Objective

The aim of this project is to simulate a Smart Grid system that dynamically balances Electric Vehicle (EV) charging requests across multiple substation services using a custom load balancer, ensuring no substation is overloaded and overall grid health is maintained.

System Architecture

The architecture includes the following services:

- charge_request_service: Public API for clients to submit EV charging requests
- load_balancer: Routes requests to substations based on real-time load metrics
- substation_service: Simulates a substation and exposes its load as Prometheus metrics
- monitoring: Prometheus + Grafana to visualize system performance
- load_tester: Sends simulated high-volume load

Technology Stack:

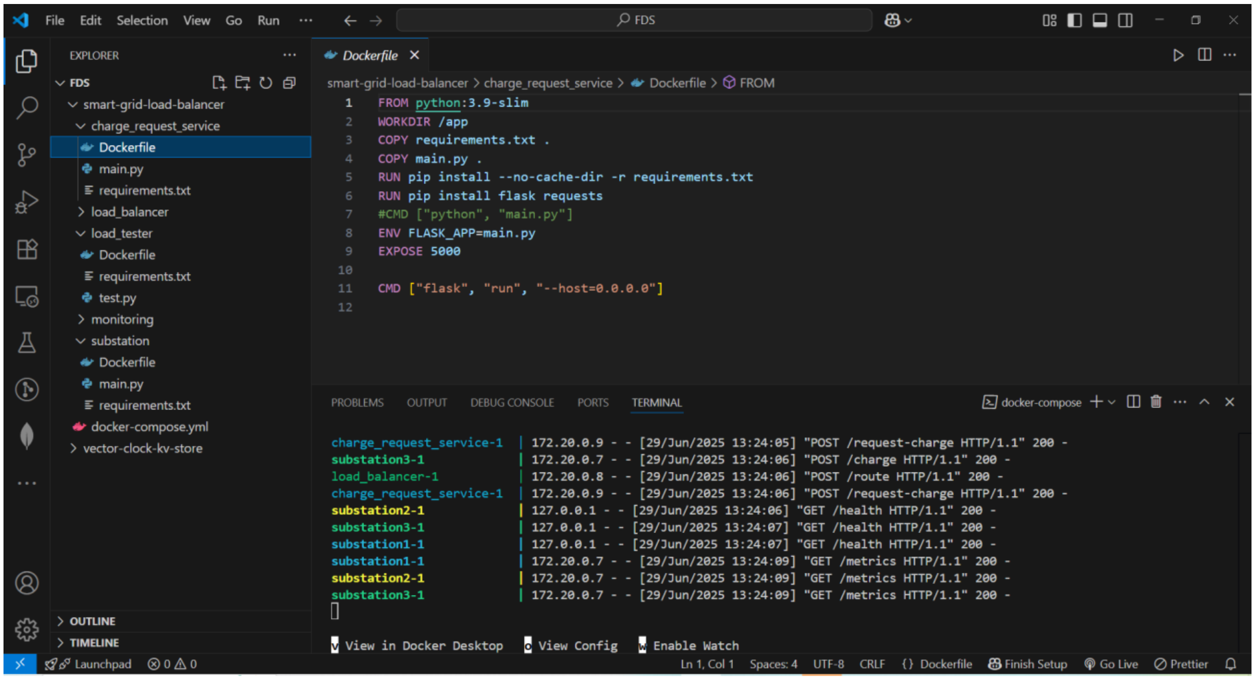
- **Language:** Python (Flask, Requests)
- **Containerization:** Docker, Docker Compose
- **Monitoring:** Prometheus (metrics), Grafana (dashboard)

Component Descriptions

Charge Request Service

Receives charging requests and forwards them to the load balancer.

