Image Processor Application

Introduction:

The image processor application allows user to apply transformations to an image uploaded by the user. User can send a combination of transformations that are needed to be done on the image and those transformation would be done in the provided order. The app has been tested with jpg, png, jpeg and bmp formats and is able to perform the following transformations on the image:

1. Flip Horizontal
2. Flip Vertical
3. Rotate Left
4. Rotate Right
5. Greyscale
6. Thumbnail
7. Resize
8. Rotate to different angles.

Architecture Style/Pattern:

The application follows a Client-Server style. It is basically a two-tier architecture. The application is written in python, using Pillow library for image manipulation and exposing the functions with the Flask framework.

Below are the characteristic design patterns I used for this web application.

Web service API style:

- Resource API
- The interactions to the API are made through the POST and GET requests.

Client/Server Interaction:

- Request/Response
- I chose to implement the Request/Response pattern for quick and simple method of handling the upload and display of image.

Component Design Documentation:

User Interface (UI):

The front-end interface consists of three different pages home, processing and error. All the pages extend the layout page. The UI relies on Bootstrap, CSS and some JavaScript. Checking on the transformations checkboxes triggers the JavaScript to store the transformation in the order they are clicked by the user and store them in an array along with the number of times those transformations are needed to be performed (1 being the default value). When the user clicks on the 'Submit' button an AJAX request is triggered that wraps on this data into API compatible JSON format and POST it to the /api/createList endpoint for processing. An alert message is printed to show the user the order of transformation they have entered.

API:

The API is written in Python, using the Pillow library for image manipulation and exposing the function with the Flask framework.

The API offers four endpoints. The home endpoint is used to connect to the home page of our application. The /upload endpoint guides to uploading the image the user wants to perform the transformations on. The /createList parse the POSTed JSON data and perform the required transformations. The /images/<filename> would get the transformed image back from the images folder.

User can upload the image from his local machine and the API would save that image into the images folder, or otherwise specify the relative path, from that folder, in 'filename' parameter. If the request is correct, the transformations would be performed on the image and the modified image will be saved to the images folder and displayed to the user. In case that the file with the format other than png/jpg/jpeg/bmp is uploaded by the user an error message would be displayed.

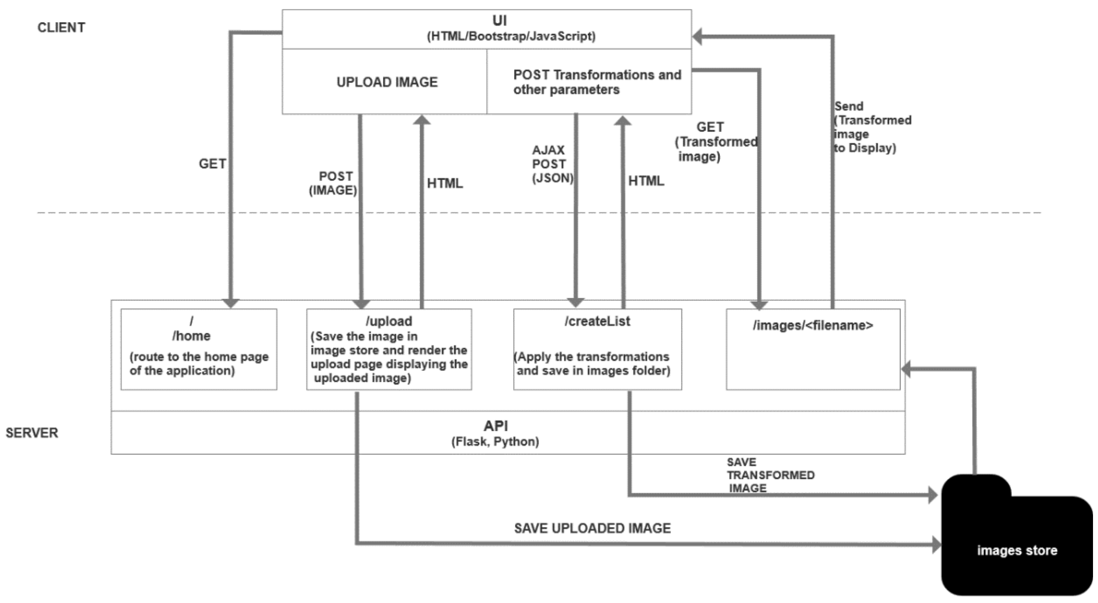
Database:

No database has been used for now. I am just using an images folder to store the uploaded and transformed image for now.

