Task

Implement a FastAPI server for image processing, ensuring its optimization through multithreading and multiprocessing. The task involves creating a Docker container to run the finished solution.

Description of the server

1. **Data receiving:**
   - The server accepts two images in the format `multipart/form-data`.
2. **Image processing**:
   - Each image is processed in several stages:
     1. Convert to shades of gray.
     2. Blur using Gaussian Blur (the core size is passed as a parameter in the request).
     3. Splitting into equal segments (the segment size is specified as a parameter in the request).
     4. Estimation of the average brightness of each segment. If the average brightness exceeds the specified threshold (a parameter in the request), the segment is marked with a red frame.
3. **Creating a collage:**
   - The processed images are combined into a large collage (grid).
4. **Optimization of processing:**
   - Use **streams** to split and process individual image segments.
   - Use **processes** to process each image separately.
   - The server automatically determines the optimal number of threads/processes depending on system resources.

The result

- The processed collage is returned to the client as a response to the request in the format `application/octet-stream`.
- The collage is also stored on a server called `final_collage.png`.

Docker (Optional)

1. **Containerization:**
    - Create a Docker container to run the FastAPI server.
2. **Launching the container:**

    - The container must be run with parameters that allow you to set the port for the server.
3. **Additional features of the container:**

    - All processed results are saved in the specified output folder (or a standard folder `results/` in a container).

Technical requirements

1. **FastAPI**:

    - Realize a server with two main endpoints:
        - `POST /process` — takes two images, processes them, and returns a collage.
        - `GET /health` — to check the server status.
2. **Image processing:**

    - Use libraries like **OpenCV** and **Pillow** for processing.
3. **Multithreading and multiprocessing:**

    - Realize processing using the **concurrent.futures** and **multiprocessing** libraries.
4. **Dockerfile**:

    - Create a Docker image that automatically installs dependencies (use `requirements.txt`).
    - Enable FastAPI using Uvicorn.

Example of a request:

Request:

```
Unset

    ○ POST /process HTTP/1.1
    ○ Content-Type: multipart/form-data
```

The body of the request:

- `image1`: image file
- `image2`: image file
- `kernel_size`: 5
- `segment_size`: 50
- `brightness_threshold`: 100

Answer:

- Collage in the format `image/png`.