- Validates input
- Calls POST /route on the load balancer

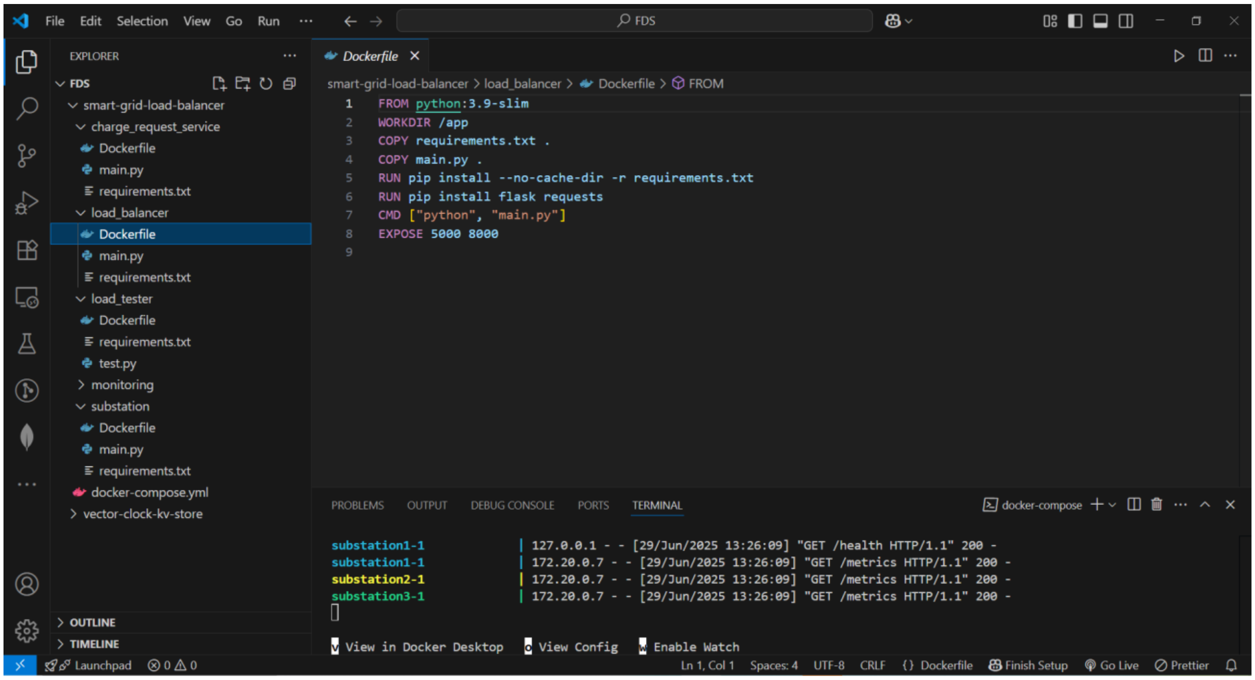


```
FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt .
COPY main.py .
RUN pip install --no-cache-dir -r requirements.txt
RUN pip install flask requests
#CMD ["python", "main.py"]
ENV FLASK_APP=main.py
EXPOSE 5000

CMD ["flask", "run", "--host=0.0.0.0"]

```

```
charge_request_service-1  | 172.20.0.9 - - [29/Jun/2025 13:24:05] "POST /request-charge HTTP/1.1" 200 -
substation3-1             | 172.20.0.7 - - [29/Jun/2025 13:24:06] "POST /charge HTTP/1.1" 200 -
load_balancer-1           | 172.20.0.8 - - [29/Jun/2025 13:24:06] "POST /route HTTP/1.1" 200 -
charge_request_service-1  | 172.20.0.9 - - [29/Jun/2025 13:24:06] "POST /request-charge HTTP/1.1" 200 -
substation2-1             | 127.0.0.1 - - [29/Jun/2025 13:24:06] "GET /health HTTP/1.1" 200 -
substation3-1             | 127.0.0.1 - - [29/Jun/2025 13:24:07] "GET /health HTTP/1.1" 200 -
substation1-1             | 127.0.0.1 - - [29/Jun/2025 13:24:07] "GET /health HTTP/1.1" 200 -
substation1-1             | 172.20.0.7 - - [29/Jun/2025 13:24:09] "GET /metrics HTTP/1.1" 200 -
substation2-1             | 172.20.0.7 - - [29/Jun/2025 13:24:09] "GET /metrics HTTP/1.1" 200 -
substation3-1             | 172.20.0.7 - - [29/Jun/2025 13:24:09] "GET /metrics HTTP/1.1" 200 -

v View in Docker Desktop   o View Config   w Enable Watch
```

