Dharmsinh Desai University,Nadiad

Faculty of Technology

Department of Computer Engineering

B. Tech. CE Semester – VI

Subject: System Design Practice

Topic: Image Enhancement

1) Het Desai, roll no: CE001, Id: 19CEUON74

2) Smit Bhoraniya, roll no: CE017, Id:19CEUOS057

Guided by: Prof. Shaifali P. Malukani

DHARMSINH DESAI UNIVERSITY

NADIAD-387001, GUJARAT

CERTIFICATE

This is to certify that the project entitled as "**Image enhancement** " is a bonafide report of the work carried out by

1) Mr. Het Desai, Student ID No: **19CEUON074**

2) Mr. Smit Bhoraniya, Student ID No:**19CEUOS057**

of Department of Computer Engineering, semester VI, under the guidance and supervision of **Prof. Shaifali P. Malukani** for the subject System Design Practice during the academic year 2019-2020.

Project Guide Head of the Department

Assistance Professor
Prof. Shaifali .P. Malukani

Department of Computer
Engineering,
Faculty of Technology,
Dharmsinh Desai University,
Nadiad

Dr. C.K Bhensdadia
Prof. & Head,

Department of Computer
Engineering,
Faculty of Technology,
Dharmsinh Desai University,
Nadiad

Contents:

Abstract

- Image enhancement is the desired improvement of image quality.

- Physiological experiments have shown that very small changes in luminanceare recognized by the human visual system in regions of continuous grey- tones and are not seen at all in regions of some discontinuities

- Therefore, a design goal for image enhancement is often to smooth imagesin more uniform regions but to preserve edges.

- Conversely, it has also been shown that somehow degraded images with enhancement of certain features can simplify image interpretation both for ahuman observer and for machine recognition.

- A second design goal, therefore, is image sharpening .

Introduction

- The principal objective of image enhancement is to process a given image so that the result is more suitable than the original image for aspecific application.

- There are two kinds of methods use 1) spatial filtering and 2) Pseudo-color image processing

- Spatial filtering includes Smoothing filters , Sharpening filters

- It accentuates or sharpens image features such as edges, boundaries, or contrast to make a graphic display more helpful for display and analysis.
- Enhancement in the frequency domain

- Pseudo-color image processing

- The enhancement doesn't increase the inherent information content of the data, but it increases the dynamic range of the chosen features so that they can be detected easily.

Technologies/tools used :

- Platform used: Visual Studio 2019
- Test application: Android studio
- Test application: Expo Go
- Storage: Cloudinaryserver
- Frontend: React Native
- Backend: Flask

Software Requirement Specifications:

Image Enhancement

R 1.1 Open app

Description: Useropenstheapp here

R 1.1.1 Select Image Description: User selects image here
Input: selection from gallery
Output: taken image

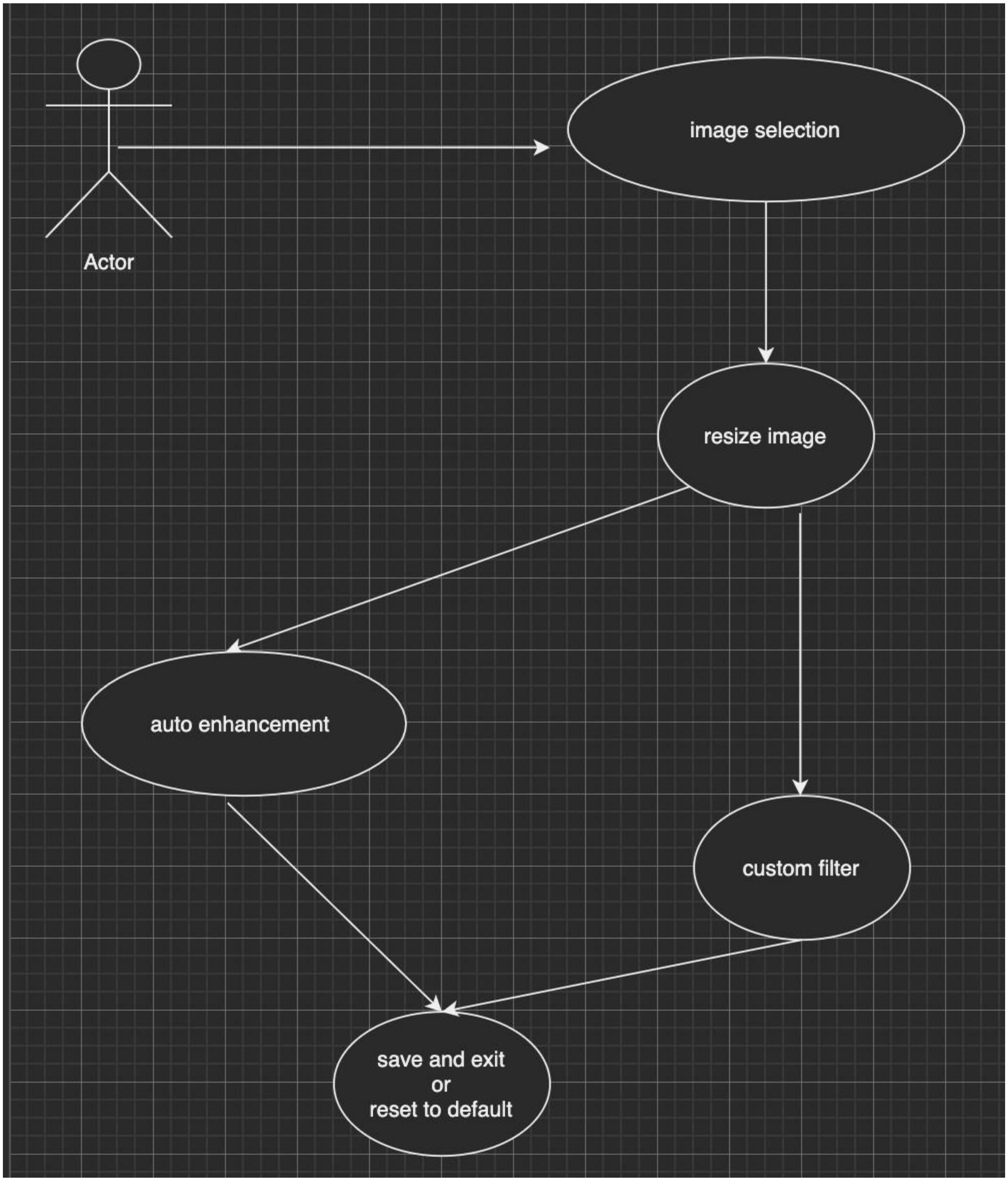
R 1.1.2 crop image
Description: options to resizeimage here
Input: selected image
Output: loadingwidget

R 1.1.3 enhancement / filter
Description: can choose for auto enhancement or custom filters
Input: resized image
Output: processed image

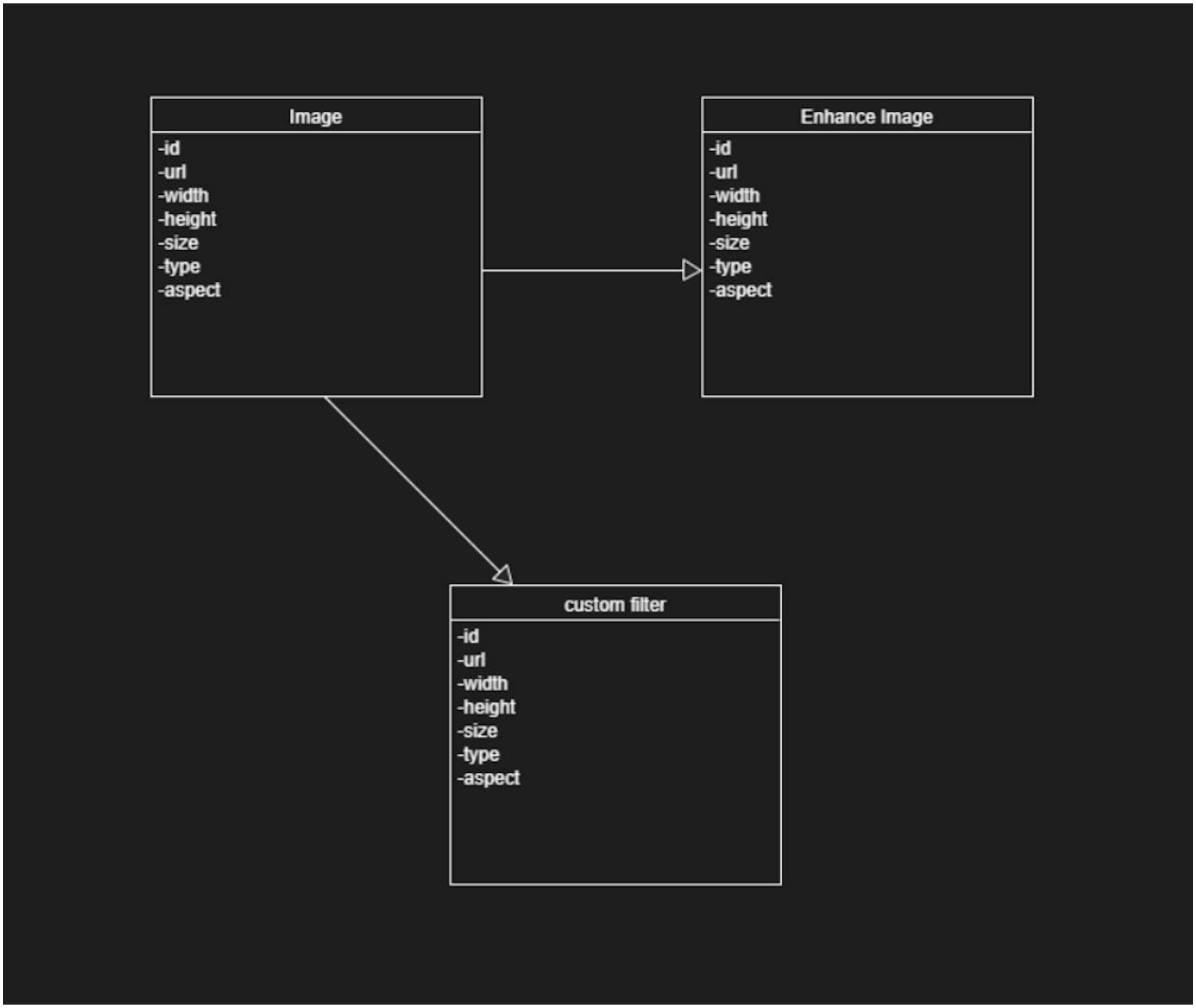
R 1.1.4 Processed image
Description: user can save or reset to default
Input: give image to model
Output: showing saved or defaultimage

