Application Programming Interface

Definitions

```
Flag – A bit indicating the resource that should be used for
the computations.
```

Project Purpose

The purpose of this project is to create a system which enables the update, storage and retrieval of the `Flag` .

Project Stack

- Python 3
- FastAPI
- Docker
- Redis

Abstract

A simple API to bridge the HoloLight application with our Reinforcement Learning system. The artefact from this project will be used to poll our decision.

Access

This system would be hosted on `localhost` on an available free `port` . Therefore the system requesting resources should allow for input of an port and or IP address when connecting to this API.

Endpoints

```
VAL: [Int] = Placeholder for value stored in system
```

```
Auto
/get-flag
```

`TYPE: GET`

`RESPONSE: JSON =` `{ "renderOnPC": VAL }`

```
Auto
/update-flag
```

`TYPE: POST`

`PARAMETERS:`

```
Plain Text
curl -X POST -d '{"renderOnPC": VAL }' \
-H 'Content-Type: application/json' \
localhost:port
```

`RESPONSE: JSON =` `{"renderOnPc": VAL}`

Source

`poll.py`

```Python
import os
import redis
from fastapi import FastAPI
from pydantic import BaseModel
class RenderInfo(BaseModel): renderOnPC: int
app = FastAPI()
conn = redis.Redis(
    host=os.getenv("REDIS_HOST", "localhost"),
    port=int(os.getenv("REDIS_PORT", 6379)),
    decode_responses = True
)
@app.get("/get-flag")
def get_flag():
    if not conn.exists("renderOnPC"): return {"render-
OnPC": -1}
    return {"renderOnPC": conn.get('renderOnPC')}
@app.post("/update-flag")
def update_flag(info: RenderInfo):
    conn.set("renderOnPC", info.renderOnPC)
    return {"renderOnPC": conn.get('renderOnPC')}
```

`requirements.txt`

Plain Text

```
annotated-types==0.7.0
anyio==4.8.0
async-timeout==5.0.1
certifi==2024.12.14
click==8.1.8
dnspython==2.7.0
email_validator==2.2.0
exceptiongroup==1.2.2
fastapi==0.115.7
fastapi-cli==0.0.7
h11==0.14.0
httpcore==1.0.7
httptools==0.6.4
httpx==0.28.1
idna==3.10
Jinja2==3.1.5
markdown-it-py==3.0.0
MarkupSafe==3.0.2
mdurl==0.1.2
pydantic==2.10.5
pydantic_core==2.27.2
Pygments==2.19.1
python-dotenv==1.0.1
python-multipart==0.0.20
PyYAML==6.0.2
redis==5.2.1
rich==13.9.4
rich-toolkit==0.13.2
shellingham==1.5.4
sniffio==1.3.1
starlette==0.45.2
typer==0.15.1
typing_extensions==4.12.2
uvicorn==0.34.0
uvloop==0.21.0
watchfiles==1.0.4
websockets==14.2
```

Dockerfile

Auto

```
FROM python:3.10-slim
WORKDIR /app
COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt
COPY poll.py ./
EXPOSE 8000
CMD ["uvicorn", "poll:app", "--host", "0.0.0.0", "--
port", "8098"]
```

compose.yaml

Auto

```yaml
services:
  app:
    build:
      context: .
    container_name: my-app
    ports:
      - "0.0.0.0:8098:8098"
    environment:
      - REDIS_HOST=redis
      - REDIS_PORT=6379
    depends_on:
      - redis
  redis:
    image: redis:alpine
    container_name: redis
    ports:
      - "0.0.0.0:6379:6379"
```

`.dockerignore`

```
Auto
venv/
__pycache__/
*.pyc
*.pyo
```

Launch Instructions

Run within the folder:

```
docker-compose up
```

If you would like to request access to the GitHub repository, please let me know!