

CARTOONIFIER- WEB APP

PART 2

In this part, we will be building a simple web app (which client-server computer program that the client runs in a web browser). Will be making use of Django, which is a simple web framework for making such client server architecture. In this part, you will not be required to demonstrate much knowledge about web development. However, we will be guiding you step by step to exa

TASK 1 (Setting up the environment)

1. After extracting the files. Open up command prompt or terminal and cd into the extracted folder(Cartoonifier-Web-App).
2. Now you have to activate the virtual env, enter the following commands

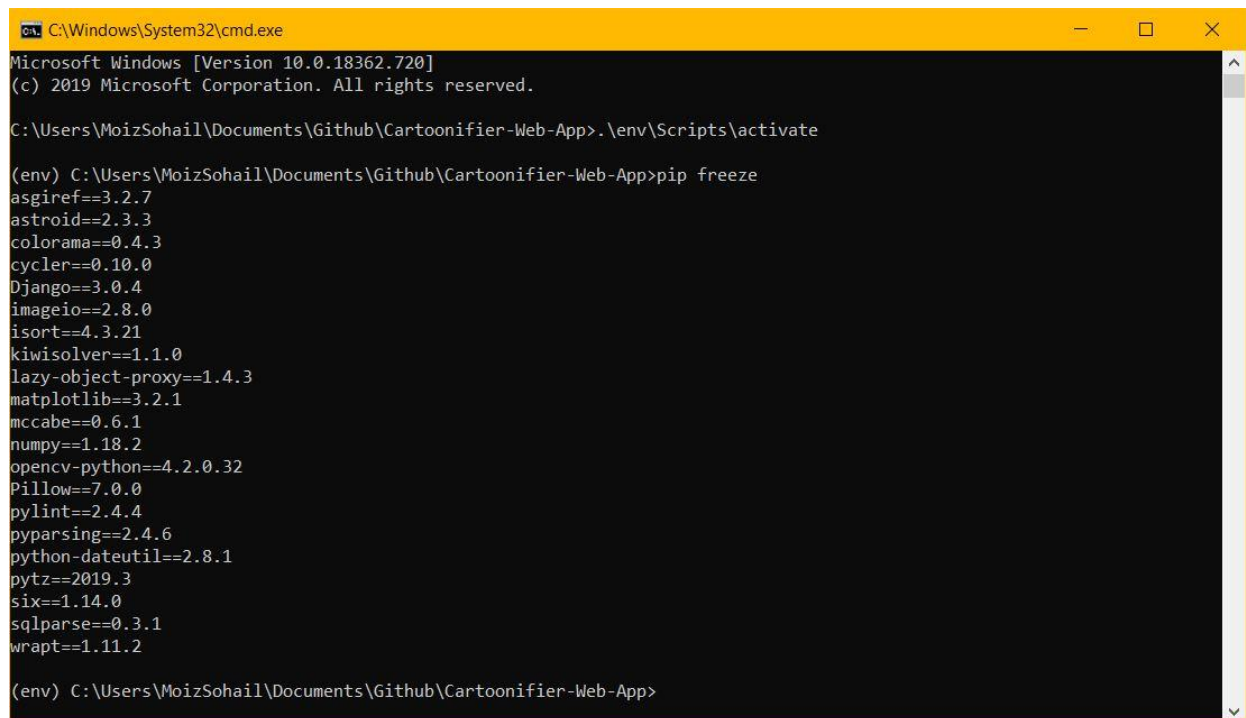
`source env/bin/activate` (for mac and linux users)

`.\env\Scripts\activate` (for win users)

To check if the environment is active. Type `pip freeze`. You will not get anything on the screen.

3. `pip install -r req.txt`

Now, if you type `pip freeze` you will get



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18362.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\MoizSohail\Documents\Github\Cartoonifier-Web-App>.\env\Scripts\activate

(env) C:\Users\MoizSohail\Documents\Github\Cartoonifier-Web-App>pip freeze
asgiref==3.2.7
astroid==2.3.3
colorama==0.4.3
cyclert==0.10.0
Django==3.0.4
imageio==2.8.0
isort==4.3.21
kiwisolver==1.1.0
lazy-object-proxy==1.4.3
matplotlib==3.2.1
mccabe==0.6.1
numpy==1.18.2
opencv-python==4.2.0.32
Pillow==7.0.0
pylint==2.4.4
pyparsing==2.4.6
python-dateutil==2.8.1
pytz==2019.3
six==1.14.0
sqlparse==0.3.1
wrapt==1.11.2

