FDS Assignment Part 2: Smart Grid Load Balancer with Monitoring

Name : Mohamed Bin Badhusha E B
Roll No : g24ai2026

1. Introduction

This project presents a dynamic and scalable system for handling Electric Vehicle (EV) charging requests in a Smart Grid setup. The main objective is to distribute incoming charging requests across multiple substations based on their real-time load, ensuring optimal usage of resources and preventing overload.

The system is composed of containerized microservices built using Python and Docker, and uses Prometheus and Grafana for observability and monitoring. It simulates real-world “rush hour” scenarios and analyzes system performance using dashboards.

2. Architecture Diagram

[EV Request Simulator]

↓

[Charge Request Service]

↓

[Load Balancer]

↙ ↘

[Substation A] [Substation B]

↓ ↓

[Prometheus] ⟷ [Grafana Dashboard]

The architecture consists of:

- EVs sending charging requests to a centralized Load Balancer
- Load Balancer polling two Substation Services
- Substations expose real-time load via /metrics
- Prometheus collects time-series data
- Grafana visualizes the substation loads live

3. System Components

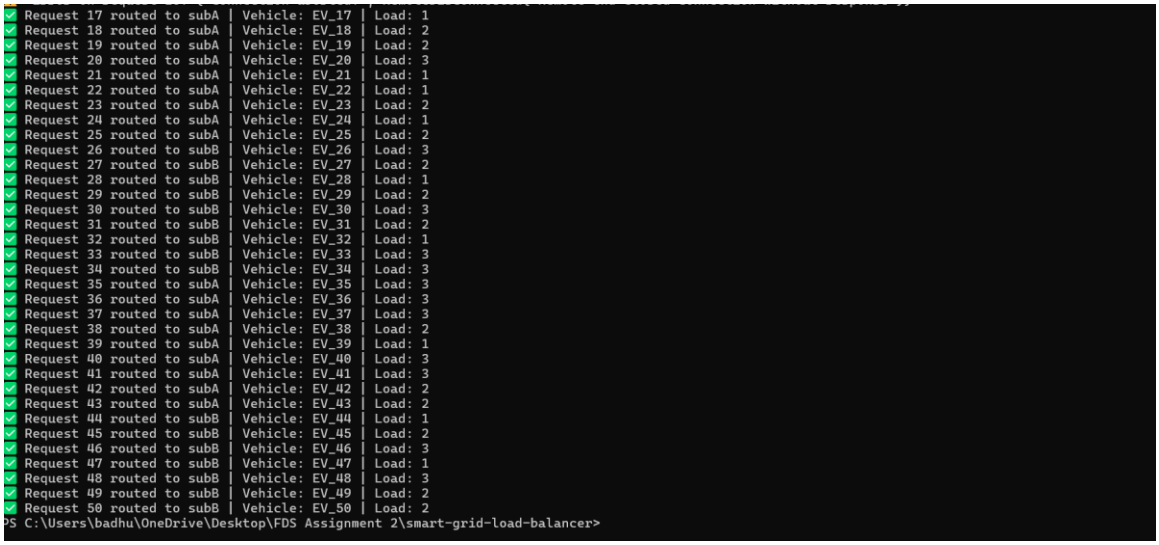
Component	Technology	Functionality
Substation	Python + Flask	Accepts charging requests, exposes current load metrics
Load Balancer	Python + Flask	Routes EV requests to least-loaded substation
Prometheus	prom/prometheus	Collects time-series data from substations
Grafana	grafana/grafana	Visualizes data from Prometheus in dashboards
Load Tester	Python	Simulates 50 EV charging requests
Docker Compose	YAML	Orchestrates all components