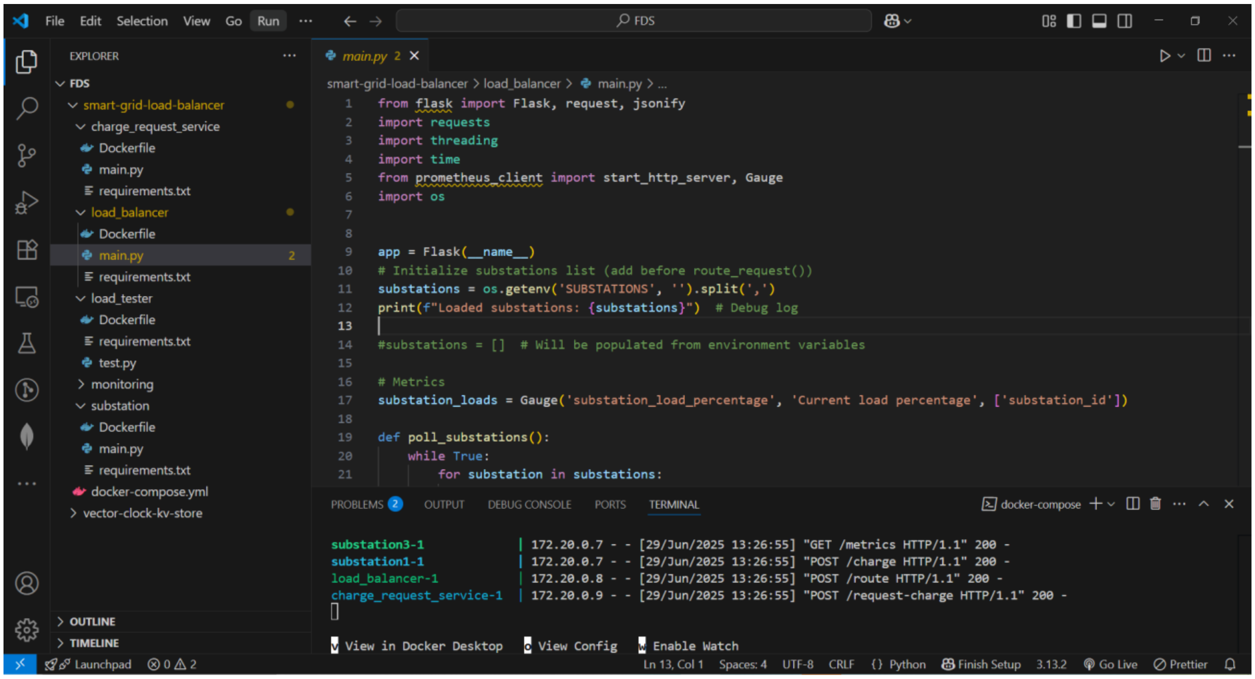
Load Balancer

- Periodically polls all substation /metrics endpoints
- Chooses the least-loaded substation
- Forwards incoming requests using HTTP POST



```
FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt .
COPY main.py .
RUN pip install --no-cache-dir -r requirements.txt
RUN pip install flask requests
CMD ["python", "main.py"]
EXPOSE 5000 8000

```

```
substation1-1             | 127.0.0.1 - - [29/Jun/2025 13:26:09] "GET /health HTTP/1.1" 200 -
substation1-1             | 172.20.0.7 - - [29/Jun/2025 13:26:09] "GET /metrics HTTP/1.1" 200 -
substation2-1             | 172.20.0.7 - - [29/Jun/2025 13:26:09] "GET /metrics HTTP/1.1" 200 -
substation3-1             | 172.20.0.7 - - [29/Jun/2025 13:26:09] "GET /metrics HTTP/1.1" 200 -

v View in Docker Desktop   o View Config   w Enable Watch
```

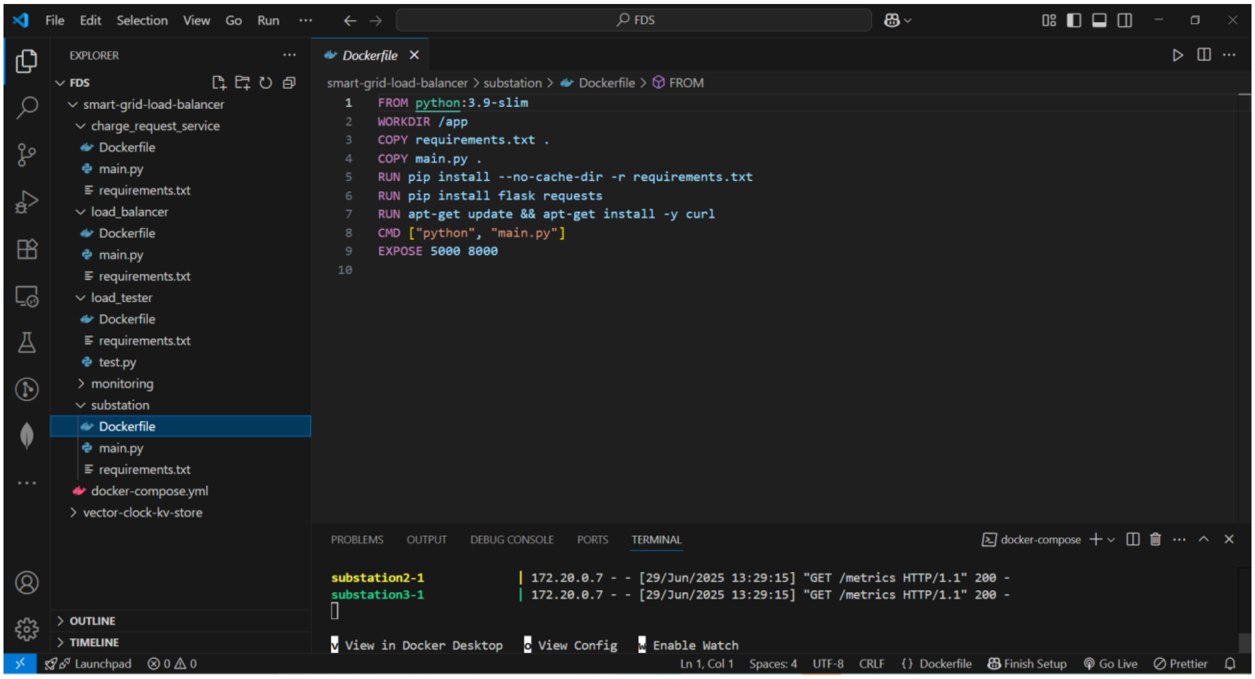


```python
from flask import Flask, request, jsonify
import requests
import threading
import time
from prometheus_client import start_http_server, Gauge
import os


app = Flask(__name__)
# Initialize substations list (add before route_request())
substations = os.getenv('SUBSTATIONS', '').split(',')
print(f"Loaded substations: {substations}")  # Debug log

#substations = []  # Will be populated from environment variables

# Metrics
substation_loads = Gauge('substation_load_percentage', 'Current load percentage', ['substation_id'])

def poll_substations():
    while True:
        for substation in substations:
```

```
substation3-1             | 172.20.0.7 - - [29/Jun/2025 13:26:55] "GET /metrics HTTP/1.1" 200 -
substation1-1             | 172.20.0.7 - - [29/Jun/2025 13:26:55] "POST /charge HTTP/1.1" 200 -
load_balancer-1           | 172.20.0.8 - - [29/Jun/2025 13:26:55] "POST /route HTTP/1.1" 200 -
charge_request_service-1  | 172.20.0.9 - - [29/Jun/2025 13:26:55] "POST /request-charge HTTP/1.1" 200 -

v View in Docker Desktop   o View Config   w Enable Watch
```

