**Artistify**

Jamoliddinkhuja Odilkhujaev: U1610092

Multimedia Computing, Final Assignment

Computer Science and Software Engineering, Inha University in Tashkent

E-mail: [j.odilkhujaev@student.inha.uz](mailto:j.odilkhujaev@student.inha.uz)

Project link: <https://github.com/Green-Blood/Artistify>

Abstract

Python is becoming enormously popular programming language because it is a free, high-level language, that has a wide set of free libraries. One of the reasons to rise is the growing interest in Computer vision and image processing techniques. This paper provides basic image processing techniques which have been used in applications such as Instagram, Snapchat and etc. Besides image processing techniques, face, age, and parts of the face recognition is discussed in this paper. The basic description of the libraries and algorithms used with the examples and the results are provided. This application will be developed using Flask  micro web framework written in Python and for UI basic bundle of HTML5, CSS3, and JavaScript is used.

**Keywords:** face, detection, parts, age, gender, recognition, system, OpenCV

1. Introduction

Image processing and computer vision (CV) began by development in C/C++ and theusage of MATLAB software. Desptire the fact that MATLAB offers an proficient significant level platform for prototyping and testing algorithms, its performance does not rival with a well designed and optimised C/C++ implementation. As of late, potential and valuable solutions have developed for improving image processing and computer vision algorithms in Python. For image processing or computer vision development, two Python libraries are prominently used: NumPy/SciPy and OpenCV with a Python wrapper. This paper looks at using basic computer vision routines from OpenCV, and it will be developed using Flask  micro web framework written in Python and for UI basic bundle of HTML5, CSS3, and JavaScript will be used. Moreover, several face recognition techniques are going to be used.

Python is an interpreted high-level general-purpose programming language created by Guido van Rossum in 1991. It’s design philosophy is a emphasising the code readability. Python supports multiple programming paradigms including object-oriented, imperative, functional and procedural and has a large standard and comprehensive library. At the time of writing this paper the latest version is Python 3.8.2. Python is a good choice for all the researchers in the scientific community due to the following reasons [1]:

1. Scripting language, means it is interpreted, so the parts of the code can be run in the different cells.
2. Fast to learn
3. Full of free libraries
4. Useful in a wider setting than the other languages: scientific computing, scripting, web sites, etc.

OpenCV - is by far the most capable and most commonly used computer vision library. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source BSD license. It is written in C/C++, but Python bindings are added during the installation. It also gives emphasis on real time image processing.

The library has more than 2500 optimized algorithms, which incorporates a far-reaching set of both exemplary and best in class PC vision and AI calculations. These calculations can be utilized to distinguish and perceive faces, recognize objects, characterize human activities in recordings, track camera developments, track moving items, extricate 3D models of articles, produce 3D point mists from sound system cameras, fasten pictures together to create a high goals picture of a whole scene, find comparative pictures from a picture database, expel red eyes from pictures taken utilizing streak, follow eye developments, perceive the landscape and build up markers to overlay it with enlarged reality, and so forth. The library is utilized broadly in organizations, inquire about gatherings and by administrative bodies. [2]

The main objectives of this thesis are to provide the web alternative to the available image filter application, to make it platform independent and provide the set of the algorithms with at least 95% succesfull recognition rate.

2. Project structure

2.1 Folders structure

This section will give an overview about the hierarchy of the project.