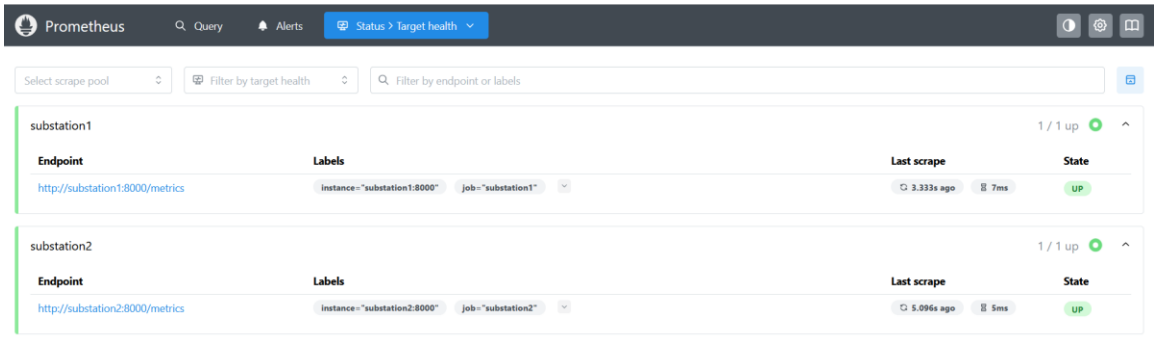
4. Execution Flow

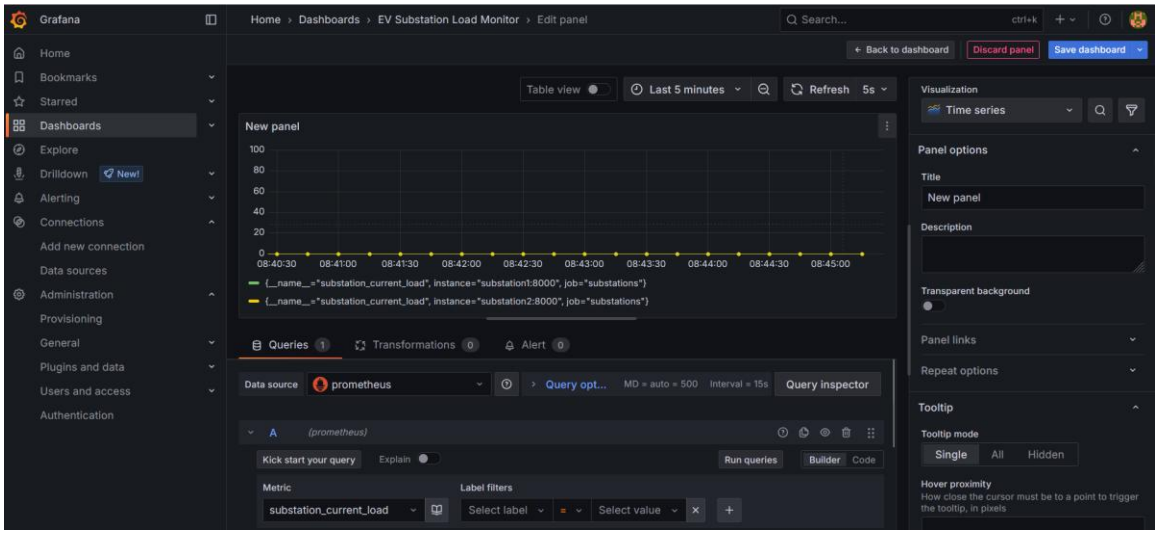
- Substation services start with Flask, exposing /charge and /metrics.
- Load balancer polls both substations every 2 seconds for current load.
- Charging requests are routed to the least-loaded substation.
- Prometheus scrapes metrics and feeds Grafana dashboard.
- Load tester simulates traffic with randomized requests.