(env) C:\Users\MoizSohail\Documents\Github\Cartoonifier-Web-App>
```

4. Great! Now you have the all required dependencies up and running. cd into backend.

5. Enter ipconfig (iwconfig for Linux) or ipconfig getifaddr en0 and locate your ip address. You will need this at a later stage. If you are connected to a WIFI it must be near wireless LAN adapter. (Mine was written near 'Wireless LAN adapter Wi-Fi' next to IPv4). Copy that or write it down somewhere.
6. Then type python manage.py runserver 0.0.0.0:8181.
7. Open up your browser and enter your ip address and the 8181 port number in this format -> <IP ADDRESS>:8181/cartoon e.g. 127.0.0.1:8181/cartoon
8. Final output:

Yay! Task1 is completed. You must have successfully setup the enviroment.

Now you can proceed to step2

If you are still seeing this message on attempting task2. Then you have not properly implemented it. Make sure you follow the instruction exactly as they are specified.

Restarting the Jupyter

TASK 2 (Integrating your code)

1. After completing the previous part, open up your .ipynb file (this is the completed notebook from part 1) in either jupyter notebook or google colab and goto File>"Download as .py"
2. Now move the downloaded .py into backend\src.
3. Open up the whole file and encase the whole code inside a function named cartoonifier

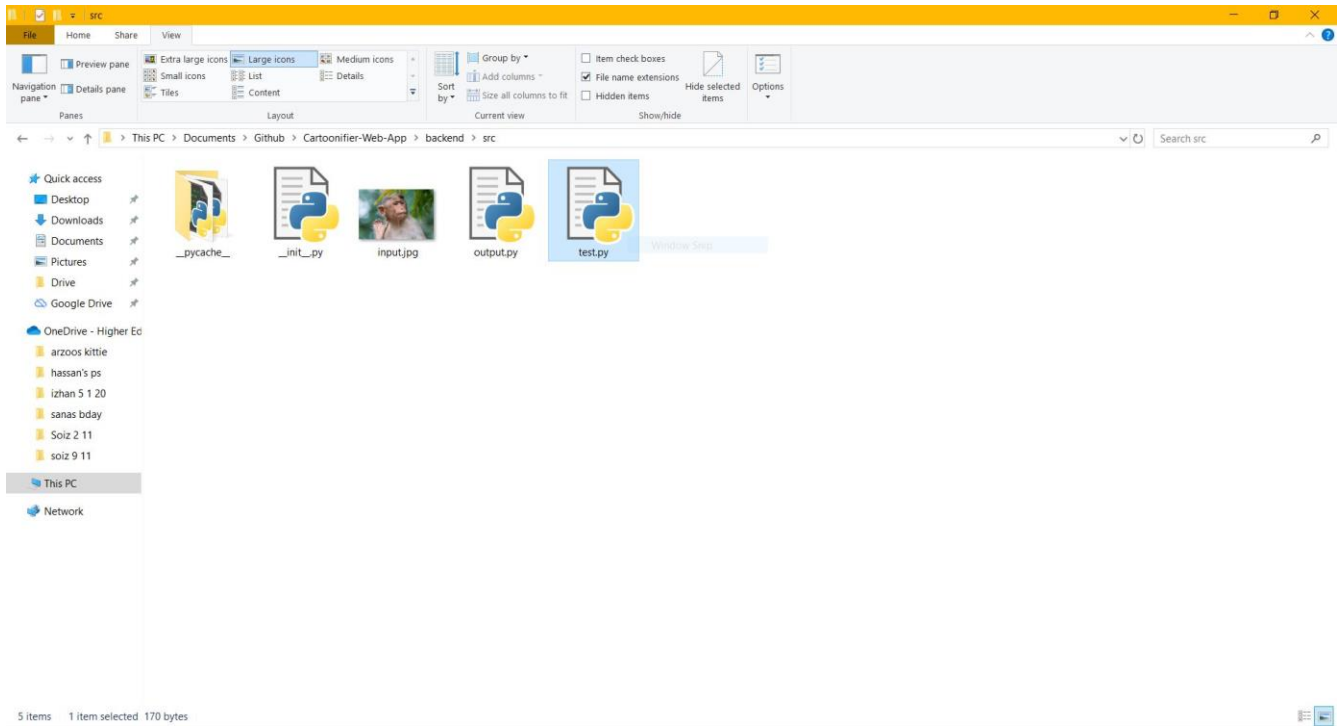
```
def cartoonifier(imagepath):
```

