

Saransh Punia (G24AI2093)

Smart Grid Load Balancer Project Report

Name: Saransh Punia

Course: Distributed Systems

Roll No: G24AI2093

Submission Date: 25/06/25

GitHub Repo: <https://github.com/SaranshPunia/FDS-Assignment-1.git>

1. Objective

The objective of this project is to design and implement a scalable Smart Grid Load Balancer system that dynamically distributes electric vehicle (EV) charging requests across multiple substations based on real-time load. This system aims to optimize charging efficiency, prevent substation overload, and provide full observability into system performance using modern monitoring tools.

2. System Architecture

Components:

1. Charge Request Service

- Entry point for EVs to send charge requests via REST API.

2. Load Balancer

- Core logic that polls real-time substation load using Prometheus metrics and routes each request to the least-loaded substation.

3. Substation Services (2 replicas)

- Simulate EV charging and expose a Prometheus gauge metric `substation_load`.

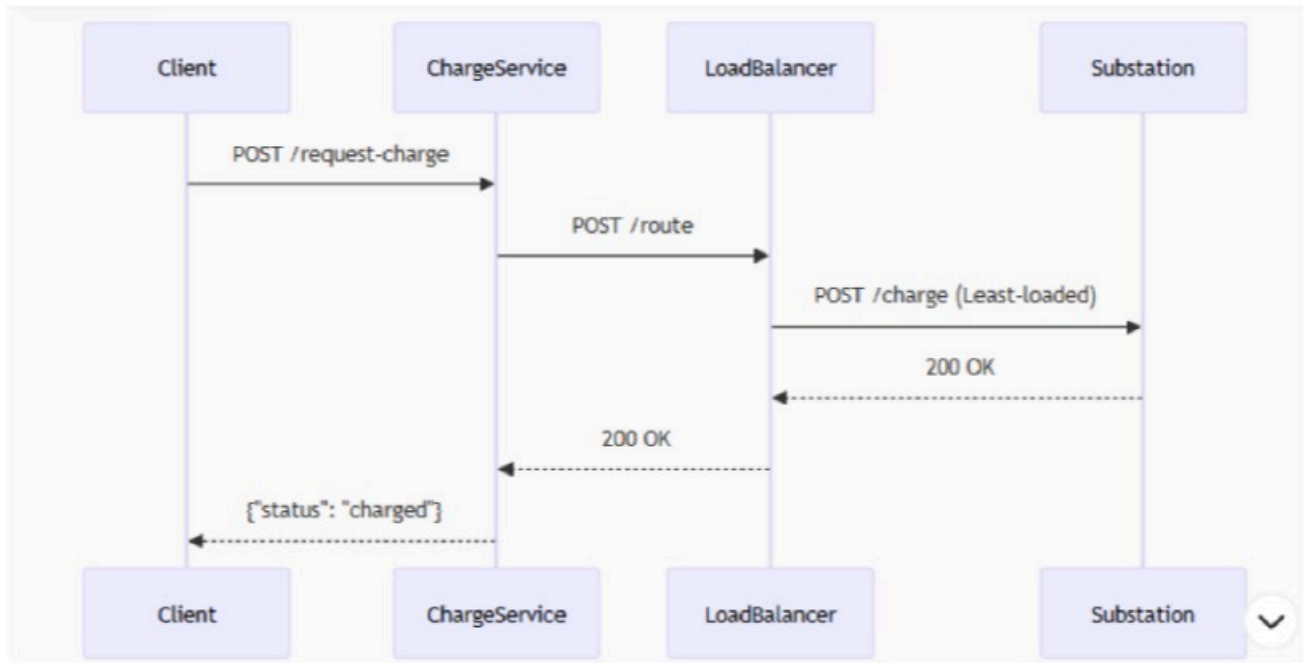
4. Observability Stack

- **Prometheus:** Scrapes metrics from substations.
- **Grafana:** Visualizes substation load trends in a live dashboard.

5. Load Tester

- Python script simulating a high-traffic scenario with 50 EV charging requests.