Design:

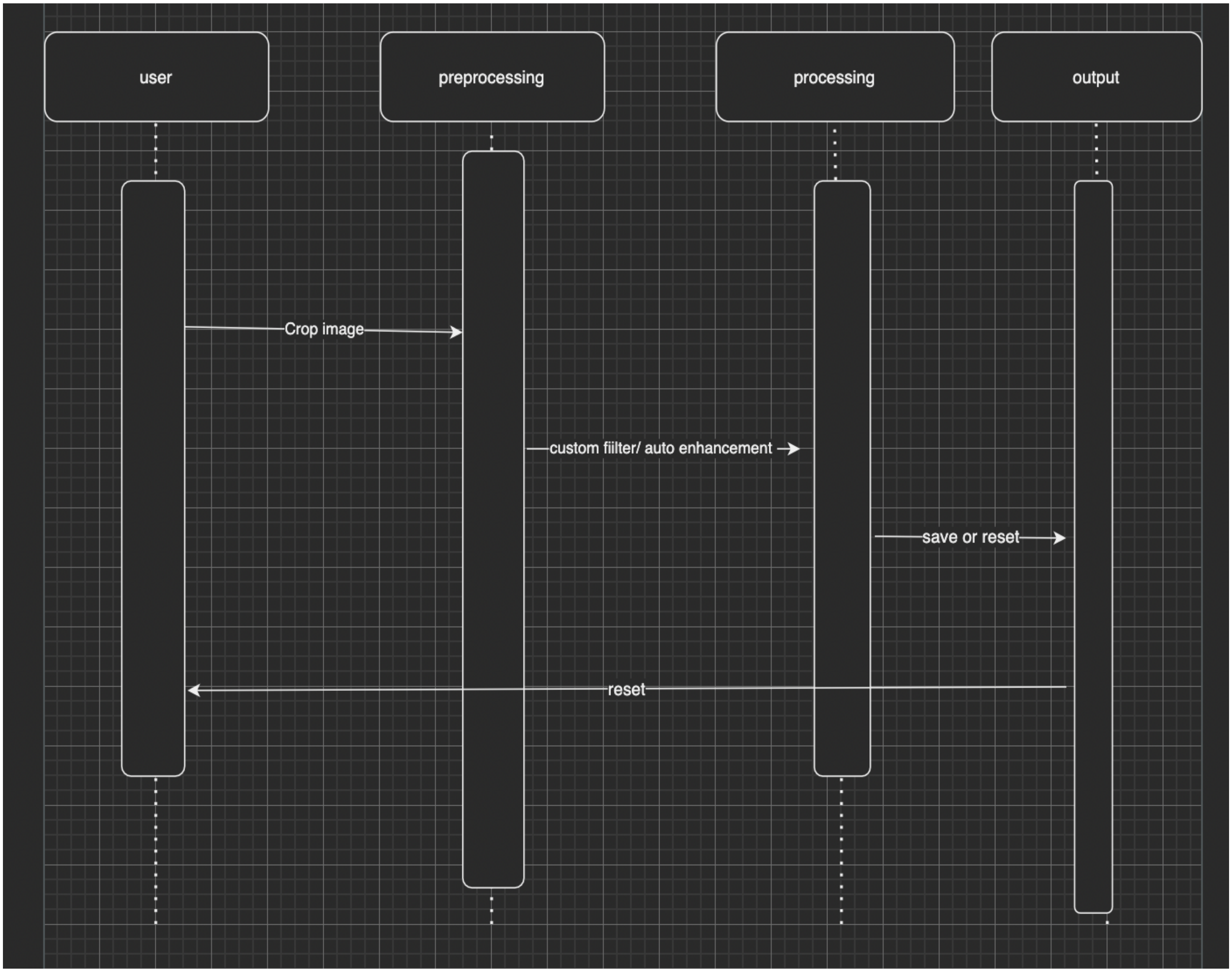
- Use Case diagram



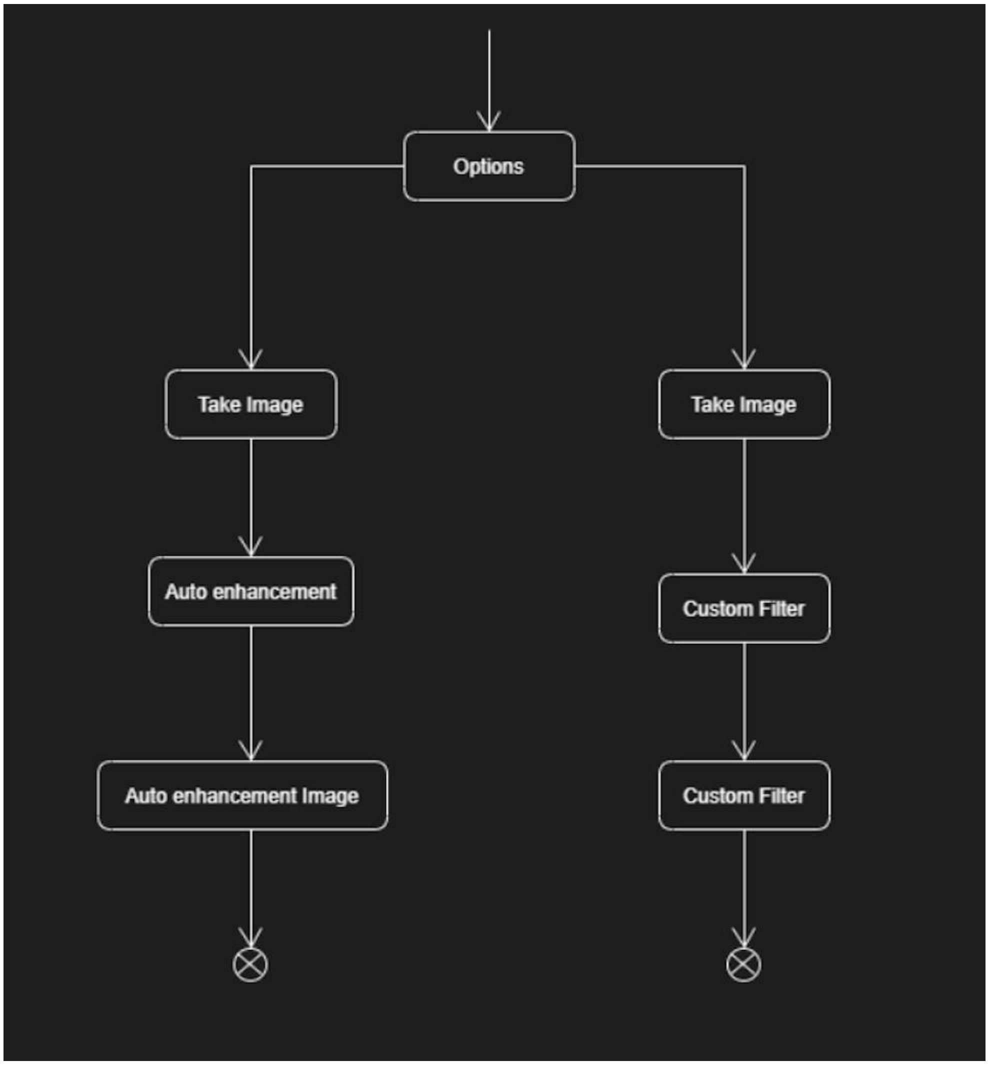
- Class Diagram



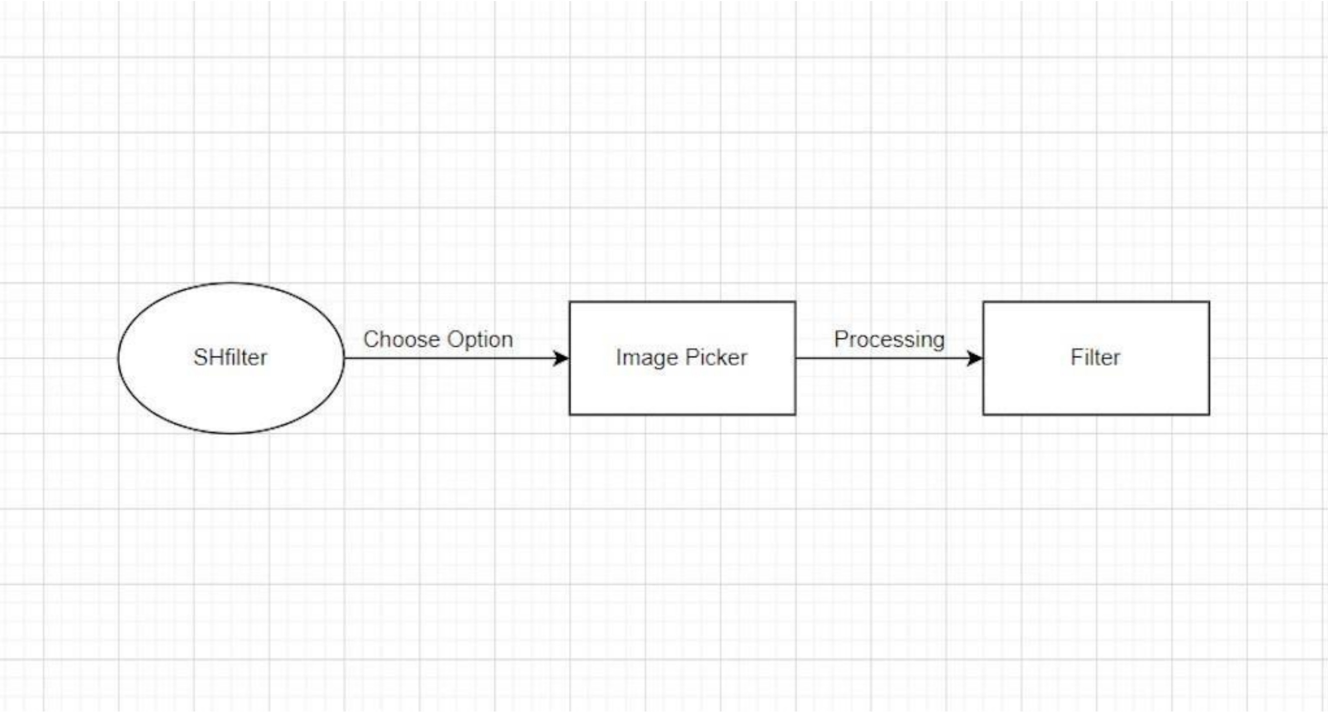
- Sequence Diagram

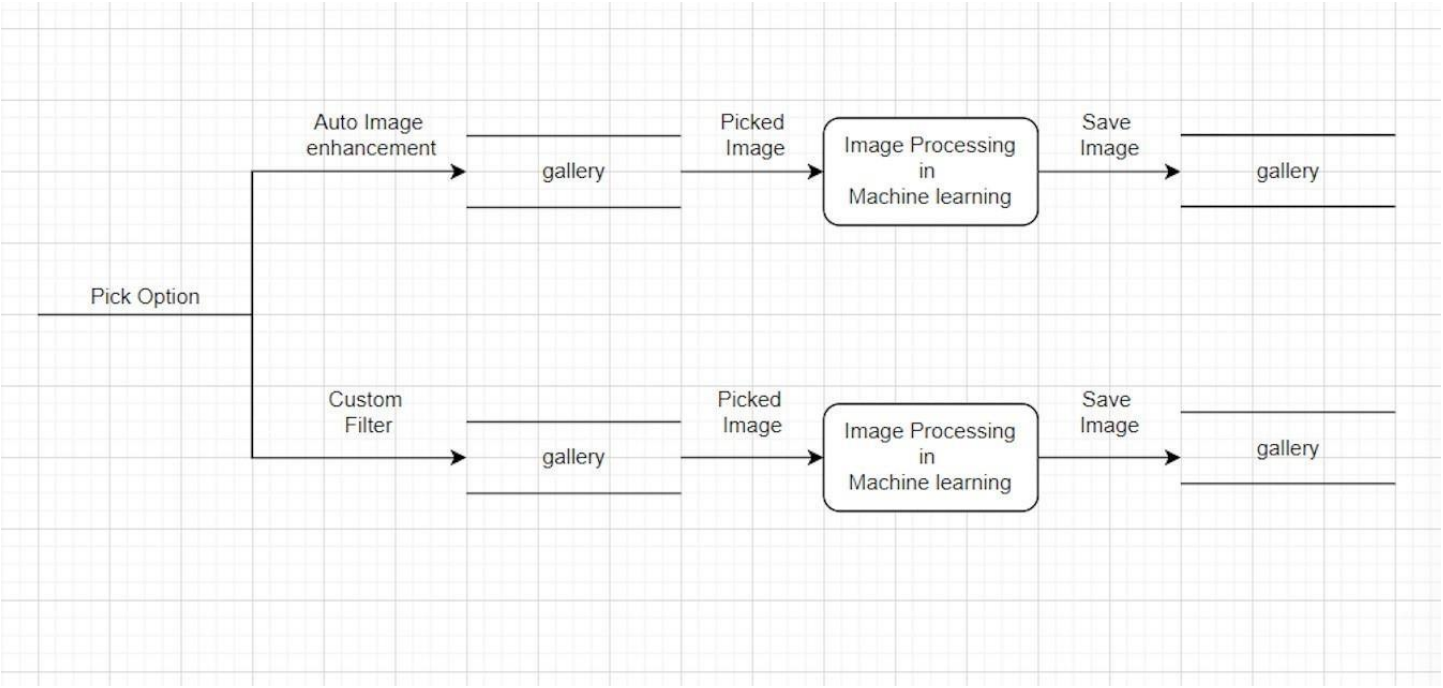


- Activity Diagram



- Data Flow Diagram




Pick Option
Auto Image enhancement
gallery
Picked Image
Image Processing in Machine learning
Save Image
gallery
Custom Filter
gallery
Picked Image
Image Processing in Machine learning
Save Image
gallery

Implementation Detail:

1. Modules:

In the following section a brief description of each module is given. Related screenshots are attached in separate sections.

Preference -module:

In these module the user selects mode of editing they want to use whether auto enhancement or custom filters. It will give an options for editing.

Resize image-module:

To meet the input requirements of the functionality the user needs to adjust in the image in a specific sized format.

Auto enhancement -module:

Here complex machine learning algorithms with pre-trained datasets auto enhances the input image and gives an enhanced image

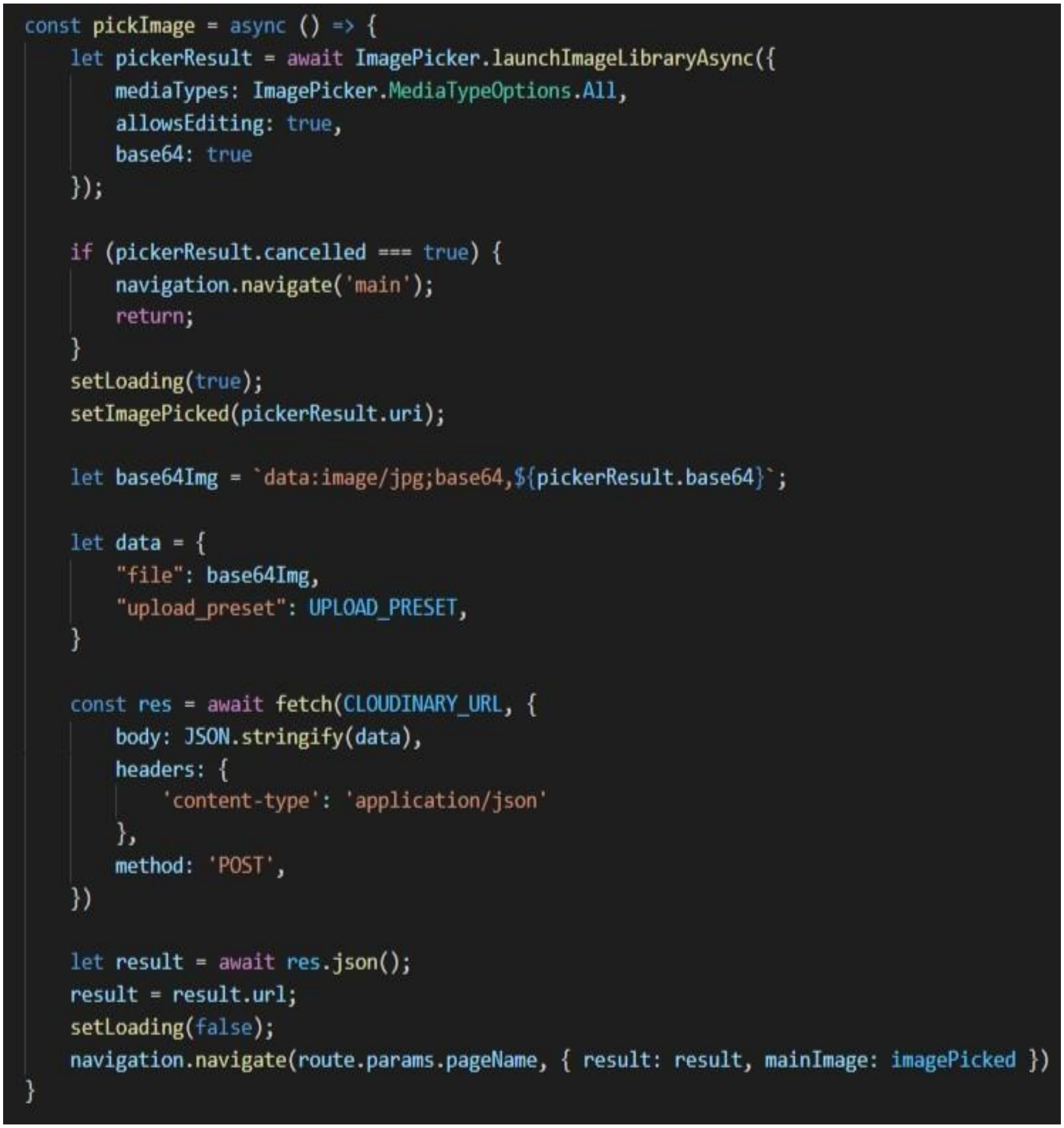
Custom-filters-module:

Here there are various kinds of filters option that user could scroll and adjust according to there need and requirement which also works on complex machine learning algorithm and that uses pre trained datasets to improve the efficiency of these module.

