REQUIREMENTS:

- Assumed Python (v.3+) is required and Installed
- Assumed Flask is installed (v.1.1)
- I have used Mac OS as my operating system and used **"Spyder"** as my development IDE, this code will work on any IDE that has Python 3 and Flask installed and configured.
- Download the Zip File and extract everything into a directory of your choice.

How to Run the Code:

- Open a <u>command prompt/Terminal</u> and go the directory where the folder is saved.
- To run the application type: python run.py (or python3 run.py)
- Open the web browser and go to http address show on the line 5 (http://127.0.0.1:5000/) in my case
  1. You will see a Welcome Page
  2. In the browser address line type upload-image (http://127.0.0.1:5000/upload-image)
  3. You will be redirected to a page with choose file and upload image buttons page.
  4. Click on choose file > choose any picture of your choice from your desired location> click upload image.
  5. Once you click on upload image a unique ID is displayed
     ```
     {
       "id": "20191104175203"
     }
     ```
  6. The Image is saved in the directory of where your folder is saved in uploads folder: gemfair >flaskapp >static >uploads >20191104175203.jpg
  7. To search the image with ID, open a new browser and type /download?id= 20191104175203 (http://127.0.0.1:5000/download?id=20191104175203) the image will be displayed.
  8. When the image is displayed you can see the button rotate, which can be used to rotate the image (90 Degrees)

- To run the test:
  python tests.py (python3 tests.py).

Questions:

- What would you have done if you had more time?
  1. I would've implemented the Database to store the images and to serve the user requests
  2. Would've implemented a Cache mechanism to serve the repeated requests to download the image
  3. Would've designed a UI (least priority) to serve and upload the images
  4. Would've designed a queue mechanism to upload the multiple images at once
  5. Logic to handle/serve the large image files

- How did you design this service to meet the high-performance requirement?
  1. Load Balancer to handle the number of requests
  2. Cloud Storage to store the images uploaded by users
  3. Cache mechanism to optimize the download of images

- How did you approach testing this functionality?
    1. I've tested the application from Postman
    2. Tested Upload Image API for different Image formats (I've restricted the image types for JPED, JPG, PNG, GIF)
    3. Tested with unknown Image Id's