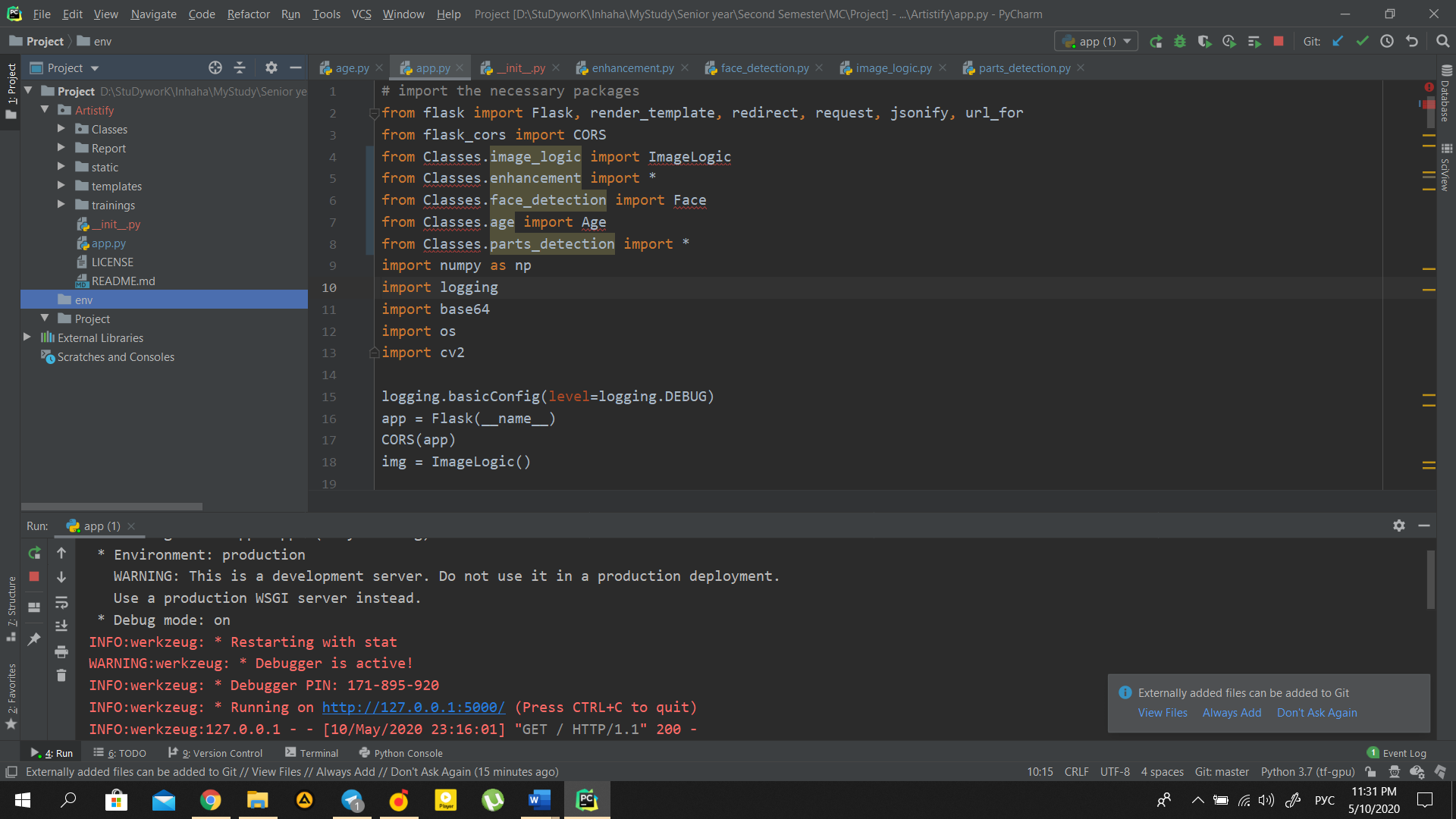
*Main directory:*

Figure 1: Main directory

Description of the project structure will begin from the Main directory which is calles Artistify and contains in itself Classes, Report, static, templates and trainings.

*Classes directory:*

In classes directory the *age.py, enhancement.py, face\_detection.py, image\_logic, and parts\_detection.py* can be found

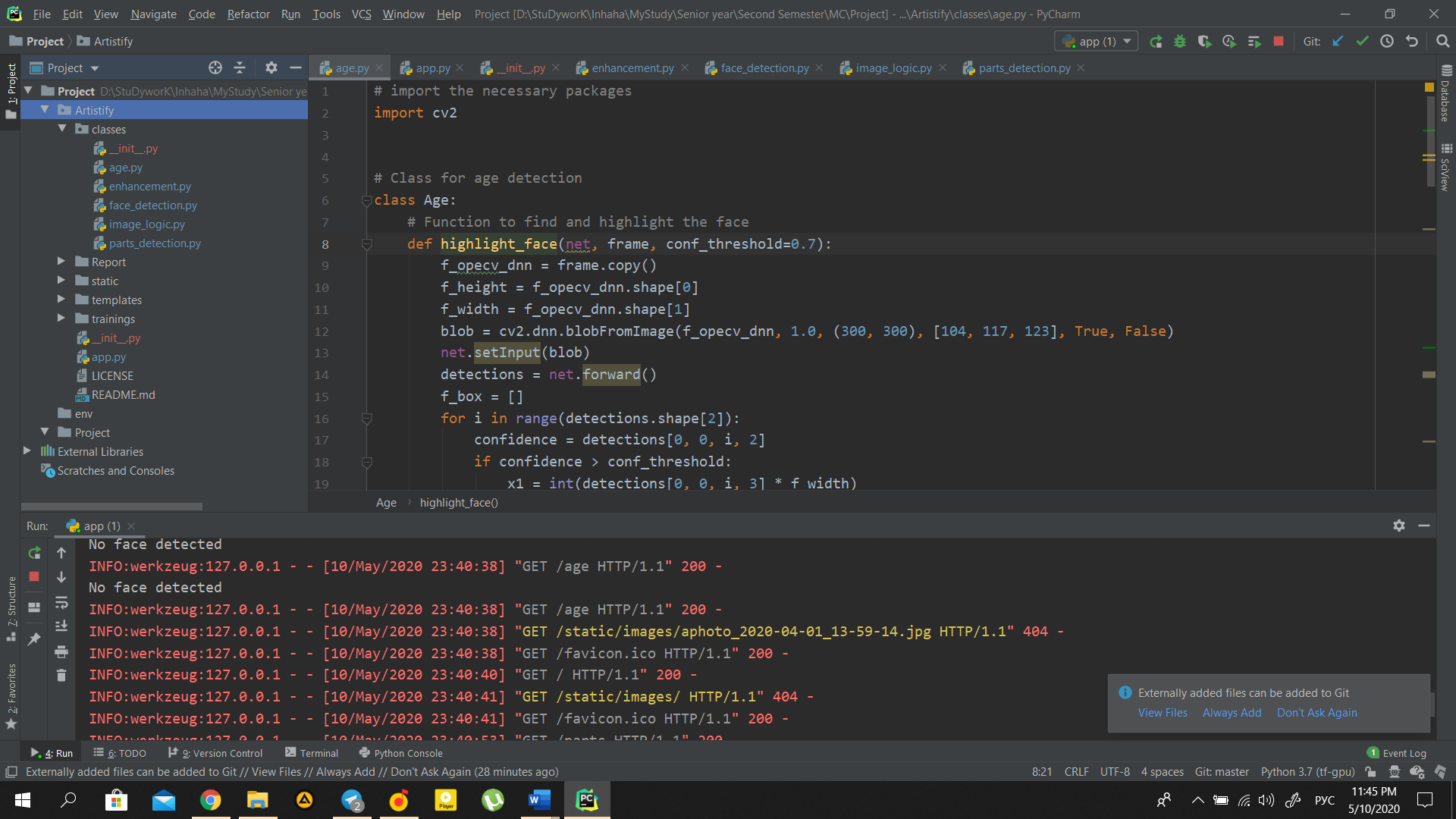
*Age.py* file contains logic for age detection part of the project.

