createCartoon(cartoon1)
            img_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            img_gray = cv2.medianBlur(img_gray, 7)
            edges = cv2.Laplacian(img_gray, cv2.CV_8U, ksize=5)
            ret, mask = cv2.threshold(edges, 100, 255, cv2.THRESH_BINARY_INV)
            cartoon2 = cv2.cvtColor(mask, cv2.COLOR_GRAY2BGR)
            cv2.imshow('Cartoon Video', cartoon2)
            if cv2.waitKey(3) & 0xFF == ord('q'):
                break
        else:
                break
    video.release()
    cv2.destroyAllWindows()
```



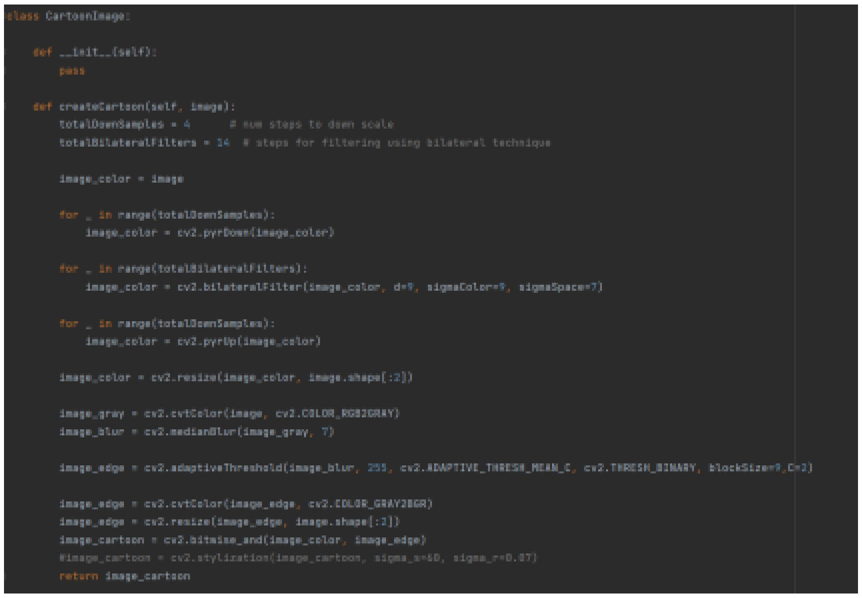
```
def createCartoon():
    img_rgb = cv2.imread(image_file)

    wf = Filter()
    cartoon1 = wf.createCartoon(img_rgb)
    cv2.imshow("cartoon1", cartoon1)
    cv2.imwrite("output/cartoon1.jpg",cartoon1)

    c = CartoonImage()
    cartoon2 = c.createCartoon(img_rgb)
    cv2.imwrite("output/cartoon2.jpg",cartoon2)
    cv2.imshow("cartoon2", cartoon2)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

uploadbutton = Button(main, text="Upload Image", command=uploadImage)
uploadbutton.place(x=350,y=100)

videobutton = Button(main, text="Upload Video", command=uploadVideo)
videobutton.place(x=500,y=100)

datasetpath = Label(main)
datasetpath.place(x=350,y=150)

cartoonbutton = Button(main, text="Cartoonify An Image", command=createCartoon)
cartoonbutton.place(x=350,y=200)

img = ImageTk.PhotoImage(Image.open("title.jpg"))
label = Label(main, image=img)
label.place(x=330,y=200)

main.config(bg='red')
main.mainloop()
```

Then the lines to upload video are followed, as this is the main method interface is created using it with all required labelled buttons and stored in the Output folder.