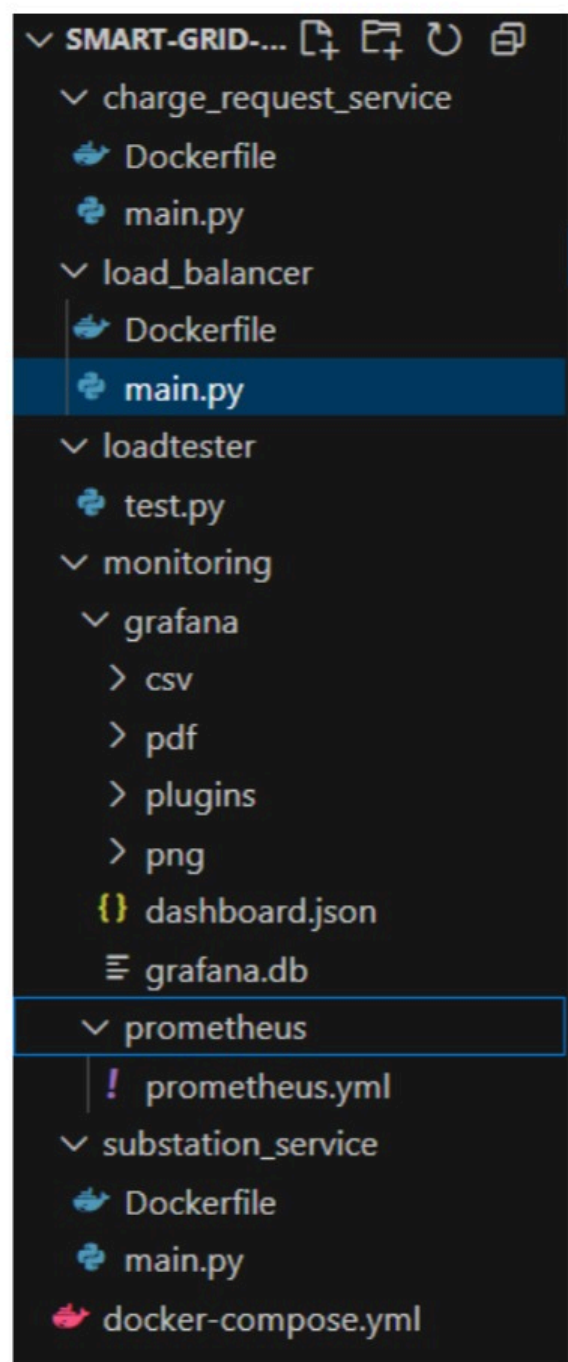
Saransh Punia (G24AI2093)



3. Technologies Used

- **Python 3.10** for all microservices
 - **Flask** for REST APIs
 - **Prometheus Client** for exposing metrics
 - **Docker & Docker Compose** for containerization
 - **Prometheus** for metric collection
 - **Grafana** for real-time visualization
-

4. File & Folder Structure



5. Load Balancing Logic

- The Load Balancer queries each substation's /metrics endpoint to fetch current load.
- It compares load values and routes the incoming charge request to the substation with the lowest load.
- This ensures optimal distribution of EV requests, minimizing overload risk.