Figure 2: Classes directory

*Enhancement.py* file contains logic for all image processing techniques which was used in this project.

*Face\_detection.py* file contains logic for number of face detection part of the project.

*Image\_logic.py* file contains logic for taking and applying techniques on image.

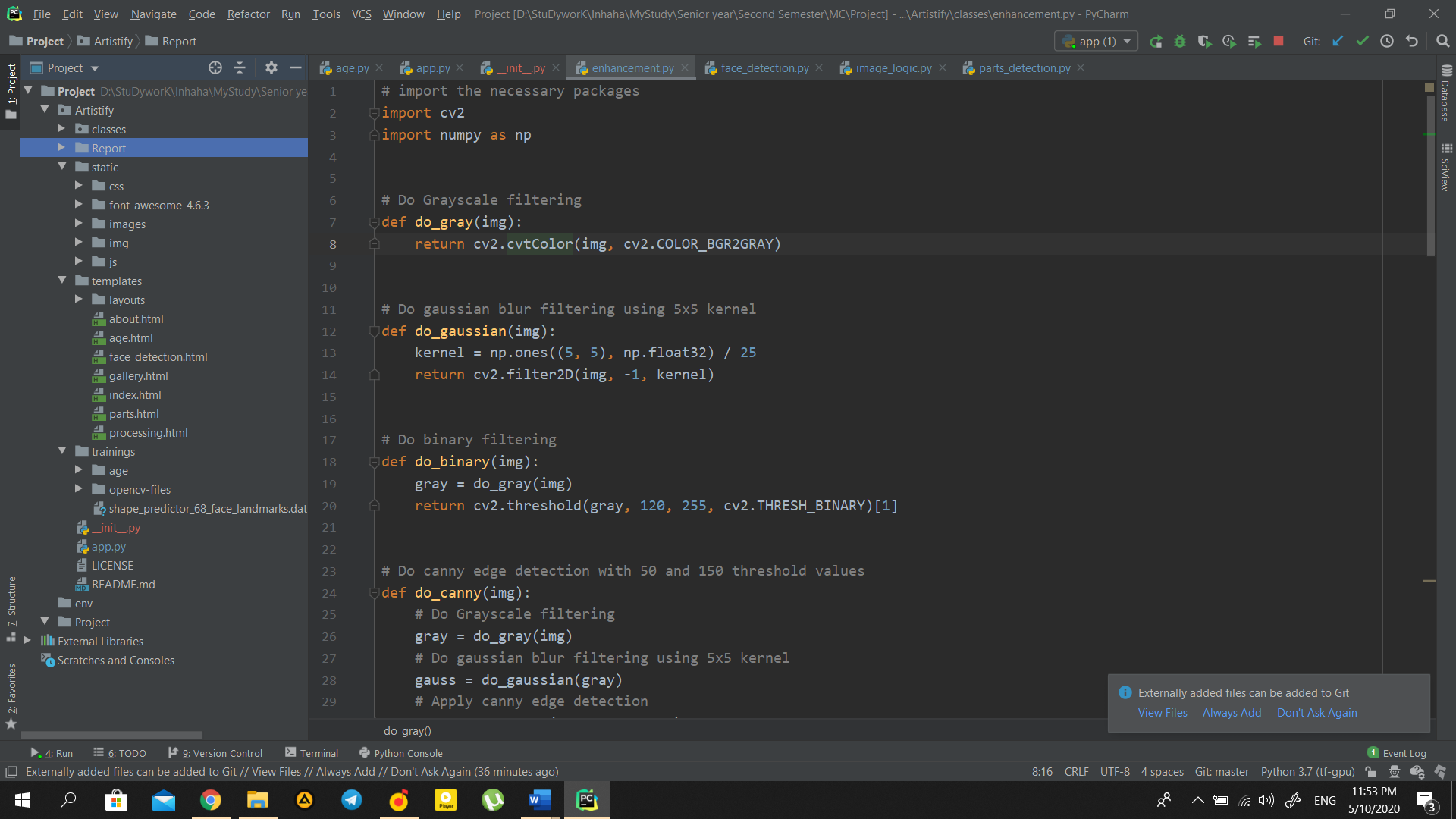
*Parts\_detection.py* file contains logic for parts of the face detection part of the project.

Figure 3: Static, Templates and Trainings folder

*Static, templates and trainings directories*:

Figure 4: Main page

In static, templates and trainings folders all the necessary dependencties for the project can be found. Last file is the app.py which is the main python file to run whole project.

2.2 Libraries

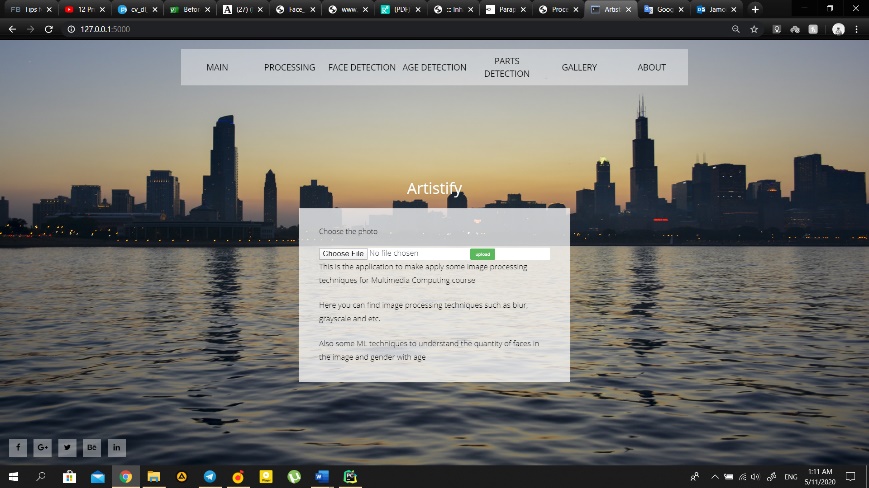
Flask is a lightweight [WSGI](https://wsgi.readthedocs.io/) web application framework. It is intended to make getting started quick and easy, with the capacity to scale up to complex applications. It started as a simple wrapper around [Werkzeug](https://palletsprojects.com/p/werkzeug) and [Jinja](https://palletsprojects.com/p/jinja) and has gotten one of the well known Python web application frameworks.