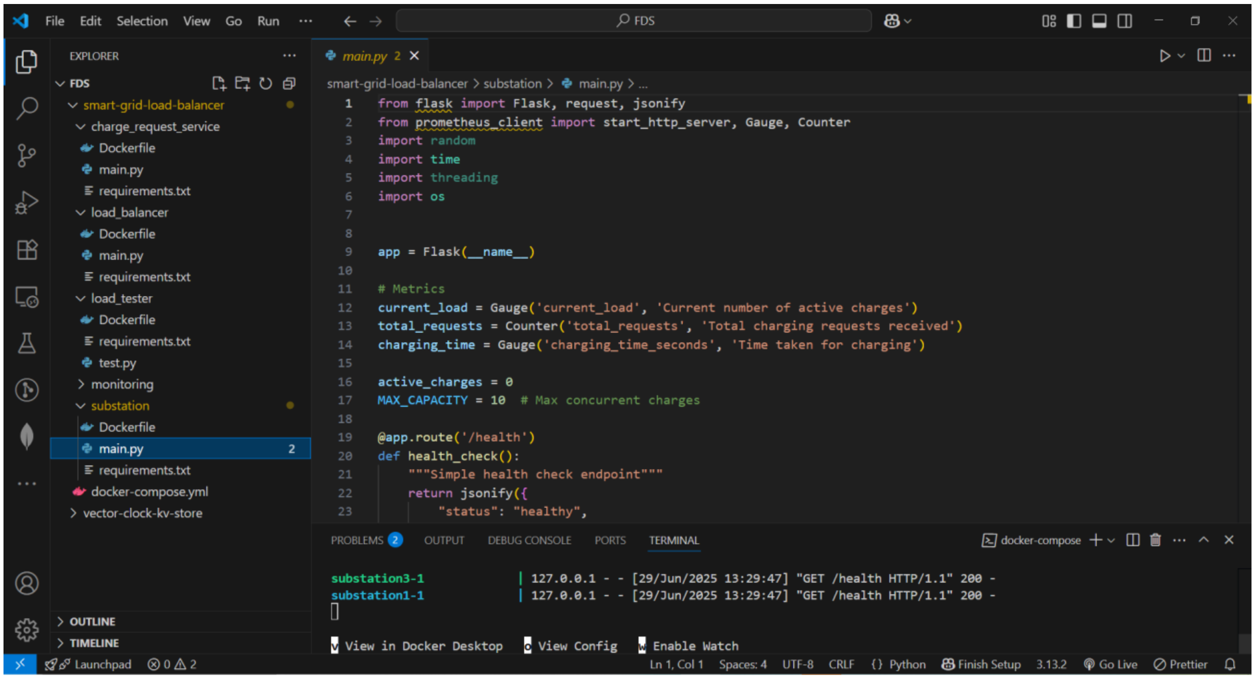
Substation Service

- Simulates charging load
- Increments load on new requests
- Exposes /metrics and /health for monitoring



```dockerfile
FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt .
COPY main.py .
RUN pip install --no-cache-dir -r requirements.txt
RUN pip install flask requests
RUN apt-get update && apt-get install -y curl
CMD ["python", "main.py"]
EXPOSE 5000 8000
```

```
substation2-1             | 172.20.0.7 - - [29/Jun/2025 13:29:15] "GET /metrics HTTP/1.1" 200 -
substation3-1             | 172.20.0.7 - - [29/Jun/2025 13:29:15] "GET /metrics HTTP/1.1" 200 -

v View in Docker Desktop   o View Config   w Enable Watch
```



```python
from flask import Flask, request, jsonify
from prometheus_client import start_http_server, Gauge, Counter
import random
import time
import threading
import os


app = Flask(__name__)

# Metrics
current_load = Gauge('current_load', 'Current number of active charges')
total_requests = Counter('total_requests', 'Total charging requests received')
charging_time = Gauge('charging_time_seconds', 'Time taken for charging')

active_charges = 0
MAX_CAPACITY = 10  # Max concurrent charges

@app.route('/health')
def health_check():
    """Simple health check endpoint"""
    return jsonify({
        "status": "healthy",
```

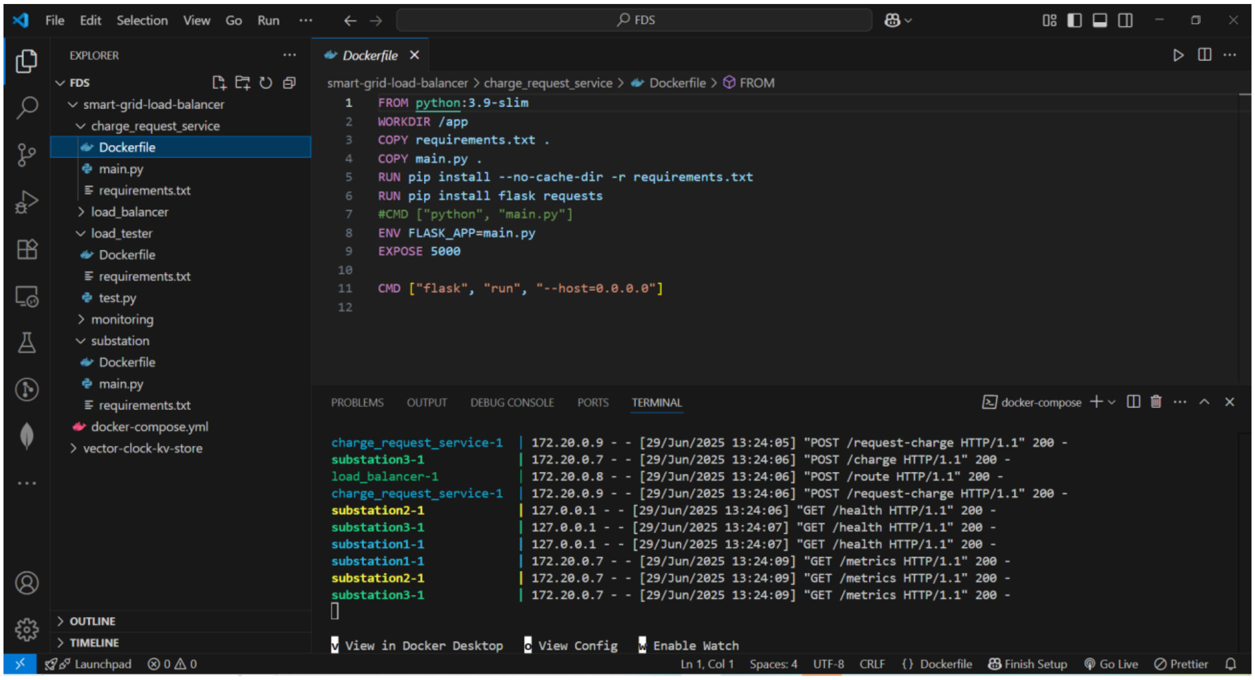
```
substation3-1            | 127.0.0.1 - - [29/Jun/2025 13:29:47] "GET /health HTTP/1.1" 200 -
substation1-1            | 127.0.0.1 - - [29/Jun/2025 13:29:47] "GET /health HTTP/1.1" 200 -


v View in Docker Desktop   o View Config   w Enable Watch
```

Monitoring & Observability

- **Prometheus** scrapes substation metrics
- **Grafana** visualizes substation loads over time
- Helps observe system behavior during load testing

Docker & Orchestration

- All services are containerized with individual **Dockerfiles** and orchestrated using a single docker-compose.yml.
  This allows seamless communication between services and independent scaling of substations.