```
    #All the code that was already there in the file.
```

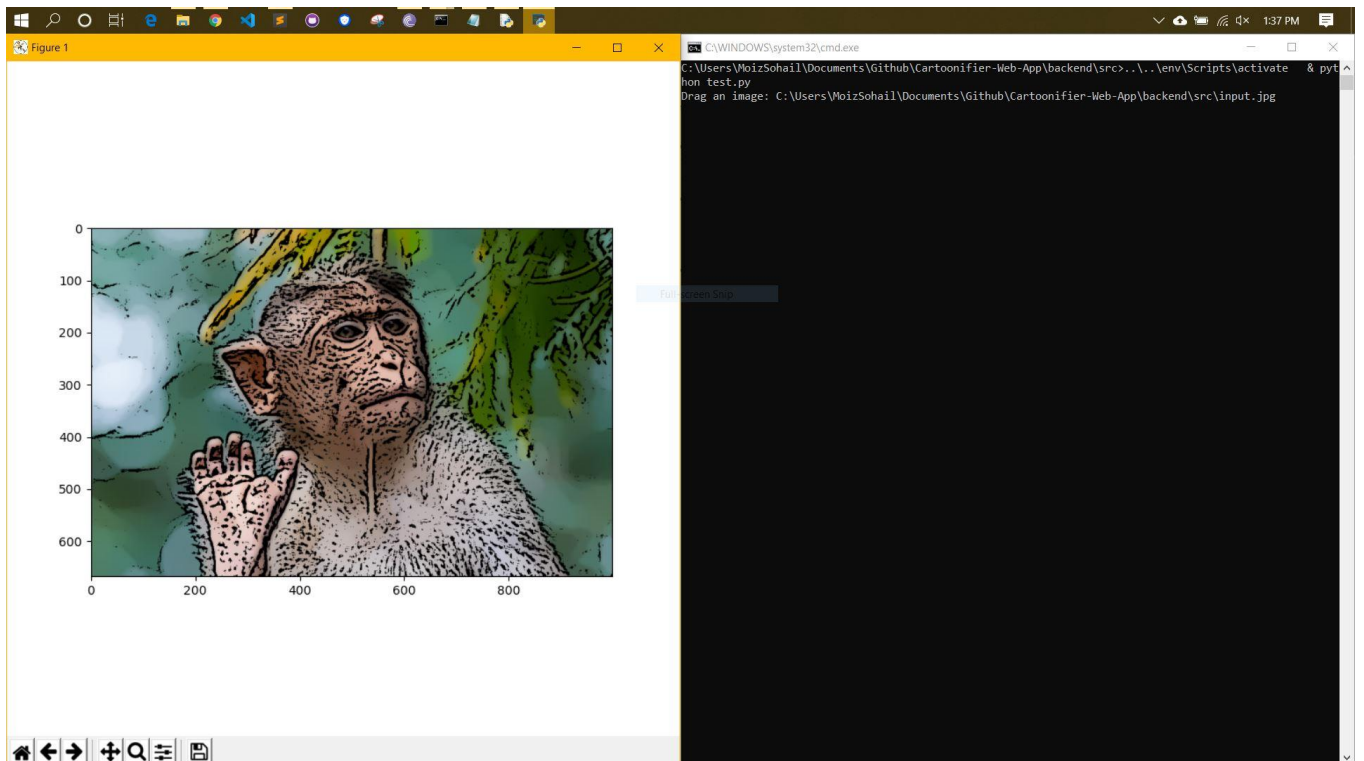
```
    return final
```

4. The function takes an imagepath as an input and returns the cartoonified image as an output. There is a possibility that you might have to edit your code to meet the input and output requirements of the webapp to function properly.
5. After doing that, rename the file as output.py. And run test.bat to check if the file is working correctly.

This is how the directory src will look like



And your output from test.bat will be

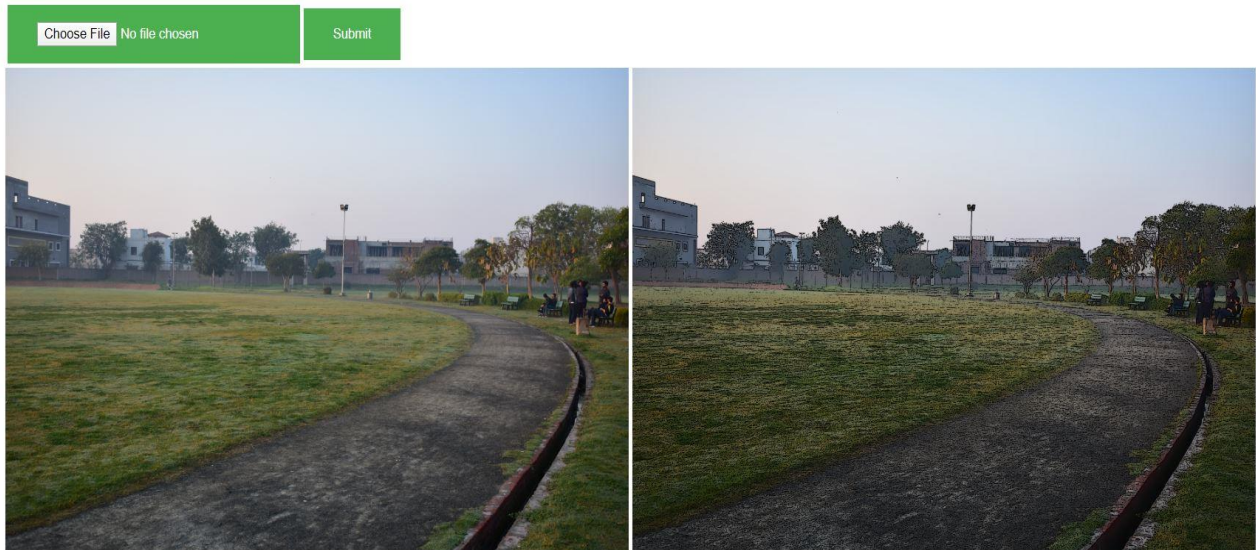


TASK 3 (Testing the app)

1. Finally, you can run `python manage.py runserver 0.0.0.0:8181` if it is not already running.
2. You can type `<IP ADDRESS>:8181/cartoon` in your cellphone browser or your neighbor's browser to test your cartoonifier web app if you want.

Take a screenshot of the final output and submit it along with the output.

Final Output example:



TASK 4 (Setting up a new route)

Now you will make some more changes in the Django code to add more functionality to your web app. We love to let you go creative but since we do not want to through you in the dark, we will guide you through the whole process. In this, we will add another functionality within the cartoonifier app that converts the given image to black and white.

You will only be editing the `cartoonifier/urls.py` and `cartoonifier/views.py`. Every file in this folder play their part in building up your webapp but for the sake of this assignment. We will not be dwelling in them.

