Mindsite Backend Developer Assignment

In this assignment, you are given a FastAPI skeleton. You need to develop an api to download first **50** images (in .zip format) from a given website url. In order to do this you should implement a REST service that provides endpoints to initiate the process of downloading images and to download images when the process is done. **The process should not happen in the request handler!**

We will provide you a function to fetch all image urls from a given url. You should use this function to get all image links from the url.

Please read this document before writing any code.

Expected REST API specification

We would like you to implement following endpoints.

`POST - /download`

Example input:

```
{
    "url": "https://example.com"
}
```

Calling this endpoint should initiate downloading images from the url. Then the endpoint should return a `download_id`, by which we can use to download the images or check the status of the process.

Example output:

```
{
    "download_id": "af0e49d5-45d5-4b74-8378-c86ff944809d"
}
```

Make sure the input of the `POST - /download` endpoint is properly validated and no malformed data is accepted. In case of errors or wrong input, return the appropriate HTTP code.

`GET - /download/<download_id>/status`

This endpoint should accept a single parameter, the `download_id` (generated and returned by the `/download` endpoint). It returns information about the download process.

Example output:

```
{
    "download_id": "af0e49d5-45d5-4b74-8378-c86ff944809d",
```

```
        "started_at": "2021-11-14T22:53:09.974195",
        "finished_at": "2021-11-15T12:56:28.351382",
        "status": "FINISHED",
        "download_url": "http://localhost:5000/download/af0e49d5-45d5-
4b74-8378-c86ff944809d"
    }
```

`GET - /download/<download_id>`

This endpoint should accept a single parameter, the `download_id` (generated and returned by the `/download` endpoint). It returns the zip file.

There should be 3 kinds of responses, since downloading images from a url could take a relatively long time:

- The zip file - in case the downloading finished and images are ready
- Appropriate HTTP response indicating that the downloading of images is not finished yet
- Appropriate HTTP response indicating that some kind of error occurred while downloading images

Skeleton project

For this task, we provide you FastAPI project that has 1 example REST endpoint along with its Dockerfile. Example router and image downloader (you should write endpoints code there) are included in the `api` directory. In `scraper.py`, you should use `get_image_urls` function to fetch image_urls from the given url. There is also `docker-compose.yml` file included. We will use `docker-compose up --build` command to run your project.

Additional description and tips

- You should use a persistent data storage to store ongoing downloading processes. SQLite is sufficent but you can use other persistent databases such as PostgreSQL or Redis. If you use something other than SQLite, don't forget to include it in `docker-compose.yml`
- To process images in the background, you can use asyncio or threads. (You can also use celery but it is not recommended because of additional complexity)
- You can use local file system to store the images you downloaded
- Typing your code is recommended but not mandatory
- Writing tests is highly recommended and will be **evaluated** but it's not strictly mandatory (You can use pytest)
- Modeling your download task objects that is stored in the database is highly recommended
- If you use any additional library, don't forget to **include it in the `requirements.txt` file** (`docker-compose` actually uses it)
- You can use `docker-compose up --build` command to test your own code. Go to `http://localhost:8000/docs` to see auto generated API documentation
- You can add `progress` to download status endpoint as a bonus requirement

Tools needed

- Python 3.7+

- Docker
- docker-compose
- git

Your implementation

Since the project scaffolding is ready-made you can solve the task by modifying only the following file:

- `api/image-downloader.py` - `start_downloading_images`, `get_download_status` and `download_images` functions for their respective endpoints

However if you think the overall project structure or the structure of the Python applications can be improved, please do so.

Also, if for any reason you're not comfortable with **FastAPI** or **asyncio**, feel free to solve this assignment with another web framework. For this reason, we also provided a sync version of `get_image_urls` function in `scraper.py` called `get_image_urls_sync`

We've deliberately left the specific implementation details out - we want to give you space to be creative with your approach to the problem, and impress us with your skills and experience.

Lastly, even if you think your solution is not 100% complete. Please send us your work. If you are stuck or didn't understand the requirements, feel free to contact us on **hakan@themindsite.com**

Good Luck!