2. Major Functions prototypes

1. Pick Image from Gallery:

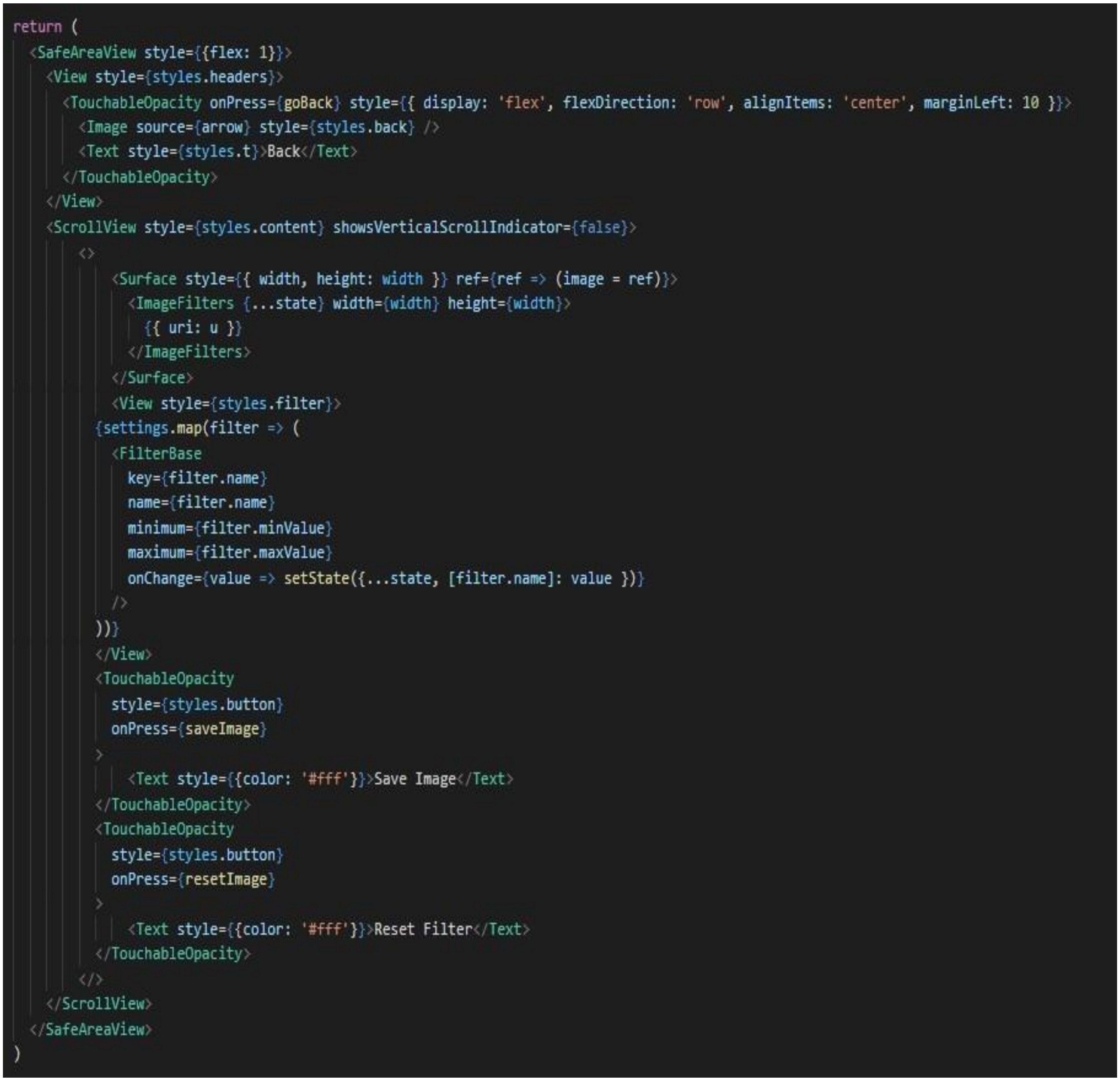
we can pick an Image from gallery and use it in image filtering. Here we used ImagePicker library which help us to get image from gallery, we can take any type of image over here like JPG, PNG etc. if get the image we convert into base64 format. If we need to pass image any where in string form this format helps us, after that we pass image to our cloud.



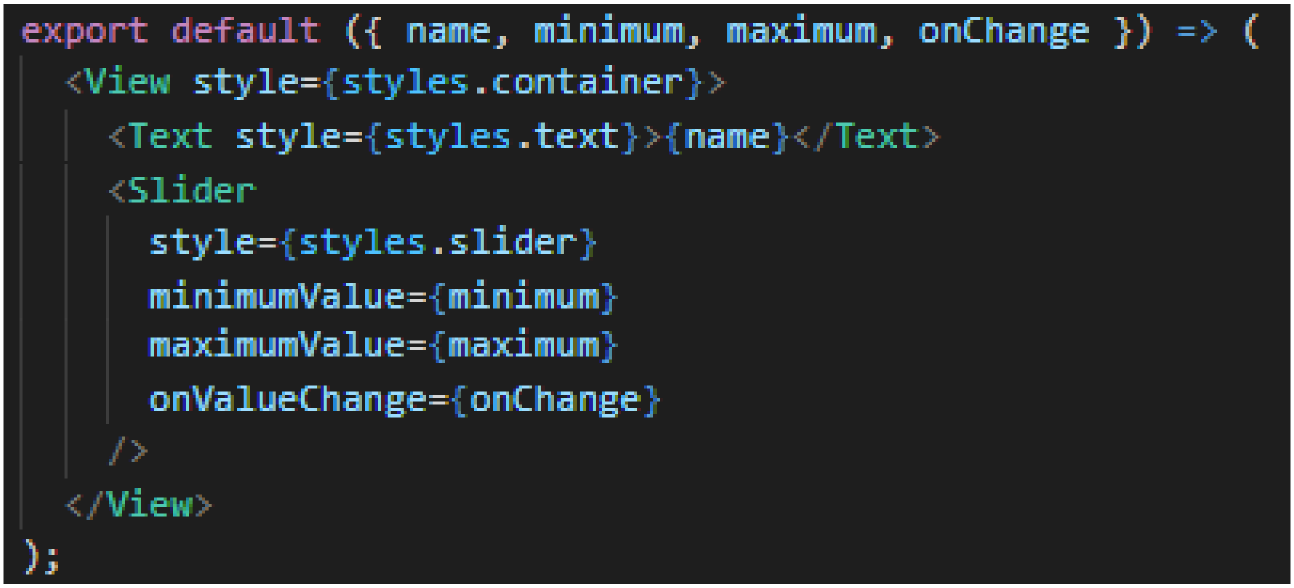
```
const pickImage = async () => {
    let pickerResult = await ImagePicker.launchImageLibraryAsync({
        mediaTypes: ImagePicker.MediaTypeOptions.All,
        allowsEditing: true,
        base64: true
    });

    if (pickerResult.cancelled === true) {
        navigation.navigate('main');
        return;
    }
    setLoading(true);
    setImagePicked(pickerResult.uri);

    let base64Img = `data:image/jpg;base64,${pickerResult.base64}`;

    let data = {
        "file": base64Img,
        "upload_preset": UPLOAD_PRESET,
    }

    const res = await fetch(CLOUDINARY_URL, {
        body: JSON.stringify(data),
        headers: {
            'content-type': 'application/json'
        },
        method: 'POST',
    })

    let result = await res.json();
    result = result.url;
    setLoading(false);
    navigation.navigate(route.params.pageName, { result: result, mainImage: imagePicked })
}
```

2. Custom Image Filter:

We are using React-gl-image-filter library for custom image filter Taken image will show over here for live image filter changing and filterBase is show specific filter like blur, saturation, etc. and they have minimum and maximum values for limitation.