<u>Specifications satisfied by the application:</u>

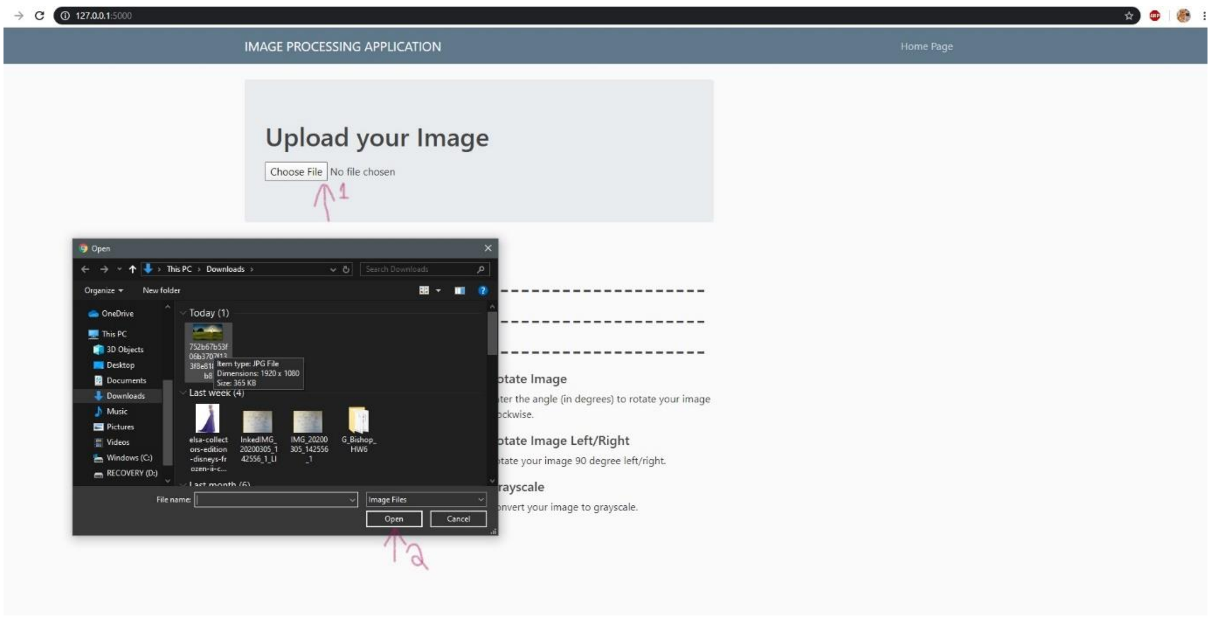
- User can send the Combination of Operations that he needed to perform.
- The transformation would be performed in order they are selected by the user (more specifically in the order the checkboxes are checked by the user).
- User can perform each operation multiple times.

Architecture Diagram


CLIENT
UI
(HTML/Bootstrap/JavaScript)
UPLOAD IMAGE
POST Transformations and other parameters
GET
POST (IMAGE)
HTML
AJAX POST (JSON)
HTML
GET (Transformed image)
Send (Transformed image to Display)
/ /home (route to the home page of the application)
/upload (Save the image in image store and render the upload page displaying the uploaded image)
/createList (Apply the transformations and save in images folder)
/images/<filename>
SERVER
API
(Flask, Python)
SAVE TRANSFORMED IMAGE
SAVE UPLOADED IMAGE
images store

Working of the Web Application:

- To test the app locally, run “imageProcessorAPI.py” and navigate to “localhost:5000”.
- Use the “Choose File” button to upload the desired image file.



Upload your Image

Choose File No file chosen

Item type: JPG File
Dimensions: 1920 x 1080
Size: 365 KB

otate Image
ter the angle (in degrees) to rotate your image
ckwise.

otate Image Left/Right
tate your image 90 degree left/right.

rayscale
nvert your image to grayscale.

- If successful, the browser would direct to the processing page.

