ASSIGNMENT 4.4

on

Microservices Best Practice

Submitted by:

Haseebullah Shaikh (2303.KHI.DEG.015)

and

Faiza Gulzar Ahmed (2303.khi.deg.001)

Dated: 14th May 2023

Task: Browse to: tasks/4_microservices_development/day_4_best_practices/app_that_doesnt_follow_best_practices/
analyze the application - which microservice best practices it doesn't follow? Think about what needs to be improved first. Have a look at the areas_for_improvement.txt file for hints. Improve the application

Solution:

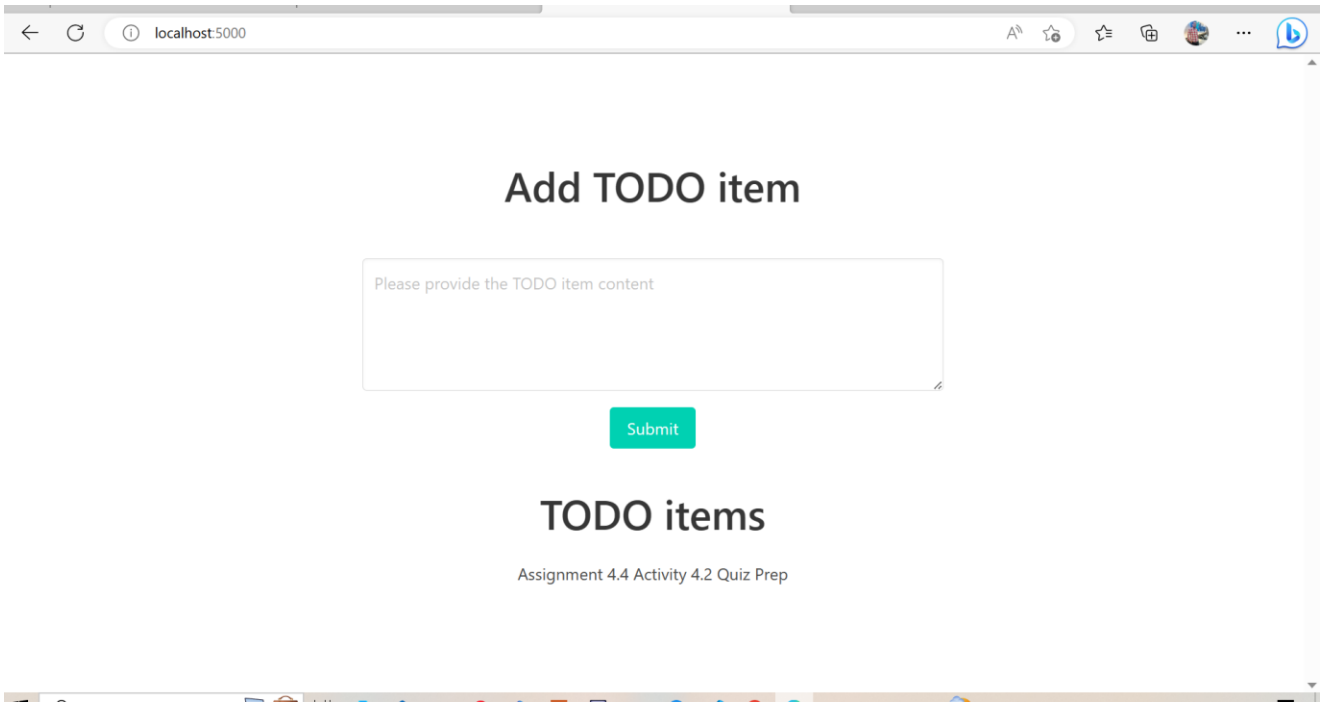
The app has been updated by considering the area of improvements, app and all related files are uploaded on you git repository.