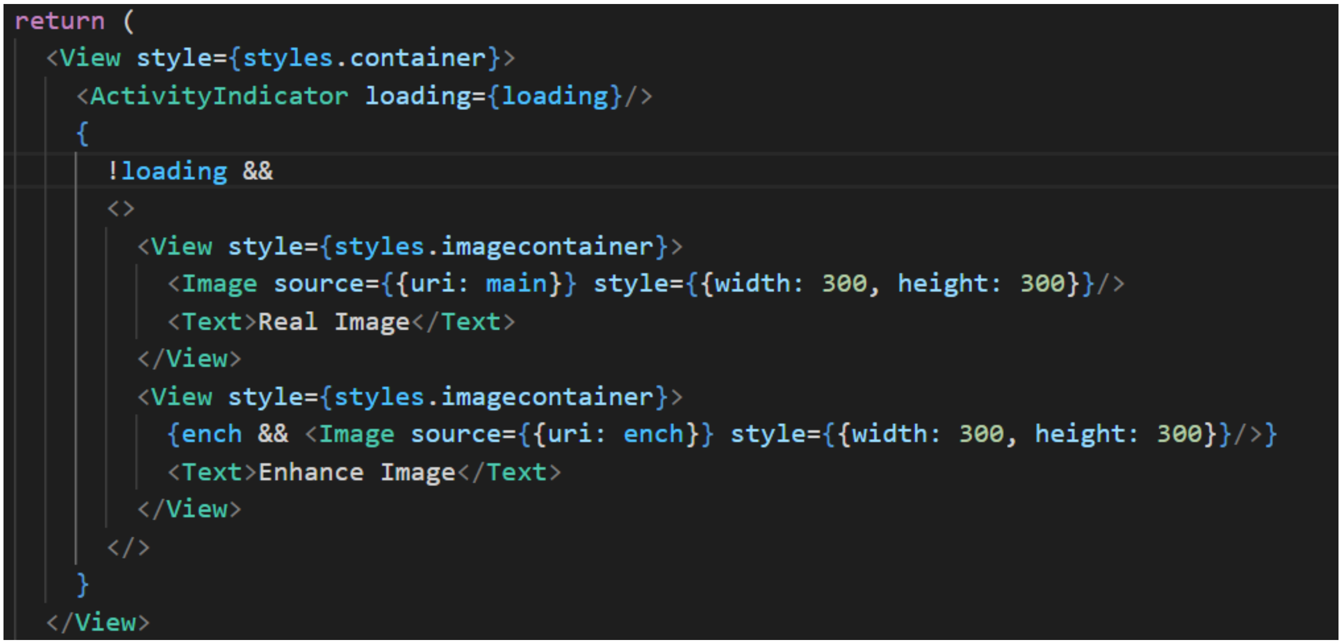
```
return (
  <SafeAreaView style={{flex: 1}}>
    <View style={styles.headers}>
      <TouchableOpacity onPress={goBack} style={{ display: 'flex', flexDirection: 'row', alignItems: 'center', marginLeft: 10 }}>
        <Image source={arrow} style={styles.back} />
        <Text style={styles.t}>Back</Text>
      </TouchableOpacity>
    </View>
    <ScrollView style={styles.content} showsVerticalScrollIndicator={false}>
        <>
            <Surface style={{ width, height: width }} ref={ref => (image = ref)}>
              <ImageFilters {...state} width={width} height={width}>
                {{ uri: u }}
              </ImageFilters>
            </Surface>
            <View style={styles.filter}>
          {settings.map(filter => (
            <FilterBase
              key={filter.name}
              name={filter.name}
              minimum={filter.minValue}
              maximum={filter.maxValue}
              onChange={value => setState({...state, [filter.name]: value })}
            />
          ))}
          </View>
          <TouchableOpacity
            style={styles.button}
            onPress={saveImage}
          >
              <Text style={{color: '#fff'}}>Save Image</Text>
          </TouchableOpacity>
          <TouchableOpacity
            style={styles.button}
            onPress={resetImage}
          >
              <Text style={{color: '#fff'}}>Reset Filter</Text>
          </TouchableOpacity>
        </>
    </ScrollView>
  </SafeAreaView>
)
```



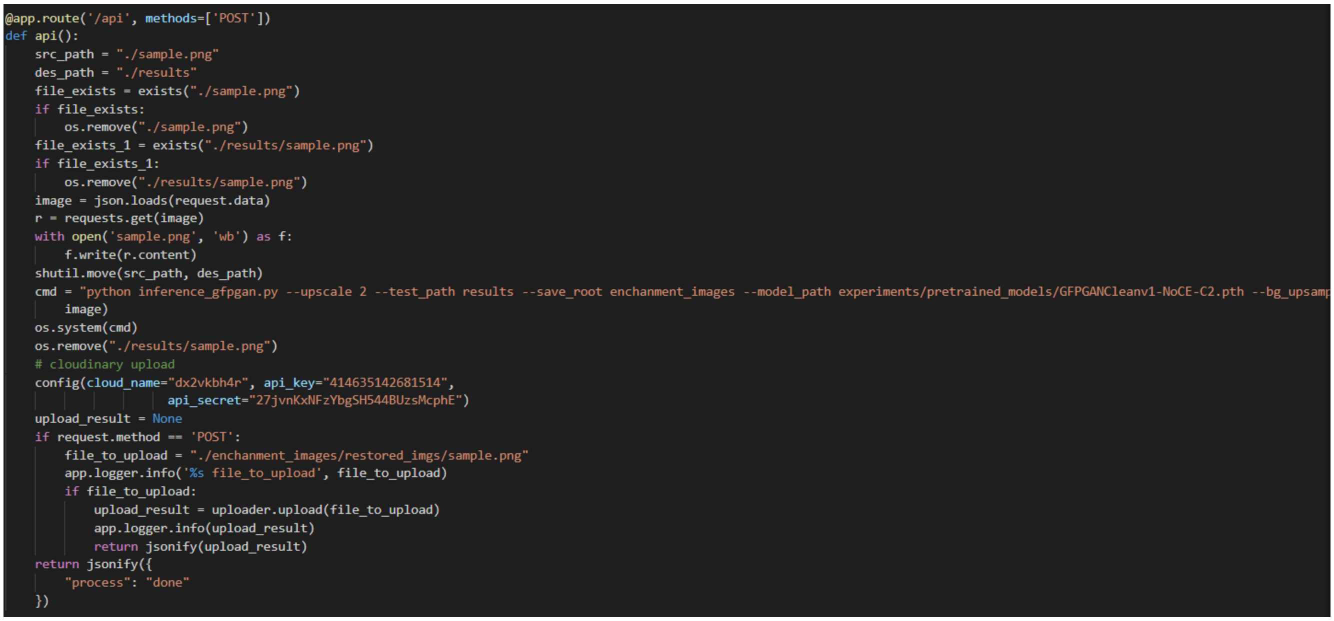
```
export default ({ name, minimum, maximum, onChange }) => (
  <View style={styles.container}>
    <Text style={styles.text}>{name}</Text>
    <Slider
      style={styles.slider}
      minimumValue={minimum}
      maximumValue={maximum}
      onValueChange={onChange}
    />
  </View>
);
```

3. Auto Image Enhancement:

It will take image from front end, use the enhancement model and enhance the image, it is trained model. Basically, it uses smoothing, beauty and many more filters for enhancement. In back end we use our model for auto image enhancement, first we check if image is available or not, than we use our model and we give and enhance image. Also we can download it.



```
return (
  <View style={styles.container}>
    <ActivityIndicator loading={loading}/>
    {
      !loading &&
      <>
        <View style={styles.imagecontainer}>
          <Image source={{uri: main}} style={{width: 300, height: 300}}/>
          <Text>Real Image</Text>
        </View>
        <View style={styles.imagecontainer}>
          {ench && <Image source={{uri: ench}} style={{width: 300, height: 300}}/>}
          <Text>Enhance Image</Text>
        </View>
      </>
    }
  </View>
```



```python
@app.route('/api', methods=['POST'])
def api():
    src_path = "./sample.png"
    des_path = "./results"
    file_exists = exists("./sample.png")
    if file_exists:
        os.remove("./sample.png")
    file_exists_1 = exists("./results/sample.png")
    if file_exists_1:
        os.remove("./results/sample.png")
    image = json.loads(request.data)
    r = requests.get(image)
    with open('sample.png', 'wb') as f:
        f.write(r.content)
    shutil.move(src_path, des_path)
    cmd = "python inference_gfpgan.py --upscale 2 --test_path results --save_root enchanment_images --model_path experiments/pretrained_models/GFPGANCleanv1-NoCE-C2.pth --bg_upsamp
        image)
    os.system(cmd)
    os.remove("./results/sample.png")
    # cloudinary upload
    config(cloud_name="dx2vkbh4r", api_key="414635142681514",
                      api_secret="27jvnKxNFzYbgSH544BUzsMcphE")
    upload_result = None
    if request.method == 'POST':
        file_to_upload = "./enchanment_images/restored_imgs/sample.png"
        app.logger.info('%s file_to_upload', file_to_upload)
        if file_to_upload:
            upload_result = uploader.upload(file_to_upload)
            app.logger.info(upload_result)
            return jsonify(upload_result)
    return jsonify({
        "process": "done"
    })
```

3. Algorithm:

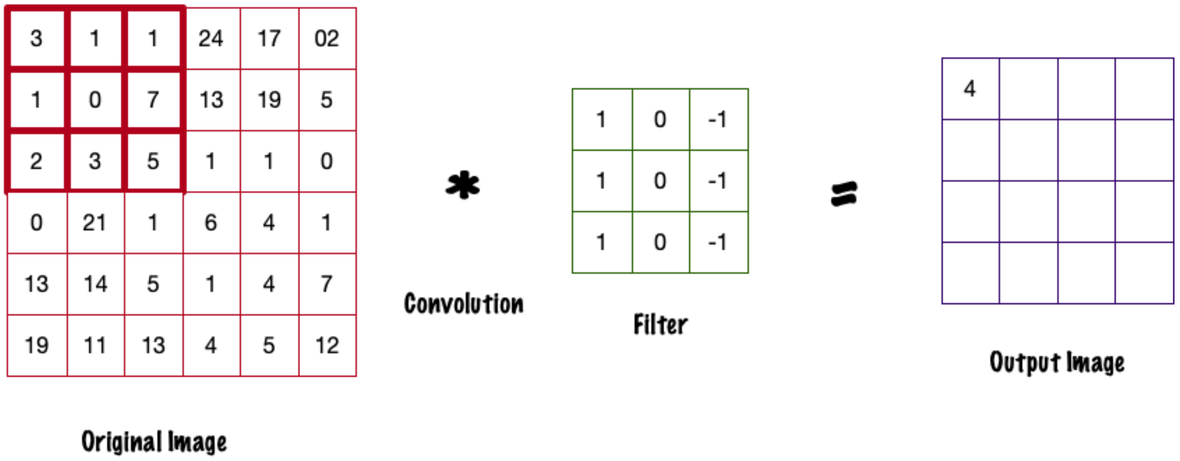
Today, several machine learning image processing techniques leverage deep learning networks. These are a special kind of framework that imitates the human brain to learn from data and make models. One familiar neural network architecture that made a significant breakthrough on image data is Convolution Neural Networks, also called CNNs. Now let’s look at how CNNs are utilised on images with different image processing tasks to build state of the art models.