Flask offers suggestions, yet doesn't uphold any dependencies or project layout. It is up to the developer to pick the tools and libraries they want to use. There are numerous extensions given by the community that makes adding new functionality easy. [3]

3. Proposed solution

3.1 Main page

At the point when image quality is mulled over, there is a plenty of factors that impact the system’s accuracy. It is critical to apply different image pre-processing techniques to standardize the images that you supply to a face recognition system. Most face recognition algorithms are extremely touchy to lighting conditions, so that if it was trained to recognize a person when they are in a dark room, it probably won’t recognize them in a bright room, etc. This problem is known as "lumination dependent", This is why it is so important to use a good image pre-processing filters before applying face recognition.

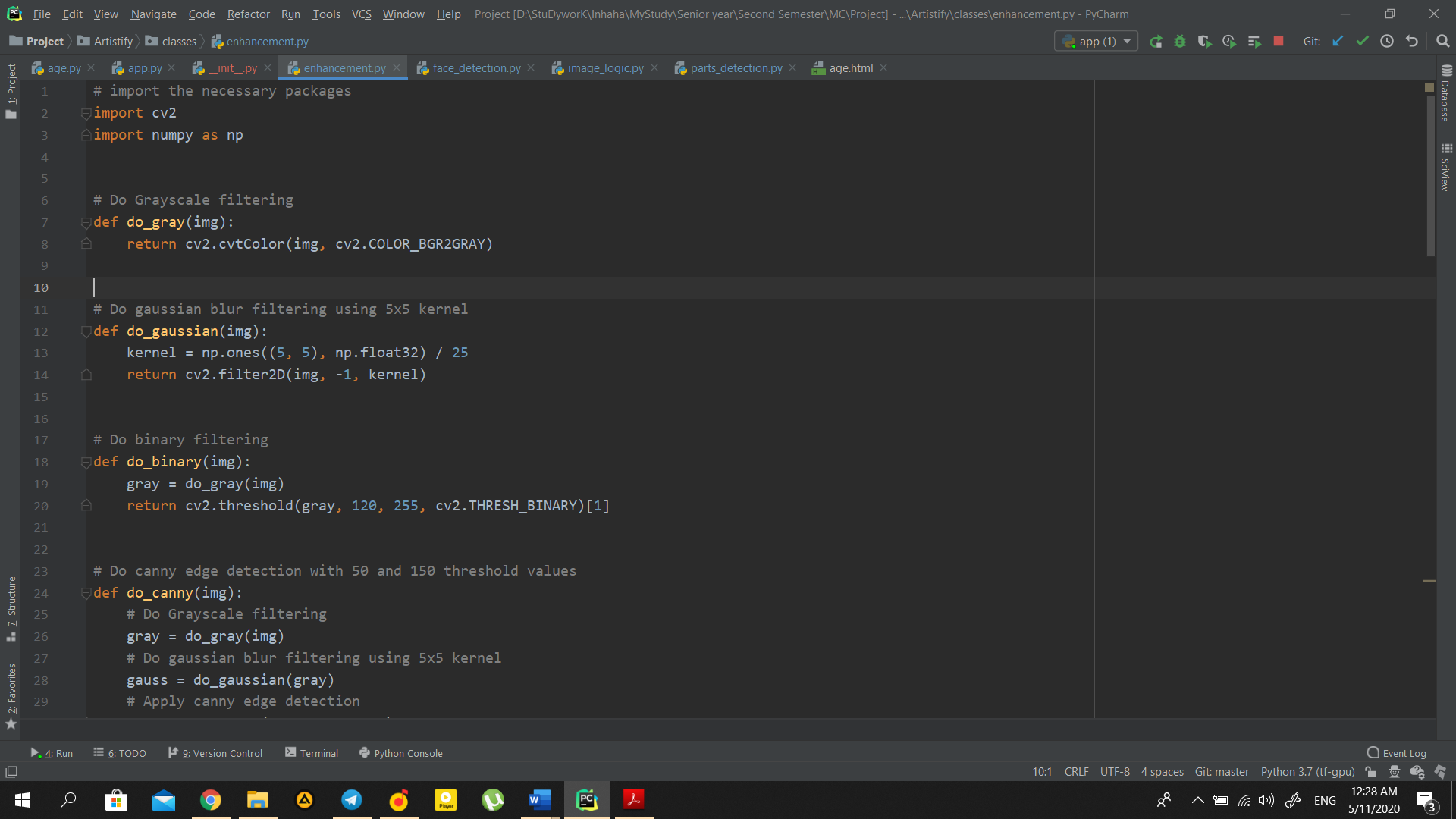
With the used Flask framework and basic HTML,CSS bundle the web-application was created

3.2 Image processing techniques

3.2.1 Grayscale filtering

In order to convert image to grayscale we first need to import necessary libraries, which will make available the functionalities needed to read the original image and to convert it to gray scale.