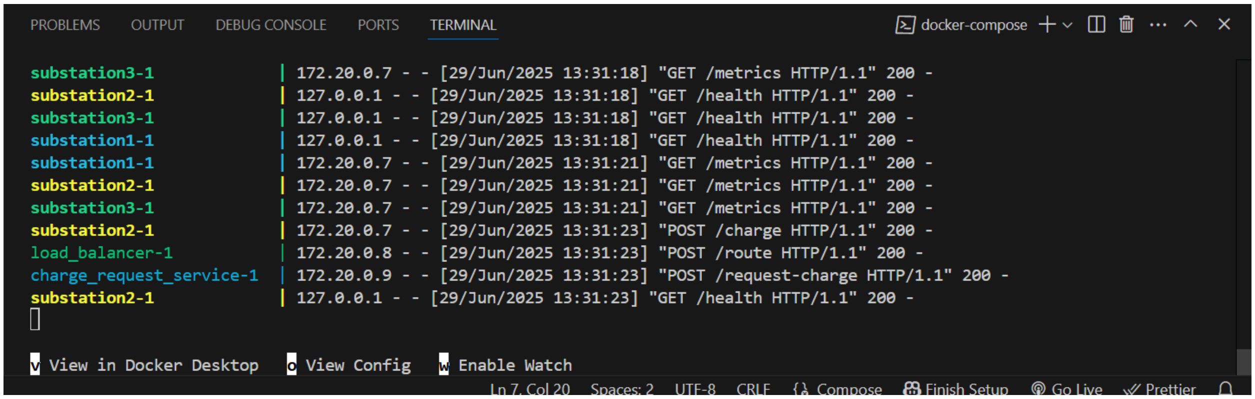


```
FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt .
COPY main.py .
RUN pip install --no-cache-dir -r requirements.txt
RUN pip install flask requests
#CMD ["python", "main.py"]
ENV FLASK_APP=main.py
EXPOSE 5000

CMD ["flask", "run", "--host=0.0.0.0"]

```

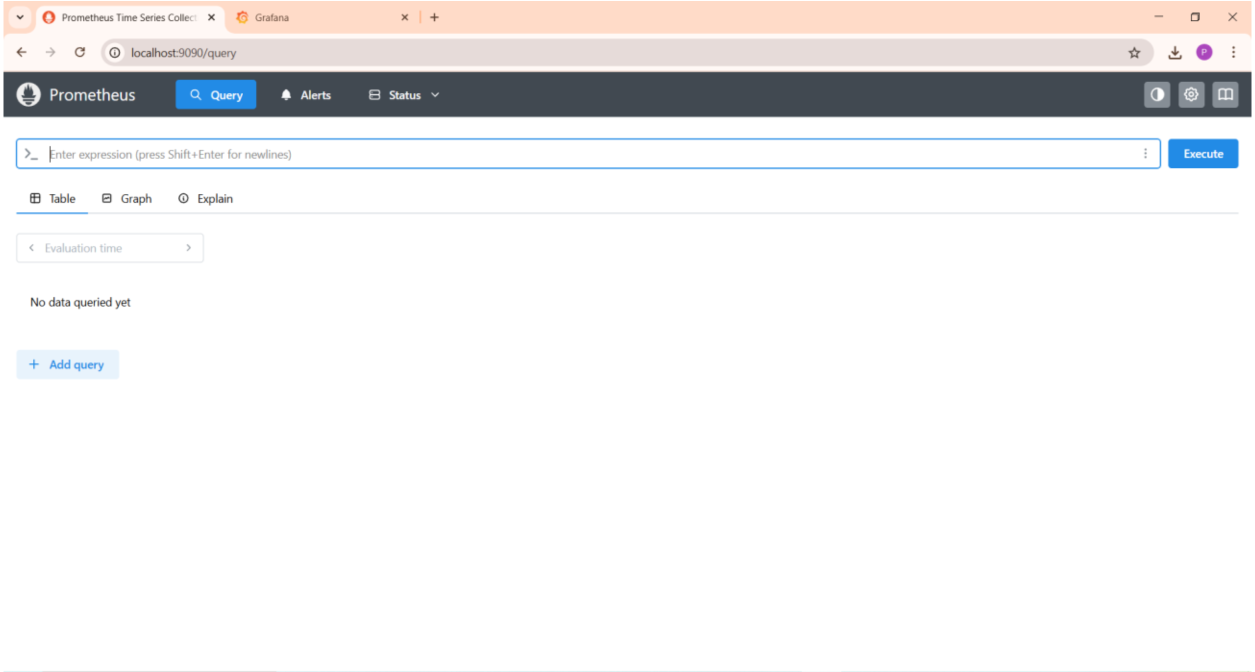
```
charge_request_service-1  | 172.20.0.9 - - [29/Jun/2025 13:24:05] "POST /request-charge HTTP/1.1" 200 -
substation3-1             | 172.20.0.7 - - [29/Jun/2025 13:24:06] "POST /charge HTTP/1.1" 200 -
load_balancer-1           | 172.20.0.8 - - [29/Jun/2025 13:24:06] "POST /route HTTP/1.1" 200 -
charge_request_service-1  | 172.20.0.9 - - [29/Jun/2025 13:24:06] "POST /request-charge HTTP/1.1" 200 -
substation2-1             | 127.0.0.1 - - [29/Jun/2025 13:24:06] "GET /health HTTP/1.1" 200 -
substation3-1             | 127.0.0.1 - - [29/Jun/2025 13:24:07] "GET /health HTTP/1.1" 200 -
substation1-1             | 127.0.0.1 - - [29/Jun/2025 13:24:07] "GET /health HTTP/1.1" 200 -
substation1-1             | 172.20.0.7 - - [29/Jun/2025 13:24:09] "GET /metrics HTTP/1.1" 200 -
substation2-1             | 172.20.0.7 - - [29/Jun/2025 13:24:09] "GET /metrics HTTP/1.1" 200 -
substation3-1             | 172.20.0.7 - - [29/Jun/2025 13:24:09] "GET /metrics HTTP/1.1" 200 -

v View in Docker Desktop   o View Config   w Enable Watch
```