The convolutional neural network is built on Two primary layers, which are:

1) Convolutional layer
2) Pooling layer

• Convolutional Layer:

The Convolution layer is the heart of CNN’s, it does most of work Most of the work in identifying the features in the given image. Then in the convolution layer, we consider square blocks of some random size of the input image and apply the dot product with the filter(random filter size). If the two matrices(the patch and the filter) have high values in the same positions, the convolution layer output will be high(which gives the bright side of the image). If they don’t, it will be low(the dark side of the image). In this way, a single value of the output of the dot product can tell us whether the pixel pattern in the underlying image matches the pixel pattern expressed by our filter. Let’s review this via an example, where we want to apply a filter to detect vertical edges from an image using convolution and see how the math works.



3	1	1	24	17	02
1	0	7	13	19	5
2	3	5	1	1	0
0	21	1	6	4	1
13	14	5	1	4	7
19	11	13	4	5	12

1	0	-1
1	0	-1
1	0	-1

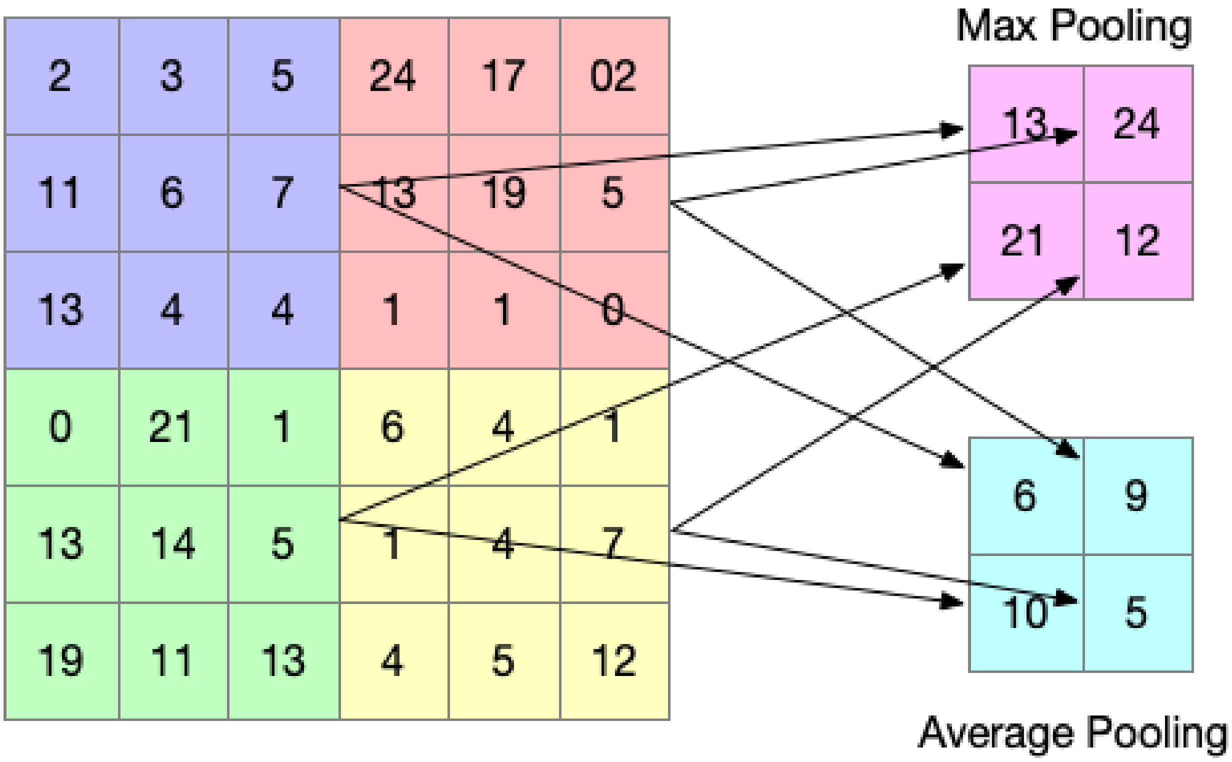
4			

- **Pooling Layer:**

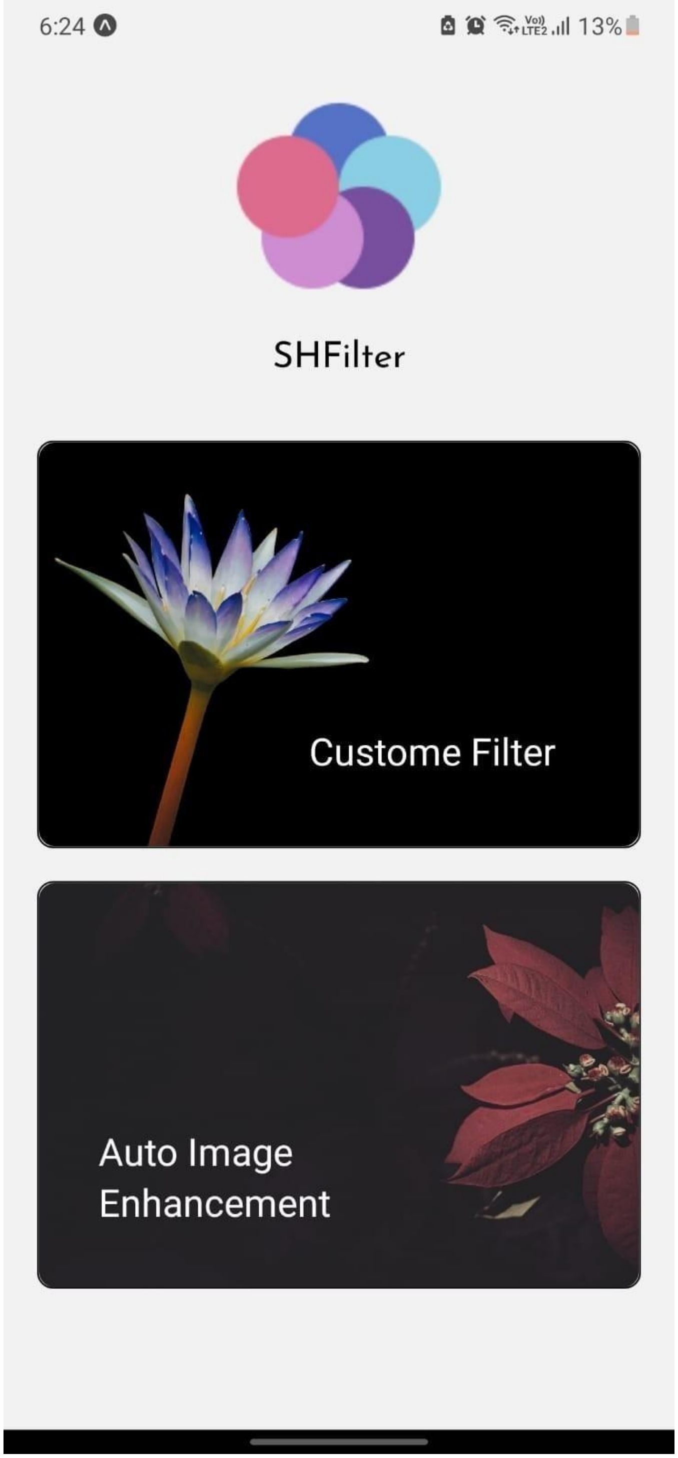
When we identify the features using the convolutional layers, we have multiple feature maps. These feature maps result when the convolutional operation is applied between the input image and the filter. Hence we need one more operation which downsamples the image. Hence to make the learning process easy for the network, the pixel values in the arrays are reduced by using the "pooling" operation. They operate autonomously on every depth slice of the input and resize it spatially, using the two different operations:

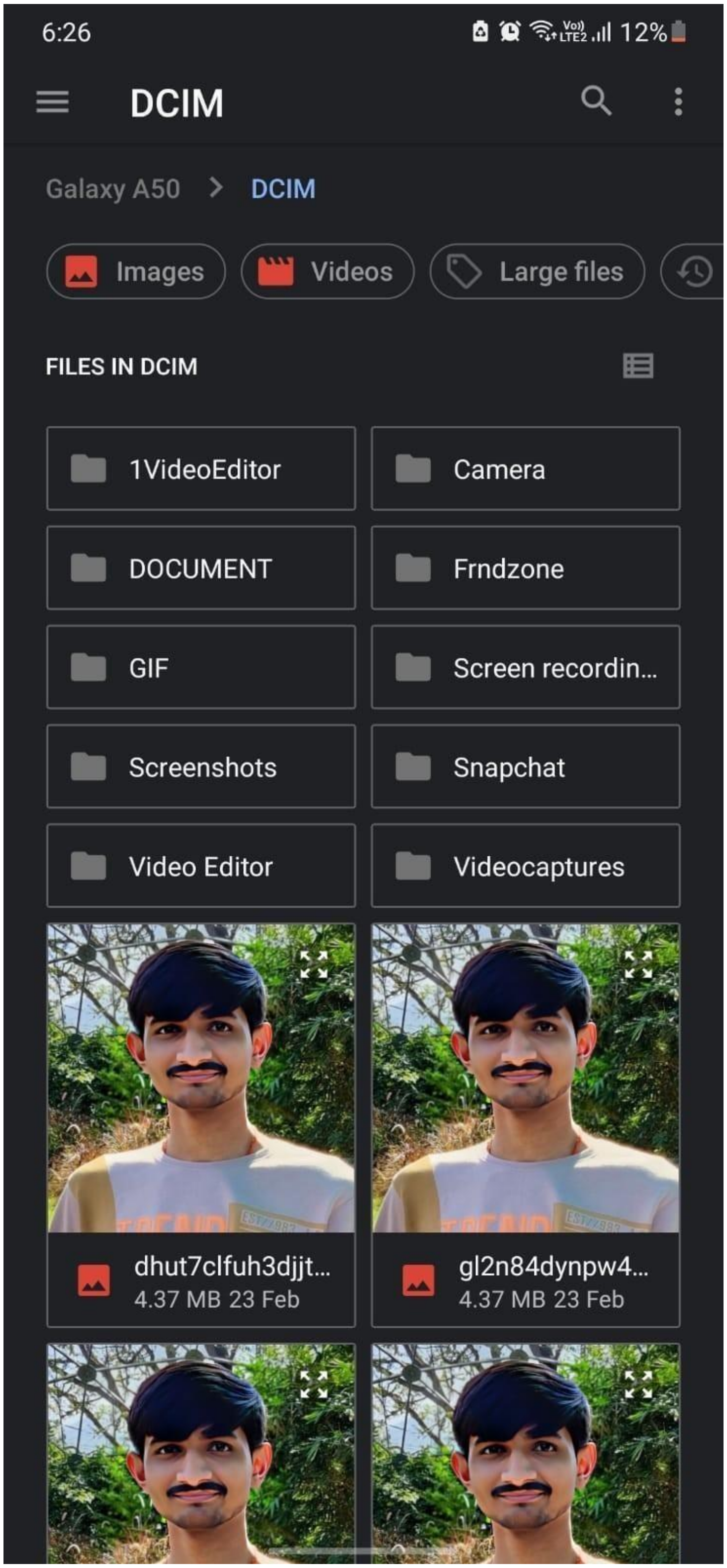
- Max Pooling - returns the maximum value from the array of the image covered by the Kernel
- Average Pooling - returns the average of all the values from the array of the image covered by the Kernel.