To have easier way to navigate through code, functions were divided into several classes, which were discussed *Project Structure* part of the document. Therefore, we are calling *do\_gray*function from *Enhancement*class [4]

 To do conversion we need to call the *cvtColor* function, which gives the opportunity to convert the frame from a color space to another.

As first input, this function receives the current frame. As second input, it takes the color space conversion code. Since we need to convert our frame from the BGR to gray color space we need to use the **COLOR\_BGR2GRAY** code.

Here is the result of the code compilation:

Figure 5: Grayscale Filter

3.2.3 Blurring filter

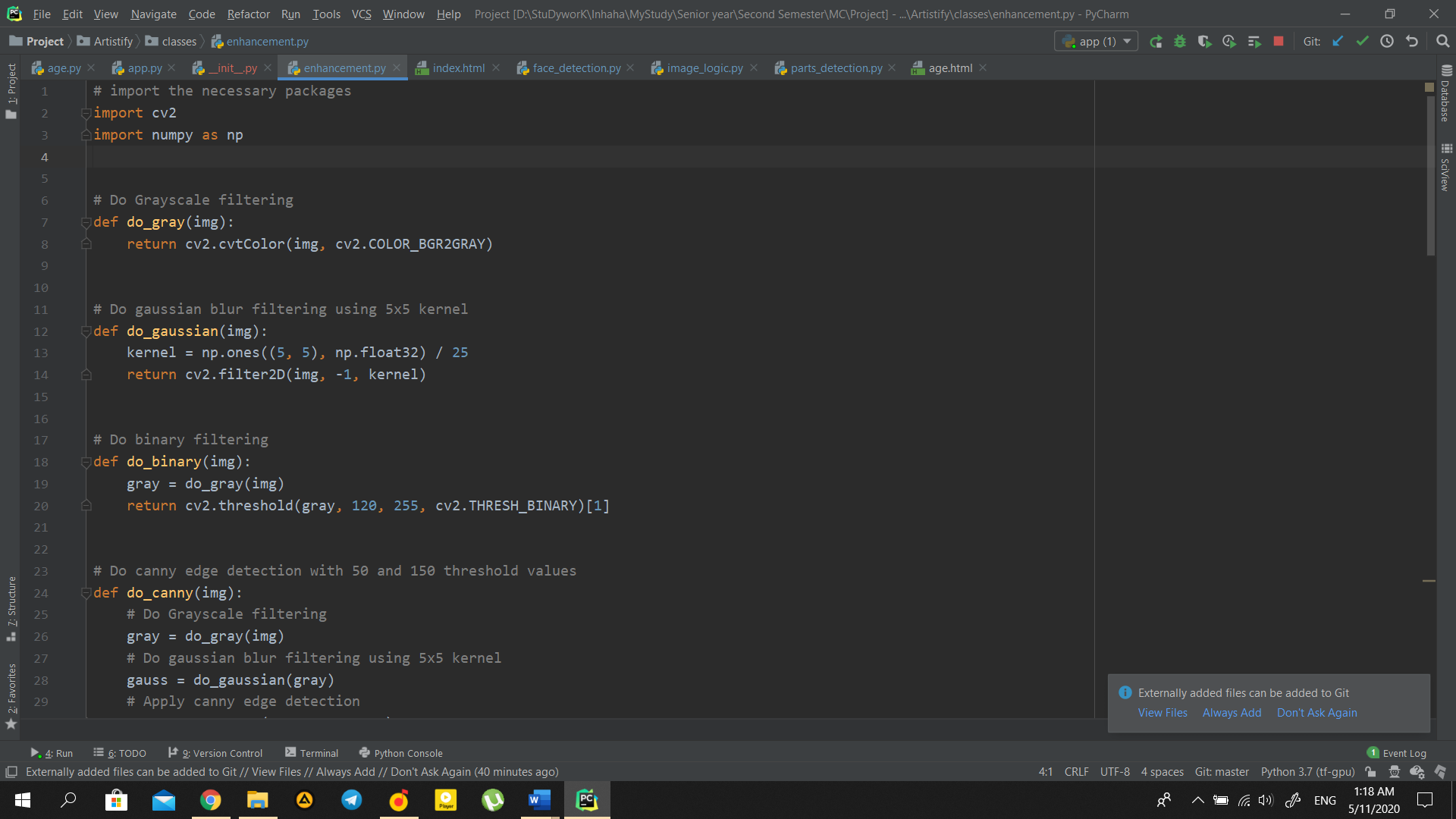
Each of the pixels for a grayscale image is described by a single number that describes the brightness of the pixel. In order to smoothen an image, the typical answer would be to modify the value of a pixel with the average value of the pixel intensities around it. Averaging out the pixels to reduce the noise will be done by a kernel. This kernel of normally distributed numbers(np.array([[1,2,3],[4,5,6],[7,8,9]]) is run across our entire image and sets each pixel value equal to the weighted average of its neighboring pixels, thus smoothening our image. In our case we will apply a 5x5 **Gaussian kernel**.

Figure 6: Sepia filter

Image blurring is achieved by convolving the image with a low-pass filter kernel. It is useful for removing noise. It actually removes high frequency content (eg: noise, edges) from the image. So, edges are blurred a little bit in this operation (there are also blurring techniques which don't blur the edges)

We should specify the width and height of the kernel which should be positive and odd. We also should specify the standard deviation in the X and Y directions, sigmaX and sigmaY respectively. If only sigmaX is specified, sigmaY is taken as the same as sigmaX. If both are given as zeros, they are calculated from the kernel size. Gaussian blurring is highly effective in removing Gaussian noise from an image.