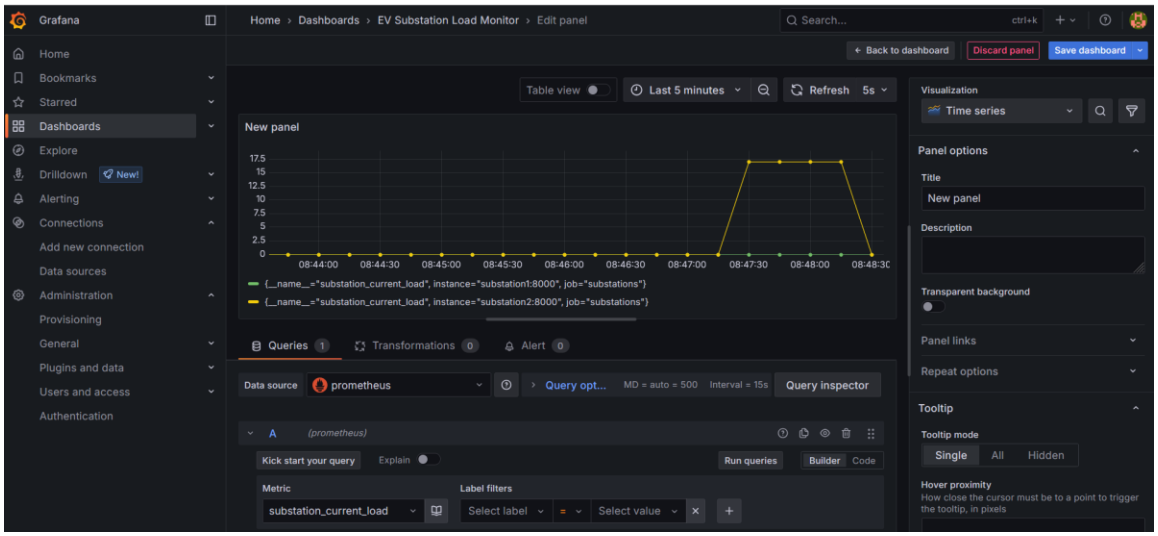
5. Observability & Monitoring

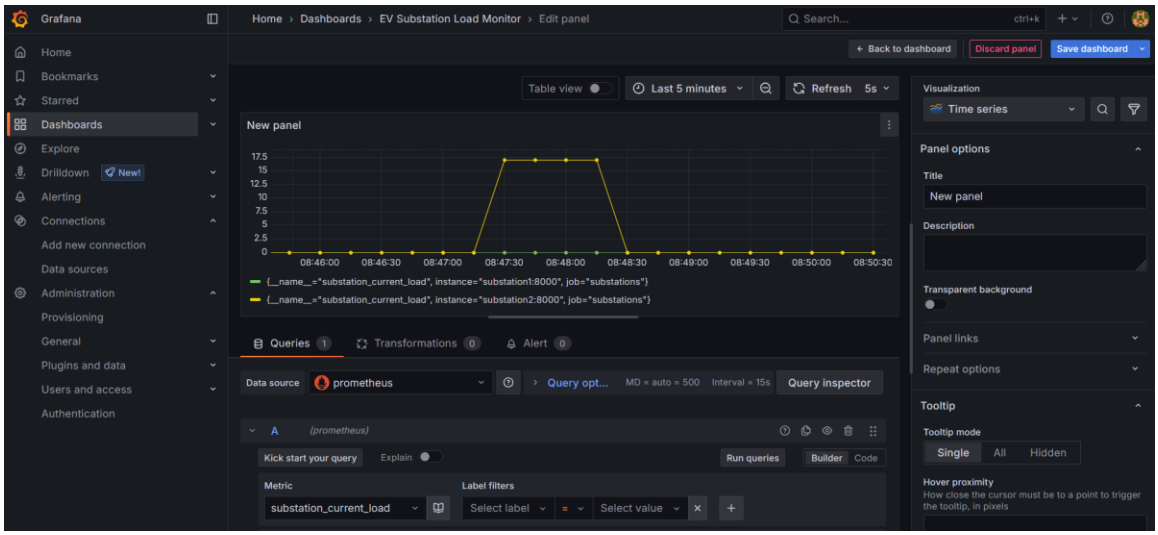
Prometheus is configured with prometheus.yml to scrape data from both substations.
Grafana uses a custom dashboard (dashboard.json) to visualize the metric substation_current_load.
The dashboard auto-refreshes every 5 seconds to show live data.


✅ Request 17 routed to subA | Vehicle: EV_17 | Load: 1
✅ Request 18 routed to subA | Vehicle: EV_18 | Load: 2
✅ Request 19 routed to subA | Vehicle: EV_19 | Load: 2
✅ Request 20 routed to subA | Vehicle: EV_20 | Load: 3
✅ Request 21 routed to subA | Vehicle: EV_21 | Load: 1
✅ Request 22 routed to subA | Vehicle: EV_22 | Load: 1
✅ Request 23 routed to subA | Vehicle: EV_23 | Load: 2
✅ Request 24 routed to subA | Vehicle: EV_24 | Load: 1
✅ Request 25 routed to subA | Vehicle: EV_25 | Load: 2
✅ Request 26 routed to subB | Vehicle: EV_26 | Load: 3
✅ Request 27 routed to subB | Vehicle: EV_27 | Load: 2
✅ Request 28 routed to subB | Vehicle: EV_28 | Load: 1
✅ Request 29 routed to subB | Vehicle: EV_29 | Load: 2
✅ Request 30 routed to subB | Vehicle: EV_30 | Load: 3
✅ Request 31 routed to subB | Vehicle: EV_31 | Load: 2
✅ Request 32 routed to subB | Vehicle: EV_32 | Load: 1
✅ Request 33 routed to subB | Vehicle: EV_33 | Load: 3
✅ Request 34 routed to subB | Vehicle: EV_34 | Load: 3
✅ Request 35 routed to subA | Vehicle: EV_35 | Load: 3
✅ Request 36 routed to subA | Vehicle: EV_36 | Load: 3
✅ Request 37 routed to subA | Vehicle: EV_37 | Load: 3
✅ Request 38 routed to subA | Vehicle: EV_38 | Load: 2
✅ Request 39 routed to subA | Vehicle: EV_39 | Load: 1
✅ Request 40 routed to subA | Vehicle: EV_40 | Load: 3
✅ Request 41 routed to subA | Vehicle: EV_41 | Load: 3
✅ Request 42 routed to subA | Vehicle: EV_42 | Load: 2
✅ Request 43 routed to subA | Vehicle: EV_43 | Load: 2
✅ Request 44 routed to subB | Vehicle: EV_44 | Load: 1
✅ Request 45 routed to subB | Vehicle: EV_45 | Load: 2
✅ Request 46 routed to subB | Vehicle: EV_46 | Load: 3
✅ Request 47 routed to subB | Vehicle: EV_47 | Load: 1
✅ Request 48 routed to subB | Vehicle: EV_48 | Load: 3
✅ Request 49 routed to subB | Vehicle: EV_49 | Load: 2
✅ Request 50 routed to subB | Vehicle: EV_50 | Load: 2
PS C:\Users\badhu\OneDrive\Desktop\FDS Assignment 2\smart-grid-load-balancer>