Commands to run app

```
PS C:\Users\dell\Downloads\Perfect_app> docker build -t perfect-app .
[+] Building 4.8s (12/12) FINISHED
 => [internal] load build definition from Dockerfile                                                    0.1s
 => => transferring dockerfile: 404B                                                                    0.1s
 => [internal] load .dockerignore                                                                       0.1s
 => => transferring context: 2B                                                                         0.0s
 => [internal] load metadata for docker.io/library/python:3.8-slim-buster                               4.5s
 => [1/7] FROM docker.io/library/python:3.8-slim-buster@sha256:89ad1c2cd09bda5bc85ada7eb93b5db57d32dc0105b7c942d272  0.0s
 => [internal] load build context                                                                       0.1s
 => => transferring context: 3.11kB                                                                     0.0s
 => CACHED [2/7] WORKDIR /home/app/                                                                     0.0s
 => CACHED [3/7] COPY requirements.txt /home/app/                                                       0.0s
 => CACHED [4/7] RUN pip install --no-cache-dir -r requirements.txt                                     0.0s
 => CACHED [5/7] COPY requirements-dev.txt /home/app/                                                   0.0s
 => CACHED [6/7] RUN pip install --no-cache-dir -r requirements-dev.txt                                 0.0s
 => [7/7] COPY ./ /home/app/                                                                            0.1s
 => exporting to image                                                                                  0.1s
 => => exporting layers                                                                                 0.0s
 => => writing image sha256:e0e15432de6a7a90f50ae7d44eaca312d585dd255bc7db40b208f0bd1c0e641e            0.0s
 => => naming to docker.io/library/perfect-app                                                          0.0s
```

```
PS C:\Users\dell\Downloads\Perfect_app> docker run -p 5000:5000 perfect-app
[2023-05-14 17:21:28 +0000] [1] [INFO] Starting gunicorn 20.1.0
[2023-05-14 17:21:28 +0000] [1] [INFO] Listening at: http://0.0.0.0:5000 (1)
[2023-05-14 17:21:28 +0000] [1] [INFO] Using worker: sync
[2023-05-14 17:21:28 +0000] [8] [INFO] Booting worker with pid: 8
```

App is accesible at port: 5000



Below is the detailed description of each and every thing that has been updated.

Given areas of improvements

1- The logs shouldn't written to a file, but to the container output.

We have used built-in logging module of python to write the logs to the container output. Every log that will be generated will be written to the console standard output.

```
logger = logging.getLogger(__name__)
logger.setLevel(logging.INFO)

stream_handler = logging.StreamHandler()
formatter = logging.Formatter("%(asctime)s,%(msecs)d %(name)s %(levelname)s %(message)s")
stream_handler.setFormatter(formatter)
logger.addHandler(stream_handler)
```

Logs of two created container for running app and statless verification.

```
PS C:\Users\dell\Downloads\Perfect_app> docker logs de0787864ebc
[2023-05-14 17:21:28 +0000] [1] [INFO] Starting gunicorn 20.1.0
[2023-05-14 17:21:28 +0000] [1] [INFO] Listening at: http://0.0.0.0:5000 (1)
[2023-05-14 17:21:28 +0000] [1] [INFO] Using worker: sync
[2023-05-14 17:21:28 +0000] [8] [INFO] Booting worker with pid: 8
PS C:\Users\dell\Downloads\Perfect_app>
```

```
ow
PS C:\Users\dell\Downloads\Perfect_app> docker logs f2f6698f009b
[2023-05-14 17:30:10 +0000] [1] [INFO] Starting gunicorn 20.1.0
[2023-05-14 17:30:10 +0000] [1] [INFO] Listening at: http://0.0.0.0:5000 (1)
[2023-05-14 17:30:10 +0000] [1] [INFO] Using worker: sync
[2023-05-14 17:30:10 +0000] [7] [INFO] Booting worker with pid: 7
PS C:\Users\dell\Downloads\Perfect_app>
```

2- It should be stateless, so that:

- it can easily be restarted without loss of data,

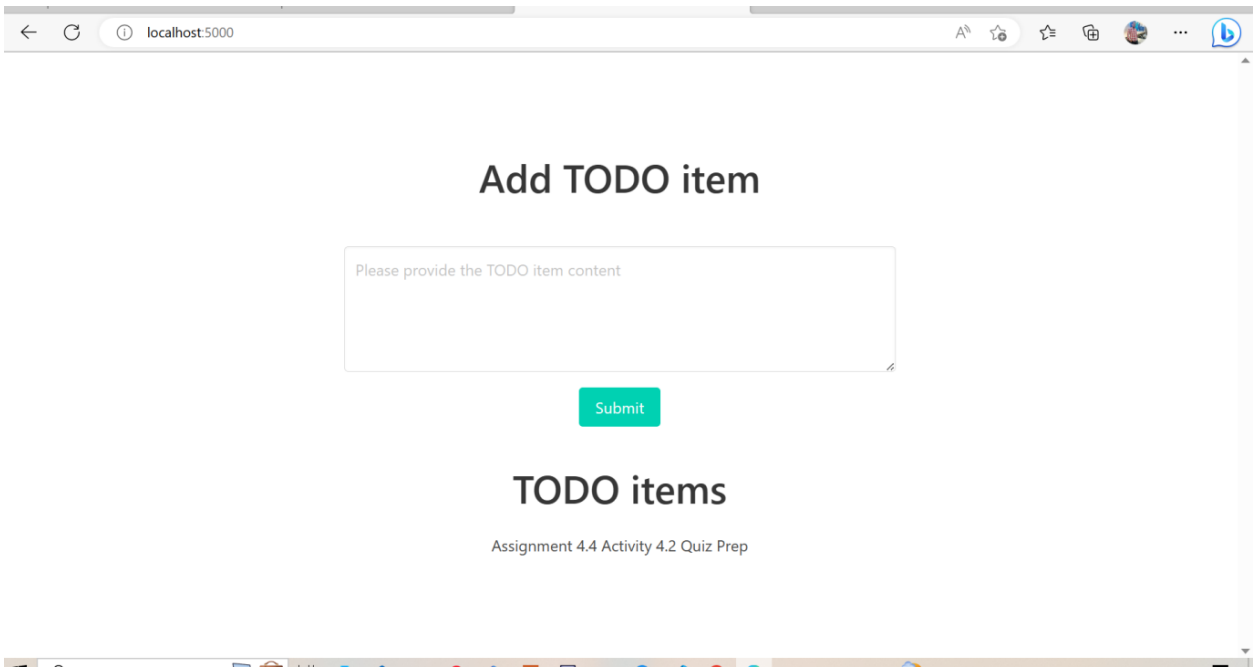
- it is easy to spawn multiple instances of the application.