Cartoonifier/templates/

This is where all your html is stored. But since most of you are not aware of html and css. We will not go deeper in this.

Cartoonifier/urls.py

In simplest terms, this has the task of routing your requests to a function in views.py which then deals with the request. For example: `path('cartoon/', views.cartoon)` if the browser sends a <http://127.0.0.1:8181/cartoon/> requests the function looks for a function named cartoon in the views.py file and handles it accordingly.

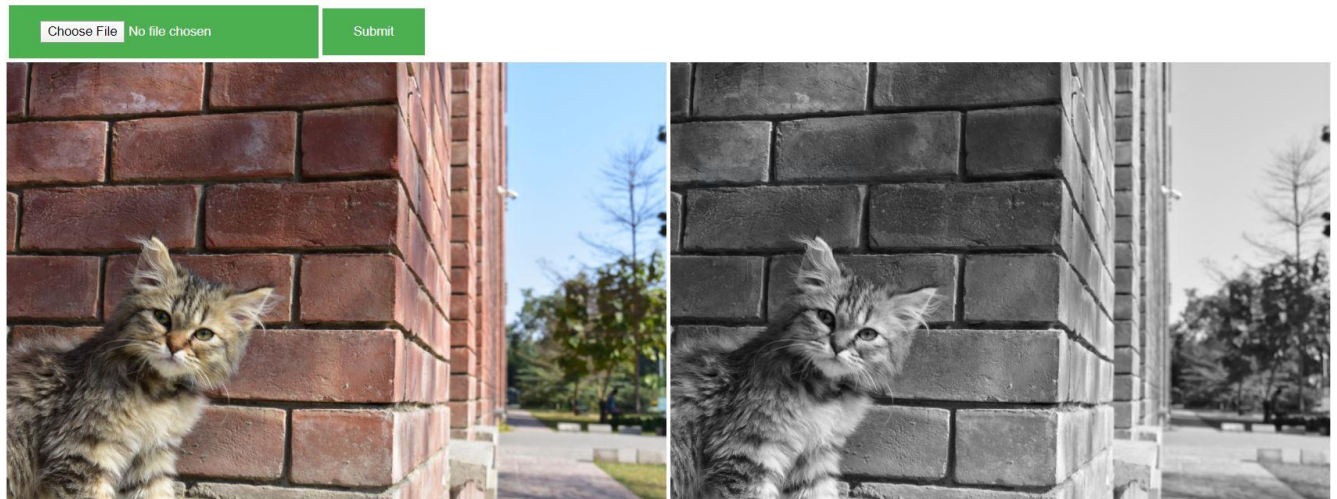
Cartoonifier/views.py

This is where all the functions that deal with the requests are defined. All of these functions are responsible for taking in the input and returning an html response after processing them. For example, in the first task the get function plays the role of taking in the request and extracting an image from it then sending it into the cartoonifier function and then finally the images is sent into the template html to be rendered at the client side.

Please read the comments in the file for further instructions on how to approach this step

After doing this, please take a screenshot of the final output and submit it with the other files.

Final output example:



Isn't she the cutest?

Submission guidelines:

You only have to submit the following files for part2:

1. Backend (whole folder)
2. Final screenshot of outputs of task 3 and 4

Zip them up and submit it in the following format: **<ROLL_NUMBER>_A3.zip**