Load Testing & Observations

- Load testing simulates multiple vehicle charge requests during a "rush hour" window.
- Prometheus and Grafana show that the load balancer successfully routes requests to the least loaded substations.
- No substation crosses overload threshold.


PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL
docker-compose
Ln 7, Col 20 Spaces: 2 UTF-8 CRLF {} Compose Finish Setup Go Live Prettier

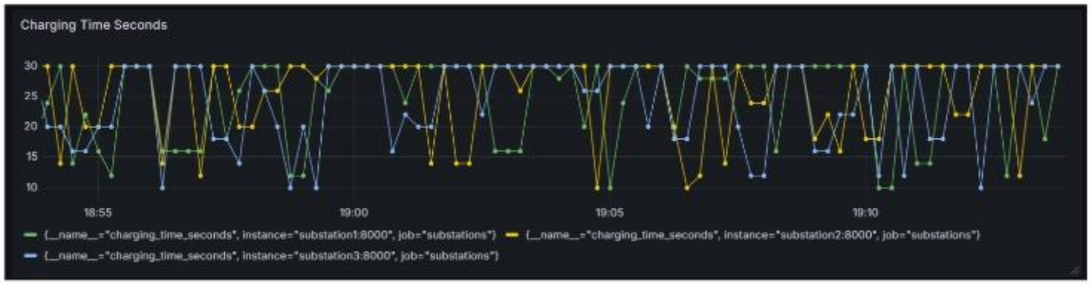
```
substation3-1             | 172.20.0.7 - - [29/Jun/2025 13:31:18] "GET /metrics HTTP/1.1" 200 -
substation2-1             | 127.0.0.1 - - [29/Jun/2025 13:31:18] "GET /health HTTP/1.1" 200 -
substation3-1             | 127.0.0.1 - - [29/Jun/2025 13:31:18] "GET /health HTTP/1.1" 200 -
substation1-1             | 127.0.0.1 - - [29/Jun/2025 13:31:18] "GET /health HTTP/1.1" 200 -
substation1-1             | 172.20.0.7 - - [29/Jun/2025 13:31:21] "GET /metrics HTTP/1.1" 200 -
substation2-1             | 172.20.0.7 - - [29/Jun/2025 13:31:21] "GET /metrics HTTP/1.1" 200 -
substation3-1             | 172.20.0.7 - - [29/Jun/2025 13:31:21] "GET /metrics HTTP/1.1" 200 -
substation2-1             | 172.20.0.7 - - [29/Jun/2025 13:31:23] "POST /charge HTTP/1.1" 200 -
load_balancer-1           | 172.20.0.8 - - [29/Jun/2025 13:31:23] "POST /route HTTP/1.1" 200 -
charge_request_service-1  | 172.20.0.9 - - [29/Jun/2025 13:31:23] "POST /request-charge HTTP/1.1" 200 -
substation2-1             | 127.0.0.1 - - [29/Jun/2025 13:31:23] "GET /health HTTP/1.1" 200 -

v View in Docker Desktop   o View Config   w Enable Watch
```