Below is an example of how a pooling operation is computed on the given pixel array.


2 3 5 24 17 02
11 6 7 13 19 5
13 4 4 1 1 0
0 21 1 6 4 1
13 14 5 1 4 7
19 11 13 4 5 12
Max Pooling
13 24
21 12
6 9
10 5
Average Pooling

Screen-Shots:





6:26 12%

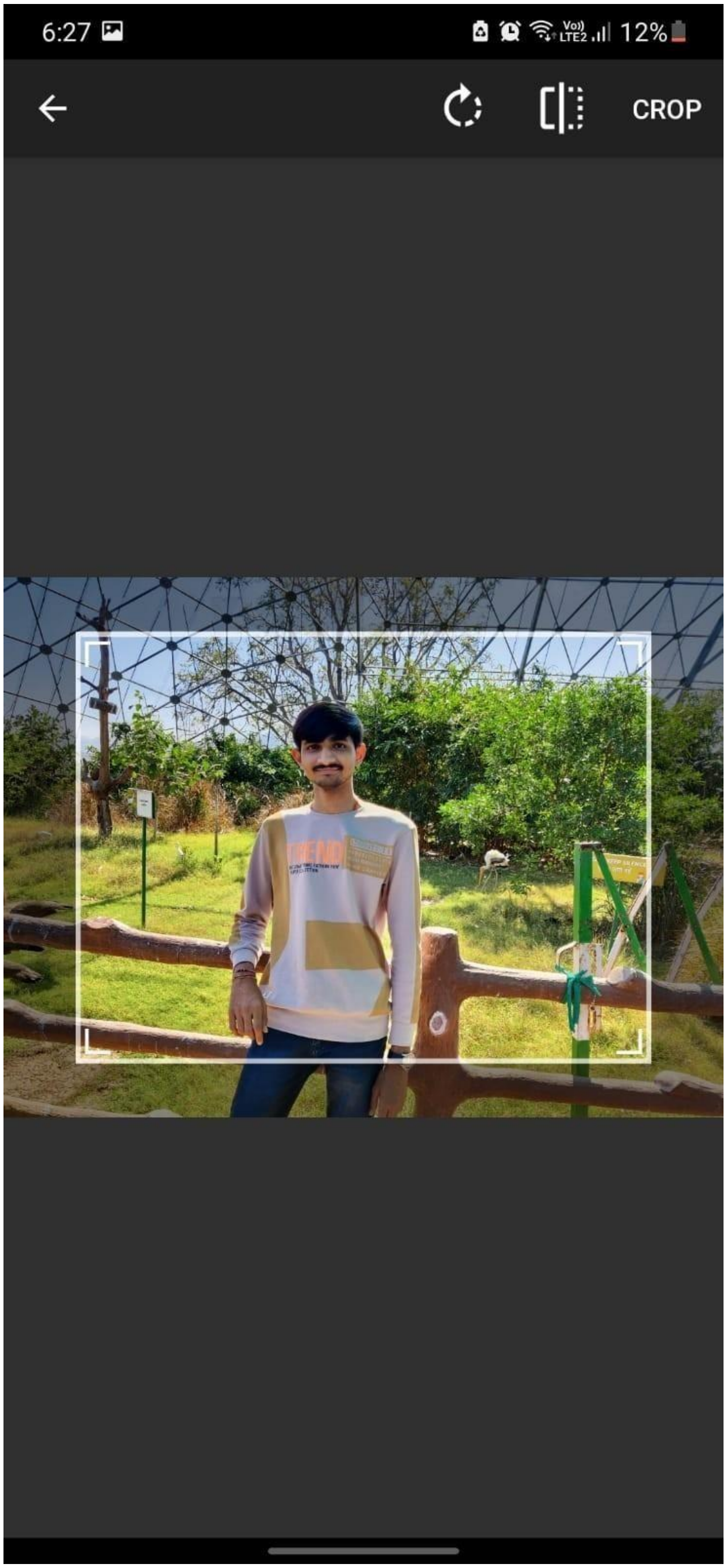
DCIM

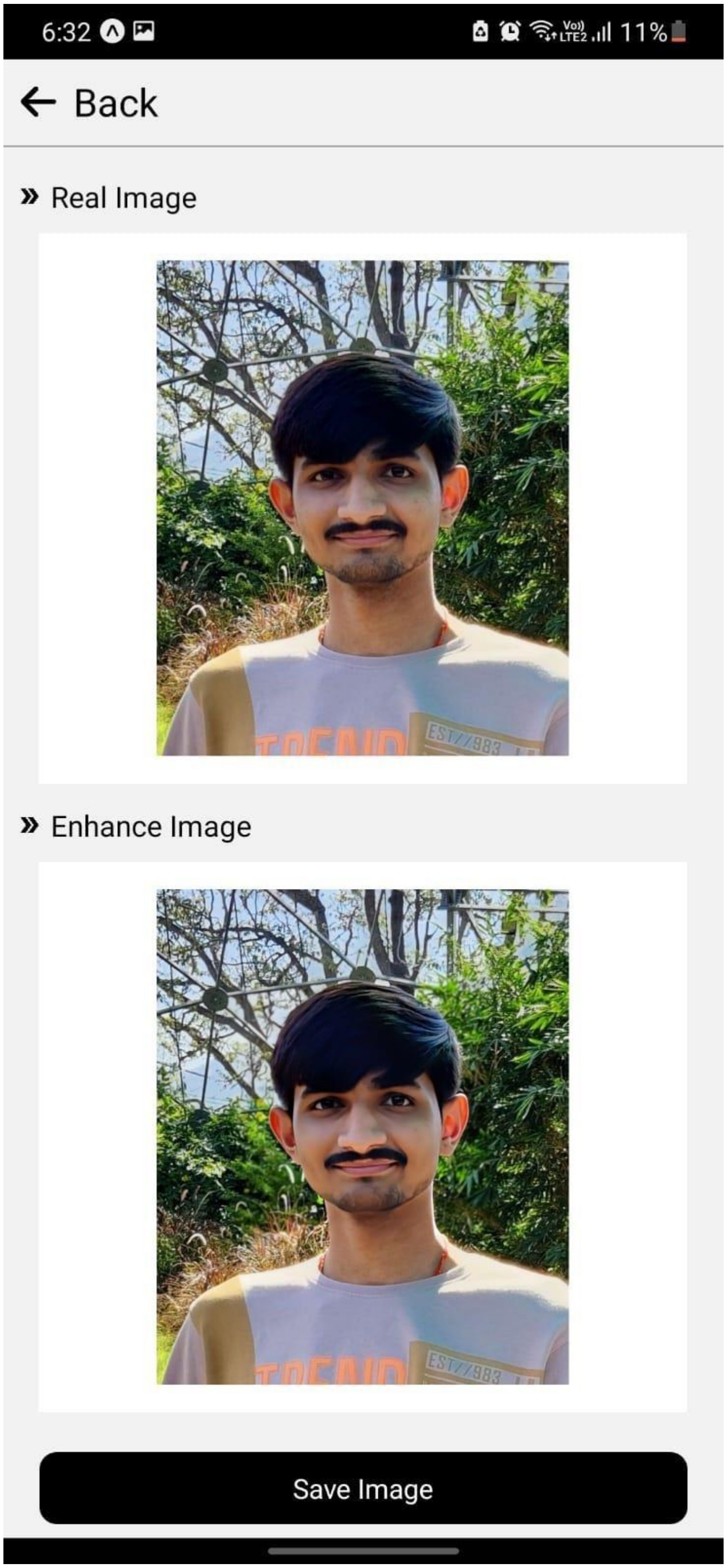
Galaxy A50 > DCIM

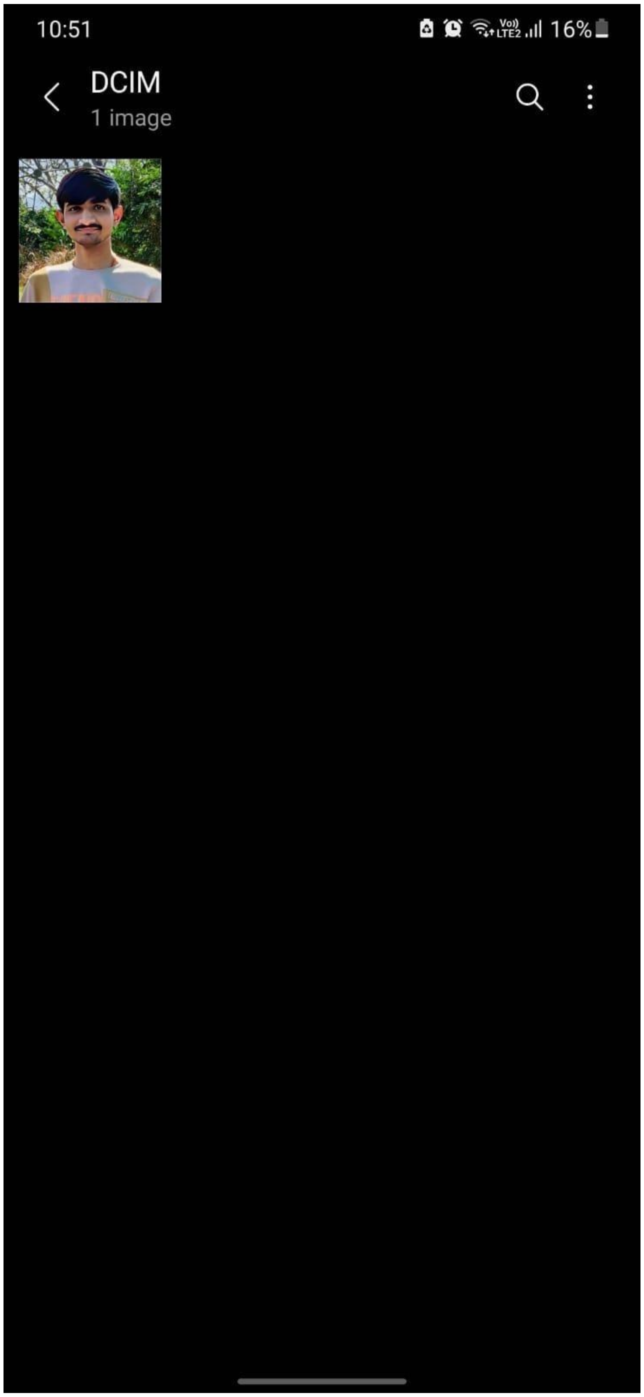
Images Videos Large files

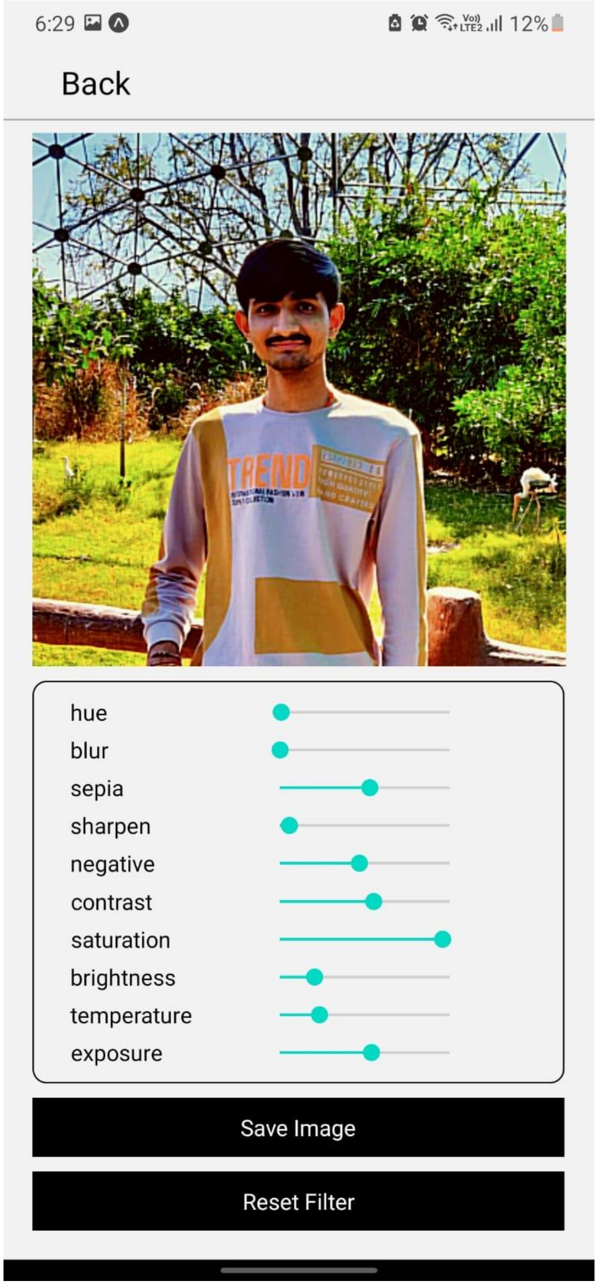
FILES IN DCIM

1VideoEditor	Camera
DOCUMENT	Frndzone
GIF	Screen recordin...
Screenshots	Snapchat
Video Editor	Videocaptures
dhut7clfuh3djjt... 4.37 MB 23 Feb	gl2n84dynpw4... 4.37 MB 23 Feb


6:27
12%
CROP


6:32
11%
Back
» Real Image
» Enhance Image
Save Image


10:51
16%
DCIM
1 image



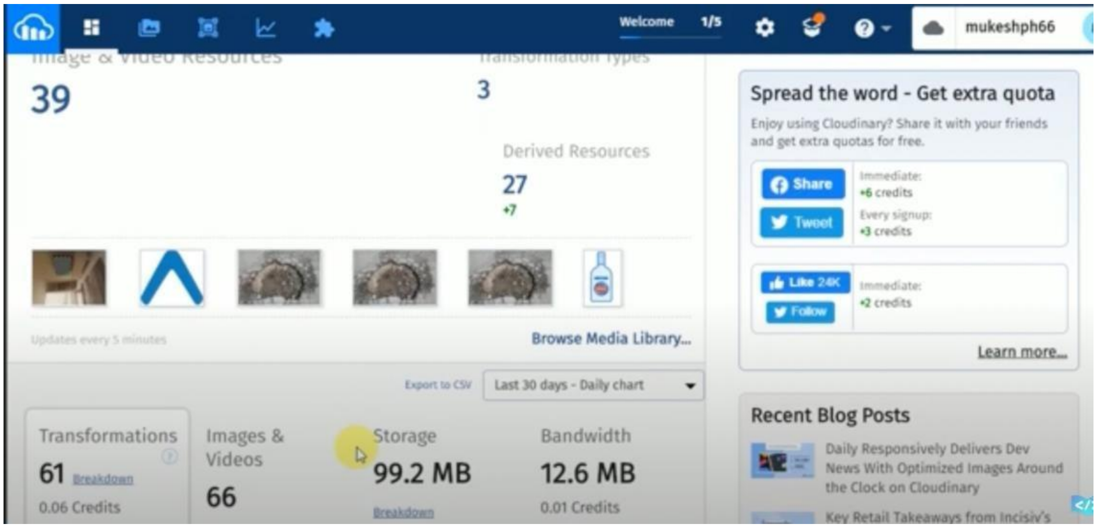


Image & Video Resources
39

Transformation Types
3

Derived Resources
27
+7

Updates every 5 minutes

Spread the word - Get extra quota

Enjoy using Cloudinary? Share it with your friends and get extra quotas for free.

Immediate:
+6 credits
Every signup:
+3 credits

Immediate:
+2 credits

Transformations
61
0.06 Credits

Images & Videos
66

Storage
99.2 MB

Bandwidth
12.6 MB
0.01 Credits

Recent Blog Posts

Daily Responsively Delivers Dev News With Optimized Images Around the Clock on Cloudinary

Key Retail Takeaways from Incisiv's

Folder Structure of the Project

❖ Front-End: -

- ➢ **MainPage.js**
- ➢ **TakeImage.js**
- ➢ **ImageEnch.js**
- ➢ **CustomFilter.js**
    - **CustomFilterBase.js**

❖ Back-End: -

- ➢ **Server.py**
    - **inference_gfpgan.py**
        - **GFPGANCleanv1-NoCE-C2.pth**

Ownership of Module:

Here all the activities and detailed work distribution is given.

1) Smit Bhoraniya: -
    - Back-end Development

2) Het Desai: -
    - Front-end Development

Conclusion:

Hence-forth in this project we have successfully implemented the Auto enhancement & custom filters functionality, user will select an image that he wants to edit and will select the mode of editing. So he would be asked to resize the image according to the application requirements . Then the resized image will be ready to transformation using few complex machine learning algorithms and would be ready for use. If he is not satisfied with the editing work he could roll back the image to default or could modify the changes accordingly .

Limitations:

1) This project is suitable for some basic level photo editing and won’t be much efficient for editing videos.

2) We can’t use these app inside other app to use its custom filters.

Future Extension:

To take over the limitations we are planning this future extension in our system.

1) Creating a chatting room where you could chat through Images.
2) Adding more editing options , using some kind of augmented reality, facial filters

Bibliography

References/resources used for developing project:

- https://github.com/TencentARC/GFPGAN.git
- https://docs.expo.dev/
- https://reactnative.dev/
- https://www.npmjs.com/package/react-native-gl-image-filters

Online Database

- https://cloudinary.com/