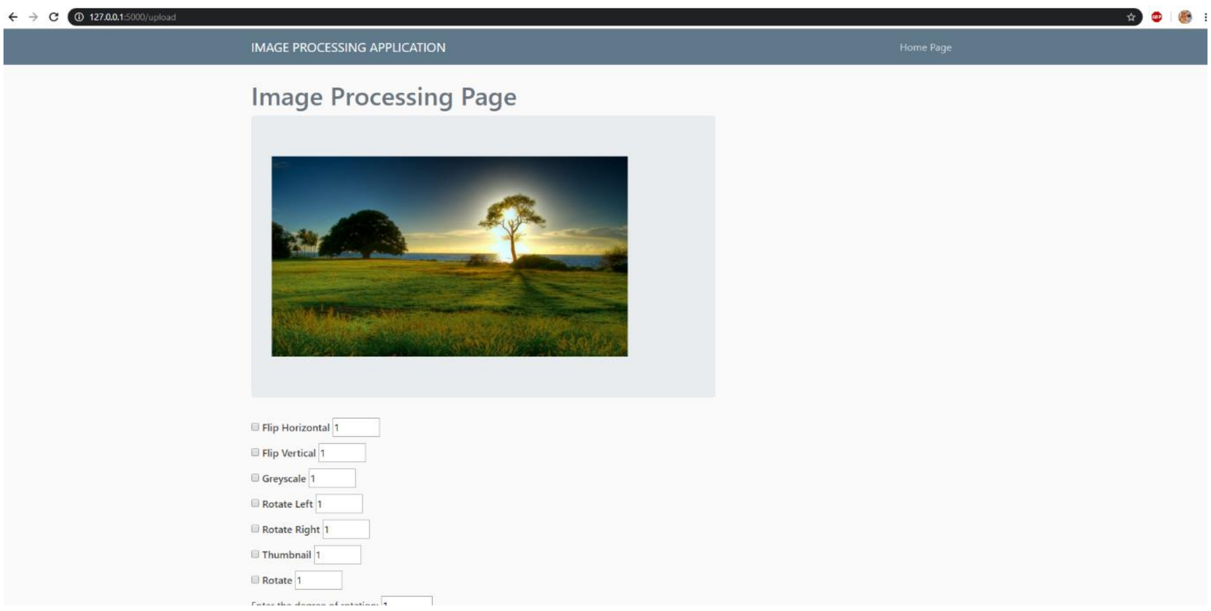


Image Processing Page

☐ Flip Horizontal 1
☐ Flip Vertical 1
☐ Greyscale 1
☐ Rotate Left 1
☐ Rotate Right 1
☐ Thumbnail 1
☐ Rotate 1

- Check on the checkboxes for the desired transformation in the required order. Also input the no. of times each transformation is needed to be performed (if more than one).
- Click on "Submit" button, the parameters would be sent through AJAX with a POST method.