Here is the result of the blurred image:



Figure 7: Blurring filter

By playing with kernel we can achieve many other filters, such as Embossing, Sharpening or Sepia.

3.3 Face detection

OpenCV gives us opportunity to do even more complex tasks relatively easy. There are example to detect quantity of faces.

def detect\_faces(img):  
 # List of the faces  
 faces\_list = []  
 # Convert the image to gray scale  
 gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
 # Load face detector  
 face\_cascade = cv2.CascadeClassifier('trainings/opencv-files/lbpcascade\_frontalface.xml')  
 # Detect multiscale images  
 faces = face\_cascade.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=5);  
  
 # If no face detected, return empty list  
 if len(faces) == 0:  
 return faces\_list  
  
 for i in range(0, len(faces)):  
 (x, y, w, h) = faces[i]  
 face\_dict = {'face': gray[y:y + w, x:x + h], 'rect': faces[i]}  
 faces\_list.append(face\_dict)  
 # Return the face image area and the face rectangle  
 return faces\_list

As it can be seen in Fig. 7 the face was succesfully detected

Here is the example of the execution:

Adding more features can make the detection more reliable, examples for that is the fact the region of the eyes is darker than the nose and etc.

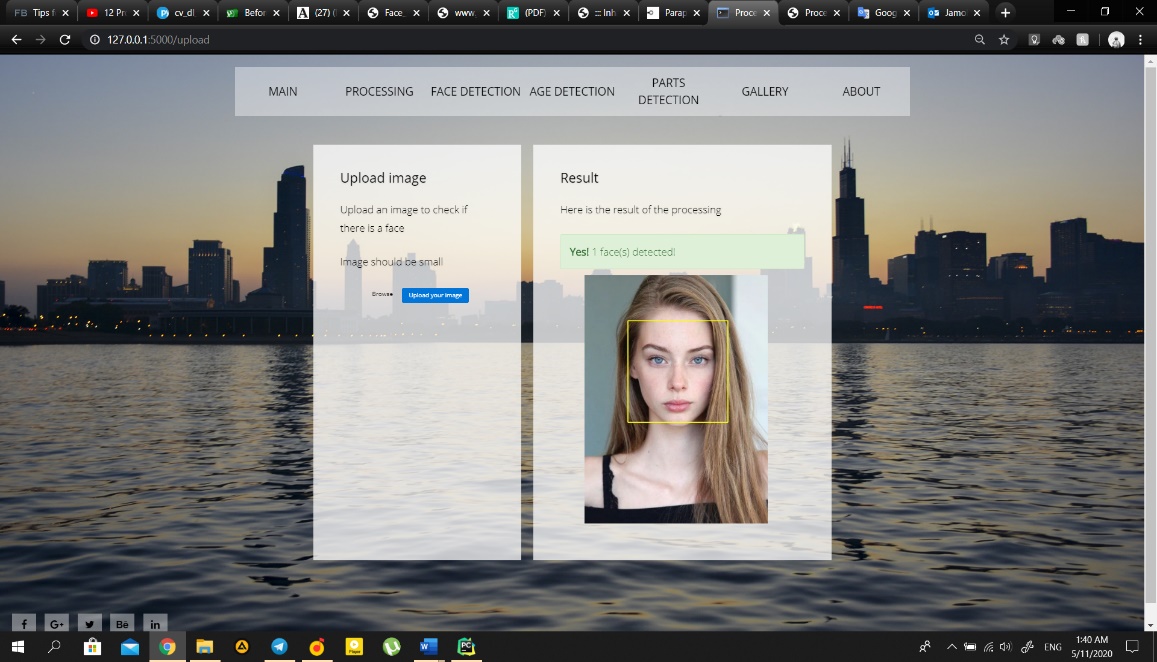


Figure 8: Face detection

3.4 Gender and Age detector

Age and gender, two of the key facial attributes, play a very fundamental role in social interactions, making age and gender estimation from a single face image an important task in intelligent applications, such as access control, human-computer interaction, law enforcement, marketing intelligence  
and visual surveillance, etc.

To understand, we will do the two stage process:

1. Detect faces in the input image
2. Extract ROI and apply age detector algorithm to predict the age

The problem here is that the person of mid 60’s who never smoked and used sunscreen can seem the same with a person in the late 30’s who smokes every day and never done any exercisies or used the sun screen.

So, unless you know any prior information about the person guessing the name can become the huge skill test.