```python
from scipy.interpolate import UnivariateSpline
import numpy as np
import cv2

class Filter:

    def __init__(self):
        self.increment_ch_lut = self._createTable([0, 64, 128, 192, 256],[0, 70, 140, 210, 256])
        self.decrement_ch_lut = self._createTable([0, 64, 128, 192, 256],[0, 30,  80, 120, 192])

    def createCartoon(self, image):

        c_r, c_g, c_b = cv2.split(image)
        c_r = cv2.LUT(c_r, self.increment_ch_lut).astype(np.uint8)
        c_b = cv2.LUT(c_b, self.decrement_ch_lut).astype(np.uint8)
        image = cv2.merge((c_r, c_g, c_b))

        c_h, c_s, c_v = cv2.split(cv2.cvtColor(image, cv2.COLOR_RGB2HSV))
        c_s = cv2.LUT(c_s, self.increment_ch_lut).astype(np.uint8)
        return cv2.cvtColor(cv2.merge((c_h, c_s, c_v)), cv2.COLOR_HSV2RGB)

    def _createTable(self, x, y):
        spline = UnivariateSpline(x, y)
        return spline(range(256))
```

Open cv, NumPy such modules are used as we are working on the images.



```python
class CartoonImage:

    def __init__(self):
        pass

    def createCartoon(self, image):
        totalDownSamples = 4      # num steps to down scale
        totalBilateralFilters = 14  # steps for filtering using bilateral technique

        image_color = image

        for _ in range(totalDownSamples):
            image_color = cv2.pyrDown(image_color)

        for _ in range(totalBilateralFilters):
            image_color = cv2.bilateralFilter(image_color, d=9, sigmaColor=9, sigmaSpace=7)

        for _ in range(totalDownSamples):
            image_color = cv2.pyrUp(image_color)

        image_color = cv2.resize(image_color, image.shape[:2])

        image_grey = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
        image_blur = cv2.medianBlur(image_grey, 7)

        image_edge = cv2.adaptiveThreshold(image_blur, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, blockSize=9,C=2)

        image_edge = cv2.cvtColor(image_edge, cv2.COLOR_GRAY2BGR)
        image_edge = cv2.resize(image_edge, image.shape[:2])
        image_cartoon = cv2.bitwise_and(image_color, image_edge)
        #image_cartoon = cv2.stylization(image_cartoon, sigma_s=60, sigma_r=0.07)
        return image_cartoon
```

Filter and Cartoon-Image are the methods where the passed input is processed and made into a grey image by recognizing the edge by edge detection process. Then the grey image is colourized by giving it cartoon effects, not only the image but videos can also be given the Cartoon effects.

3.1.1.2 Responsibility: and 3.1.1.3 Contributions:

Sno	Name	Responsibility	Contributions Percentage
1	Sai Krishna Reddy Beluri	Implemented the proposed model in python code to convert images to cartoons and participated in the documentation and performed testing.	33.33%
2	Anvesh Chinta	Implemented the proposed model in python code to convert videos to cartoons and participated in the documentation and performed testing.	33.33%

3	Srikanth Reddy Guntaka	Validated python code for both image and video parts of code and experiment with different values to get the best effects and participated in the documentation and performed testing.	33.33%

3.1.1.4 Issues/Concerns:

Even though the model is implemented it has its fair share of limitations to implement this model into a full scale working commercial model. A few of them are listed below.

- It takes a little more time to process images.
- Underdeveloped database layers
- We don't have many design choices to choose for designing the application.
- Not fully compatible in order to work on mobile/tablet.

References:

[1] P. Barla, J. Thollot and L. Markosian, “X-Toon: an extended toon shader”, Proc. 4th international symposium on Non-Photorealistic animation and rendering, Annecy, pp.127-132, 2006.

[2] H. Winnemoller, S. C. Olsen and B. Gooch, “Real-Time Video Abstraction”, In Proc. SIGGRAPH2006, pp.1221-1226, 2006.

[3] PhiliippeDecaudin, “Cartoon-Looking Rendering of 3D Scenes”, Research Report INRIA#2919, 1996.

[4] https://projectgurukul.org/cartooning-image-opencv-python/

[5] http://www.toonyphotos.com/

[6] https://www.geeksforgeeks.org/cartooning-an-image-using-opencv-python/

Various other documents and sites on the internet