We have made the app stateless in a way:

- ✓ The SQLite database is implemented to store the data, instead of saving the data locally.
- ✓ The code is defined within the module scope as it's not relying upon any external files.
- ✓ Developed database can accessed by the multiple instances of the application.
- ✓ There will be no any data loss, app can be restarted without data loss.

Stateless Verification

App running on first container, items created and submitted.



Container stopped

Re Running app again

Items are restored, no any data loss

localhost:5000

Add TODO item

Please provide the TODO item content

Submit

TODO items

Assignment 4.4 Activity 4.2 Quiz Prep

Verfying app is accesible at multiple instances and can share the same database.

```
PS C:\Users\dell\Downloads\Perfect_app> docker run -p 5001:5000 perfect-app
[2023-05-14 17:39:30 +0000] [1] [INFO] Starting gunicorn 20.1.0
[2023-05-14 17:39:30 +0000] [1] [INFO] Listening at: http://0.0.0.0:5000 (1)
[2023-05-14 17:39:30 +0000] [1] [INFO] Using worker: sync
[2023-05-14 17:39:30 +0000] [8] [INFO] Booting worker with pid: 8
[2023-05-14 17:41:05 +0000] [1] [INFO] Handling signal: int
```

localhost:5001

Add TODO item

Please provide the TODO item content

Submit

TODO items

item1

3- Requirements installation should be moved from runtime to build time.

We have updated docker file, moved the requirements installation from run time to the build time.

```
Dockerfile > ...
FROM python:3.8-slim-buster

WORKDIR /home/app/

COPY requirements.txt /home/app/
RUN pip install --no-cache-dir -r requirements.txt

COPY requirements-dev.txt /home/app/
RUN pip install --no-cache-dir -r requirements-dev.txt

COPY ./ /home/app/

ENV PYTHONPATH=${PYTHONPATH}:/home/app/
ENV FLASK_ENV=production

CMD ["gunicorn", "main:app", "-b", "0.0.0.0:5000"]
```

4- App should be able to be executed both during development, with debugging enabled, and in production, with debugging disabled.

We have used environment variable "FLASK_ENV" which made it happen to execute app in development with debugging enabled while in production debigging disabled.

Verfying the debugging on both

App is accessible at port 5000

```
PS C:\Users\dell\Downloads\Perfect_app> docker run -d -p 5000:5000 -e FLASK_ENV=development perfect-app
87a6cd896752e608d4a18849369bad015819815e51725cb9766d643811abdc0b
PS C:\Users\dell\Downloads\Perfect_app>
```

```
PS C:\Users\dell\Downloads\Perfect_app> docker run -d -p 5000:5000 -e FLASK_ENV=production perfect-app
cc9b2ce9c4915fe42263c8fb94dc21f379eaba042d05d3da5e90ee95ef2c5f23
PS C:\Users\dell\Downloads\Perfect_app>
```

5- The application should be built in such a way that the database can easily be replaced (development with production instance).

We have achieved it by using enviroment variable “TODO_DB”, by which we have set the path to the SQLite database file .

```
## Using TODO_DB environment variable setting the path to the SQLite database file
app.config['TODO_DB'] = os.environ.get('TODO_DB', 'todo.db')
```

★ *The End* 😊