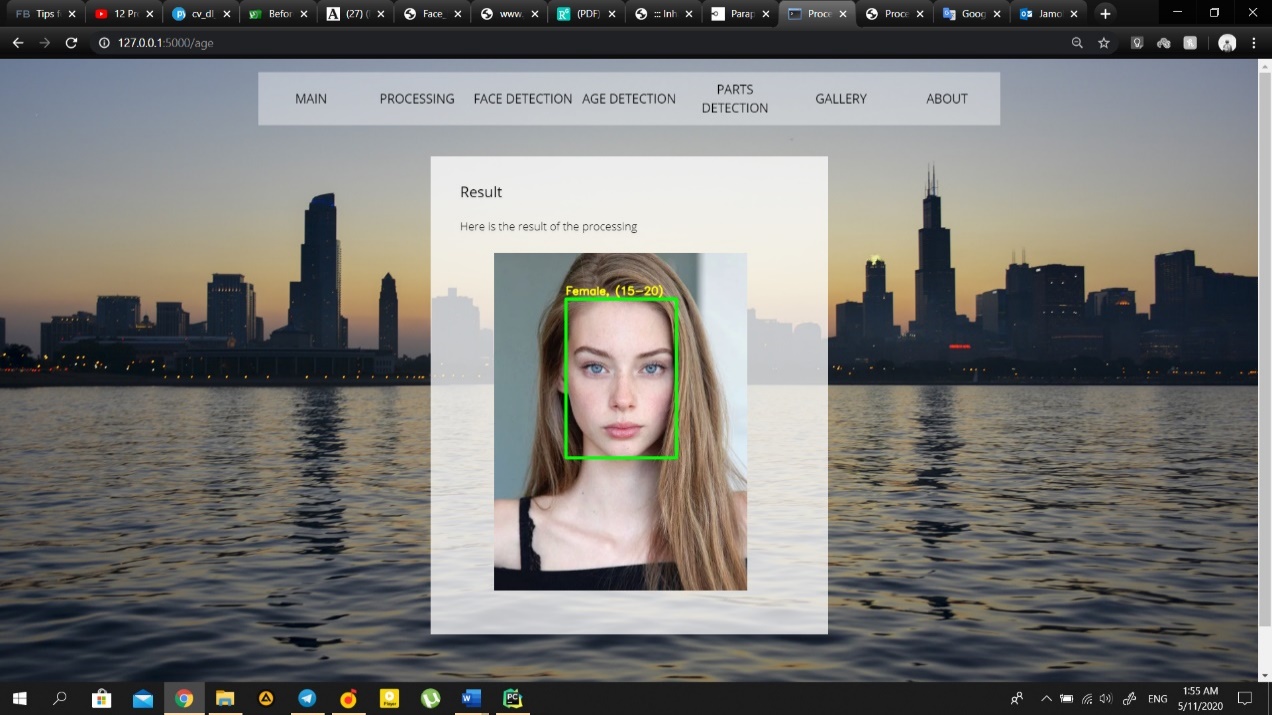


Figure 9: Age and Gender detection

For example: Take a girl from Figure 10, a person could guess, that this is a young girl about 20 years old. However, to say the exact age could be really difficult, so if a person struggles to detect the exact age, the machine will struggle too. However, the machine can predict the age brackets as a human does. So, in Figure 10, it can bee seen that the algorithm sees the girl as a female of 15-20 years bracket

Still improving the accuracy of the age detection can be harder, than it looks like, without knowing the person's background, to get better result other open models can be used, or the greater training set can be used, to understand various people’s skin differences

3.5 Face parts detection

The proposed solution uses the Dlib software which is a general purpose cross-platform software library written in the C++ language. Its design is heavily influenced by ideas from design by contract and component-based software engineering. Thus it is, first and foremost, a set of independent software components  containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems. [5] With the radiant increase in video and image databases, there is an inconceivable need for automatic comprehension and assessment of data by the intelligent systems as manually it is getting to be really hard.

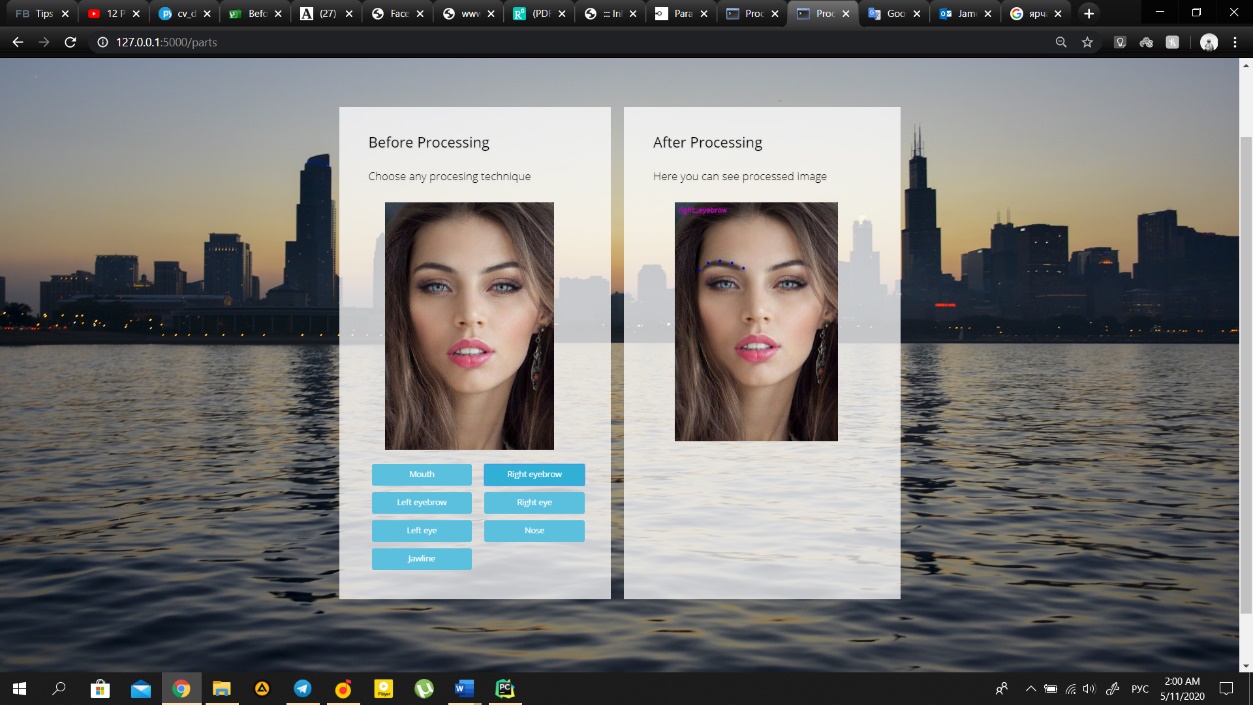


Figure 10: Right Eyebrows detection

Face plays a significant role in social intercourse for conveying identity and feelings of a person.