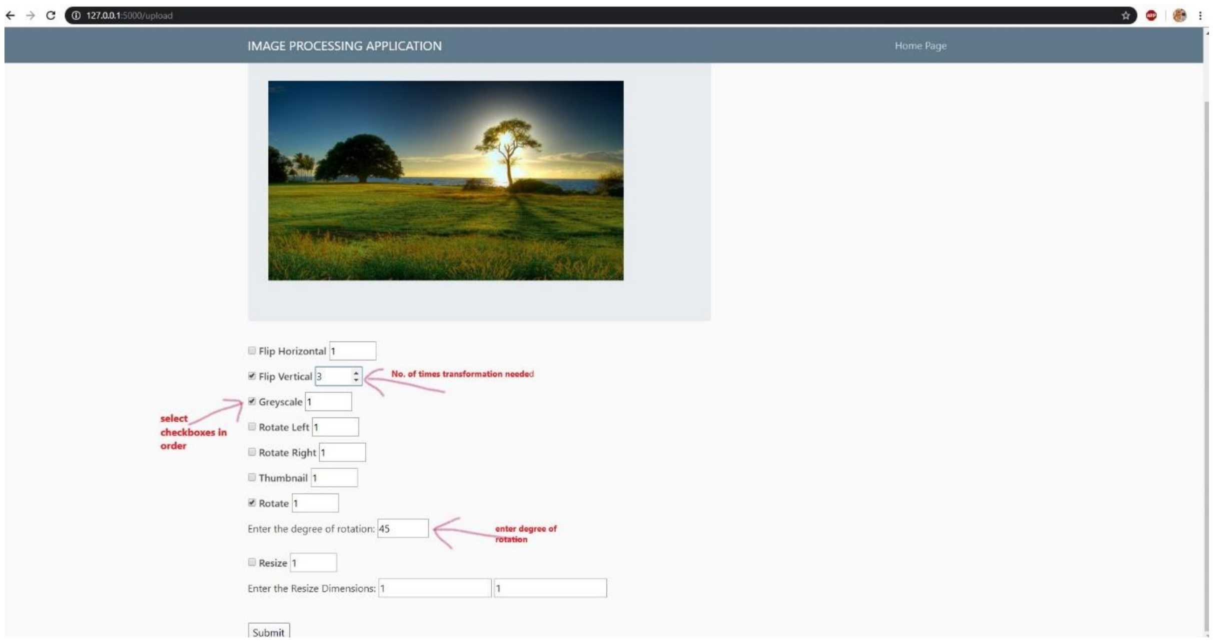

127.0.0.1:5000/upload

IMAGE PROCESSING APPLICATION

Home Page

- [ ] Flip Horizontal 1
- [x] Flip Vertical 3
- [x] Greyscale 1
- [ ] Rotate Left 1
- [ ] Rotate Right 1
- [ ] Thumbnail 1
- [x] Rotate 1

Enter the degree of rotation: 45

- [ ] Resize 1

Enter the Resize Dimensions: 1 1

Submit

No. of times transformation needed
select checkboxes in order
enter degree of rotation