Prometheus
Query
Alerts
Status > Target health
Select scrape pool
Filter by target health
Filter by endpoint or labels
substation1
1 / 1 up
Endpoint
Labels
Last scrape
State
http://substation1:8000/metrics
instance="substation1:8000"
job="substation1"
3.333s ago
7ms
UP
substation2
1 / 1 up
Endpoint
Labels
Last scrape
State
http://substation2:8000/metrics
instance="substation2:8000"
job="substation2"
5.096s ago
5ms
UP


Grafana
Home › Dashboards › EV Substation Load Monitor › Edit panel
Search...
ctrl+k
Back to dashboard
Discard panel
Save dashboard
Home
Bookmarks
Starred
Dashboards
Explore
Drilldown
New!
Alerting
Connections
Add new connection
Data sources
Administration
Provisioning
General
Plugins and data
Users and access
Authentication
Table view
Last 5 minutes
Refresh
5s
New panel
100
80
60
40
20
0
08:40:30
08:41:00
08:41:30
08:42:00
08:42:30
08:43:00
08:43:30
08:44:00
08:44:30
08:45:00
{__name__="substation_current_load", instance="substation1:8000", job="substations"}
{__name__="substation_current_load", instance="substation2:8000", job="substations"}
Queries 1
Transformations 0
Alert 0
Data source
prometheus
Query opt...
MD = auto = 500
Interval = 15s
Query inspector
A
(prometheus)
Kick start your query
Explain
Run queries
Builder
Code
Metric
Label filters
substation_current_load
Select label
=
Select value
Visualization
Time series
Panel options
Title
New panel
Description
Transparent background
Panel links
Repeat options
Tooltip
Tooltip mode
Single
All
Hidden
Hover proximity
How close the cursor must be to a point to trigger the tooltip, in pixels


Grafana
Home › Dashboards › EV Substation Load Monitor › Edit panel
Search...
ctrl+k
Back to dashboard
Discard panel
Save dashboard
Home
Bookmarks
Starred
Dashboards
Explore
Drilldown
New!
Alerting
Connections
Add new connection
Data sources
Administration
Provisioning
General
Plugins and data
Users and access
Authentication
Table view
Last 5 minutes
Refresh
5s
New panel
17.5
15
12.5
10
7.5
5
2.5
0
08:44:00
08:44:30
08:45:00
08:45:30
08:46:00
08:46:30
08:47:00
08:47:30
08:48:00
08:48:30
{__name__="substation_current_load", instance="substation1:8000", job="substations"}
{__name__="substation_current_load", instance="substation2:8000", job="substations"}
Queries 1
Transformations 0
Alert 0
Data source
prometheus
Query opt...
MD = auto = 500
Interval = 15s
Query inspector
A
(prometheus)
Kick start your query
Explain
Run queries
Builder
Code
Metric
Label filters
substation_current_load
Select label
=
Select value
Visualization
Time series
Panel options
Title
New panel
Description
Transparent background
Panel links
Repeat options
Tooltip
Tooltip mode
Single
All
Hidden
Hover proximity
How close the cursor must be to a point to trigger the tooltip, in pixels



6. Load Testing

The load_tester/test.py script sends 50 EV requests with random loads between 1 and 3. Each request is routed through the load balancer to the optimal substation based on current load. This confirms proper dynamic load balancing functionality.

7. Video Demonstration

Video Link Here –
https://drive.google.com/file/d/1juAMJLzrWHkEl7r6JaHG09z3JhRWYs7r/view?usp=sharing

- All containers running via Docker Compose
- Load test script sending EV requests
- Grafana dashboard live updates

8. GitHub Repository

Public GitHub Repo Link Here - https://github.com/MohamedBinBadhusha/smart-grid-load-balancer/tree/main

Repo follows this structure:

```
smart-grid-load-balancer/
 ├── charge_request_service/
 ├── load_balancer/
 ├── substation_service/
 ├── load_tester/
 ├── monitoring/
|    ├── prometheus/
|   └── grafana/
 ├── docker-compose.yml
└── project_report.pdf
```

9. Conclusion

This Smart Grid Load Balancer system demonstrates how microservices, monitoring tools, and load-aware routing can efficiently distribute EV charging requests. The modular architecture can be extended with predictive algorithms, multi-region deployment, or integration with real-time IoT data.