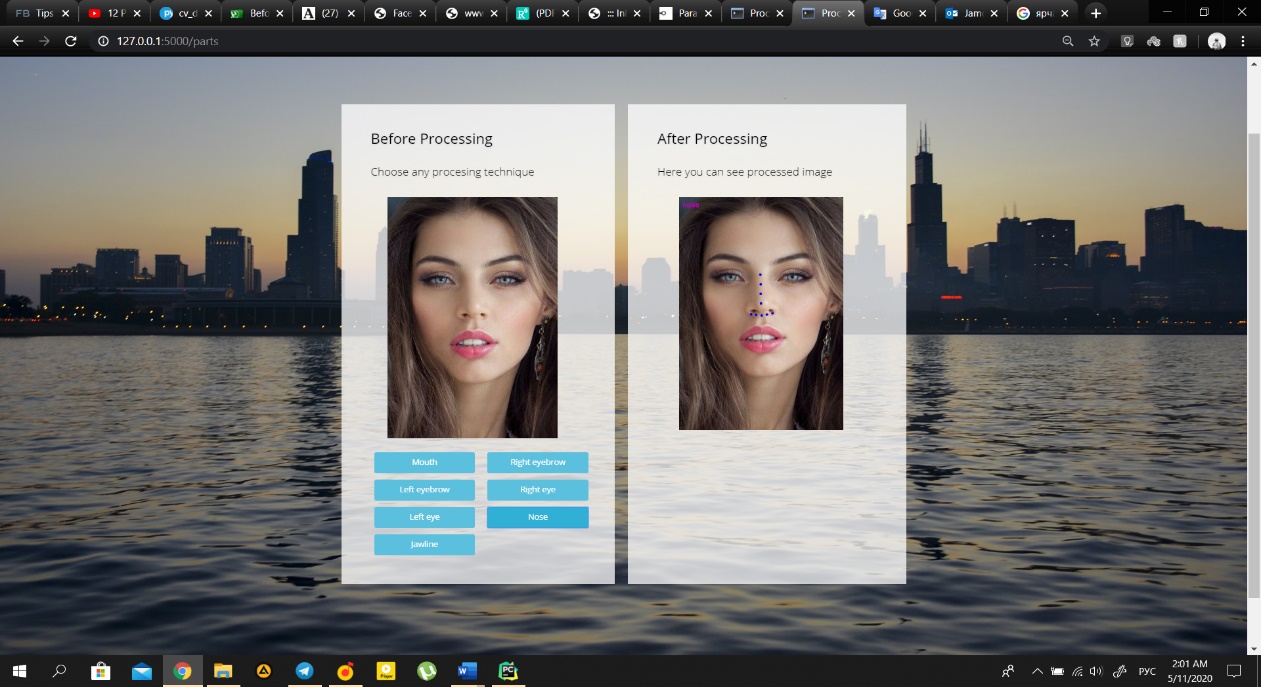
The facial landmark detector is a part of the dlib library, which produces 68 (x,y coordinates). If we examine the image we can understand, that we can index all facial regions of the model, after accessing all points we can draw the results on image. 

Figure 11: Nose detection

In Figure 8 and 9 it can be seen the result of the Dlib detector and predictor. The web application provides the use of the each part of the human face.

4 Conclusion

The paper is divided into 3 parts. The first one give a short overview of the project structure, and information about libraries used. The second part uses OpenCV library and applies several image processing techniques, there were exactly 11 filters, even though not all of them was shown in the paper, the GitHub repository was provided in the beginning of the paper, so it could be checked later. The third part applies face detection, age, and gender recognition and faces parts recognition algorithms. These areas of hard research, because of enabling better interaction between humans and computers.

Moreover, python turned out as a very reliable programming language and the libraries which were used here, such as face detection or Dlib facial landmarks predictors, give the opportunity to create wider projects without knowing the background of the algorithms.

Therefore, there are a huge amount of things to improve performance, for example adding edge detection, color processing and etc. and it is important to have a lot of variations of each person to have more accurate values. So, it is very important to make a lot of experimentations to improve the result.

In future work, I would have an interest in creating a pure new type of filter to create an anime character face from the real face. For that the I will need to use the GAN and train from the huge amount of the characters. Creation an absolutely new image from the real world is an incredibly hot topic, and using the results of it can help anime directors, to hire not only artists but also the actors.

# References

|  |  |
| --- | --- |
| [1] | D. Kuhlman, A Python Book: Beginning Python, Advanced, December 15, 2013. |
| [2] | W. G. I. Intel Corporation, "OpenCV," Intel Corporation, Willow Garage, Itseez, June 2000. [Online]. Available: https://opencv.org/. |
| [3] | A. Ronacher, "Flask | The Pallets Projects," [Online]. Available: https://palletsprojects.com/p/flask/. |
| [4] | "http://docs.opencv.org/modules/imgproc/doc/imgproc.html," [Online]. |
| [5] | D. E. King, "Dlib," 2002. [Online]. Available: http://dlib.net/. |
| [6] | A. Kaehler, and G. Bradski, Learning OpenCV: computer vision in C++ with the OpenCV library, 2nd Ed.,” O’Reilly, 2016. |
| [7] | F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 815-823, 2015. |
| [8] | S. Nagar, Introduction to Python: For Scientists and Engineers, Bookmuft, 2016. |
| [9] | Rafael C. Gonzalez, Richard E. Woods, Digital Image Processing, Bookmuft, 2018. |