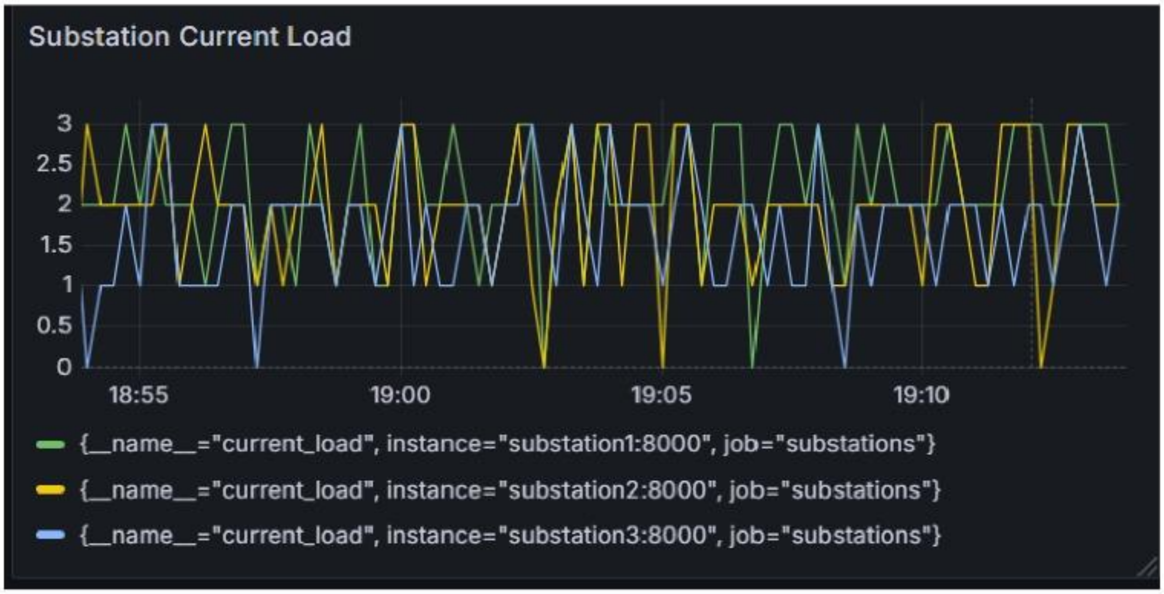
No data queried yet

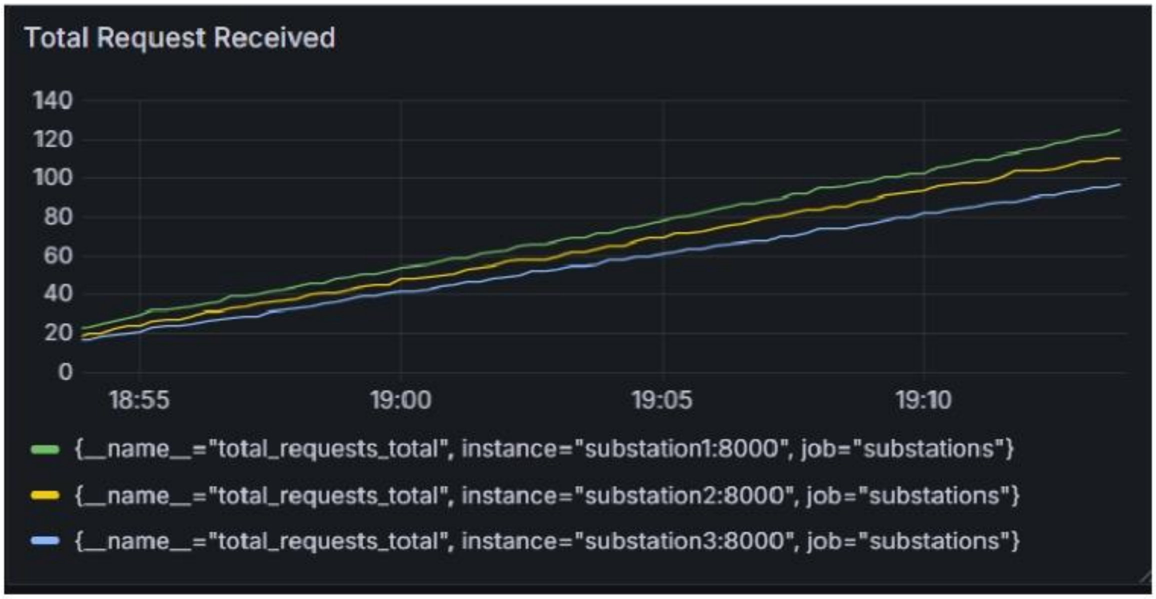
Grafana Dashboard for monitoring load:

Charging Time (in secs)



Substation Load





Conclusion

The Smart Grid load balancing system was successfully implemented using microservices. The system:

- Distributes charging load intelligently

- Exposes real-time metrics
- Demonstrates scalability through service replicas
- Ensures fairness and grid stability under stress