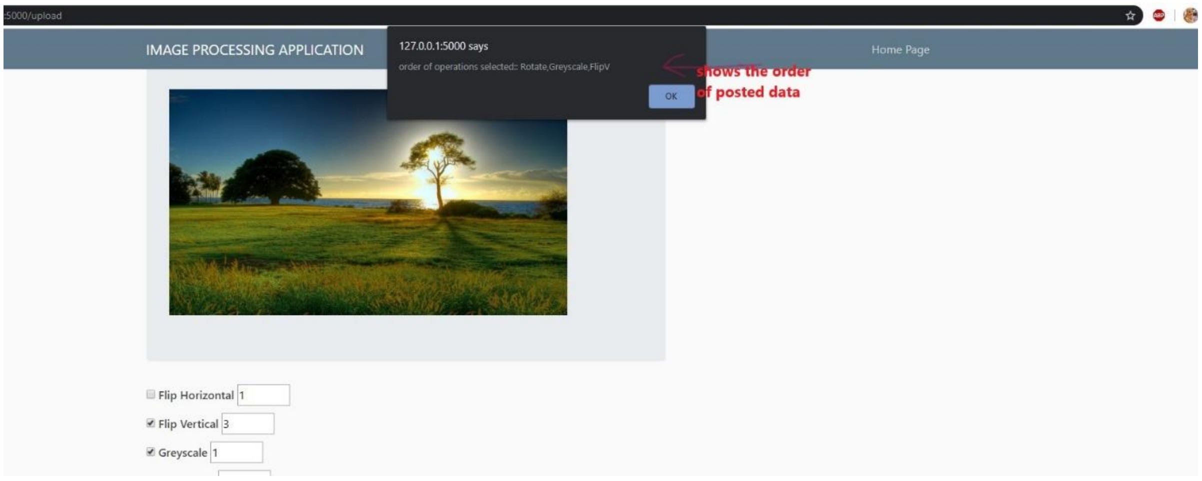

:5000/upload

IMAGE PROCESSING APPLICATION

Home Page

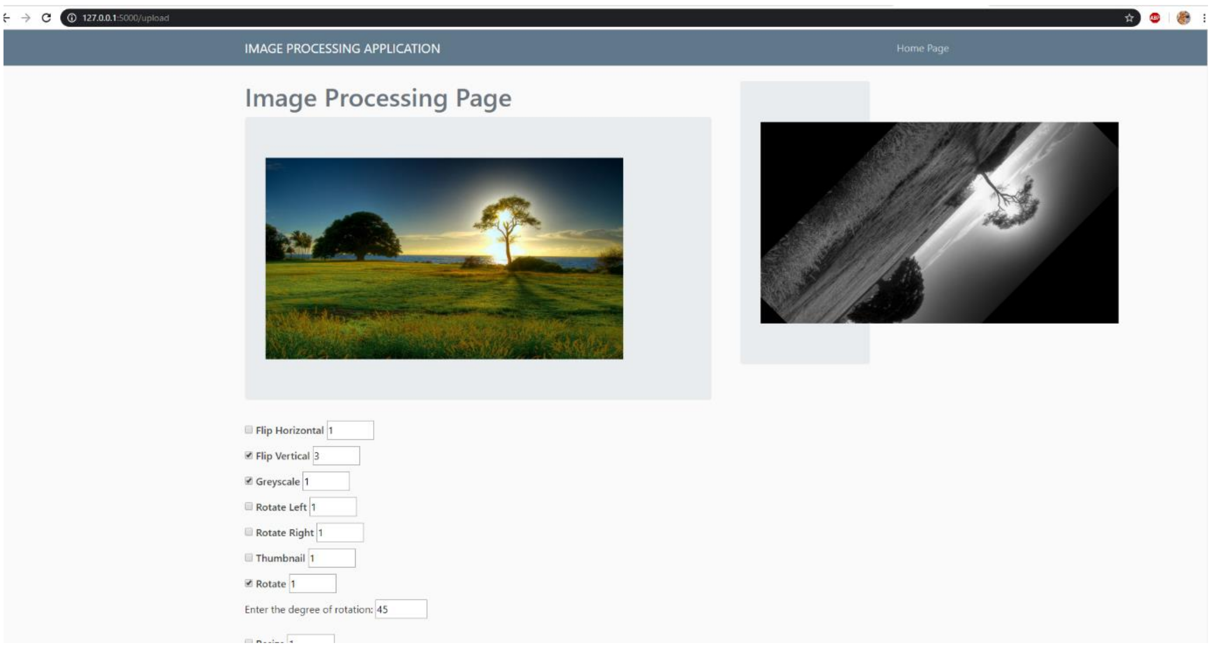
127.0.0.1:5000 says

order of operations selected:: Rotate,Greyscale,FlipV

OK

- [ ] Flip Horizontal 1
- [x] Flip Vertical 3
- [x] Greyscale 1

- The transformed image would be displayed alongside with the original image.



127.0.0.1:5000/upload

IMAGE PROCESSING APPLICATION

Home Page

Image Processing Page

☐ Flip Horizontal 1

☑ Flip Vertical 3

☑ Greyscale 1

☐ Rotate Left 1

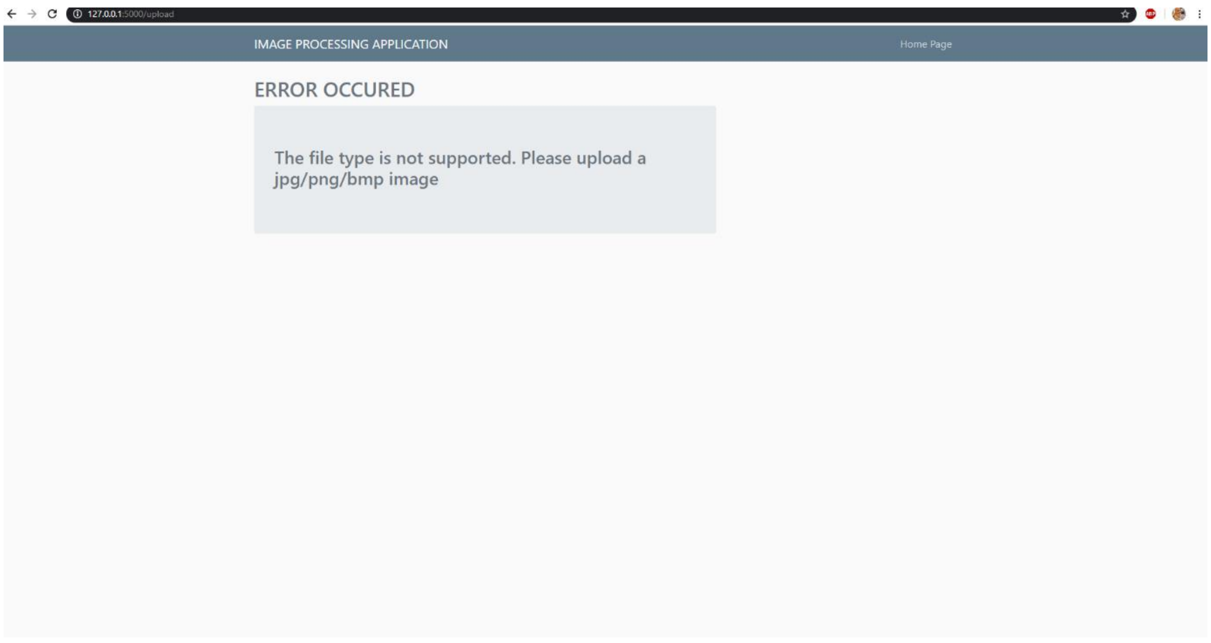
☐ Rotate Right 1

☐ Thumbnail 1

☑ Rotate 1

Enter the degree of rotation: 45

- If a file of unsupported format is uploaded user is taken to the error page.



127.0.0.1:5000/upload

IMAGE PROCESSING APPLICATION

Home Page

ERROR OCCURED

The file type is not supported. Please upload a jpg/png/bmp image