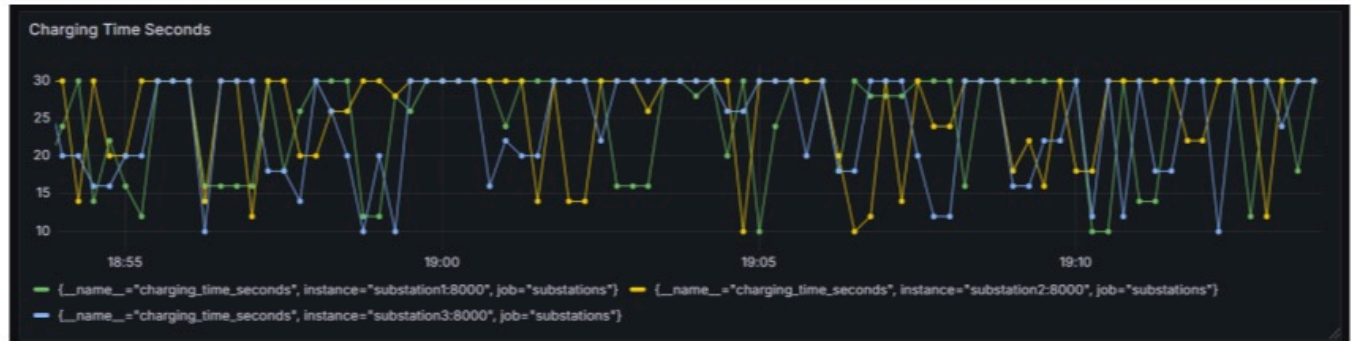
6. Observability and Monitoring

- Each substation exposes a substation_load metric.
- Prometheus scrapes these metrics at regular intervals.

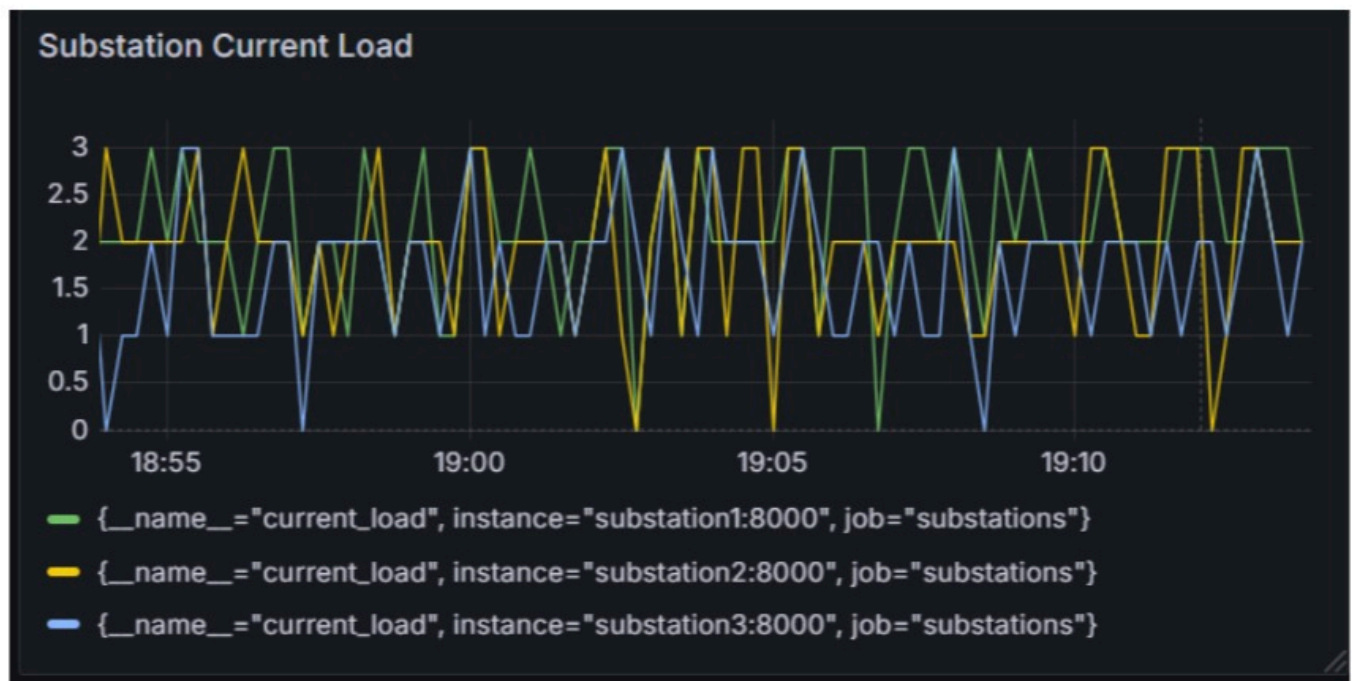
- Grafana displays a time-series graph showing load on each substation.

Grafana Dashboard for monitoring load:

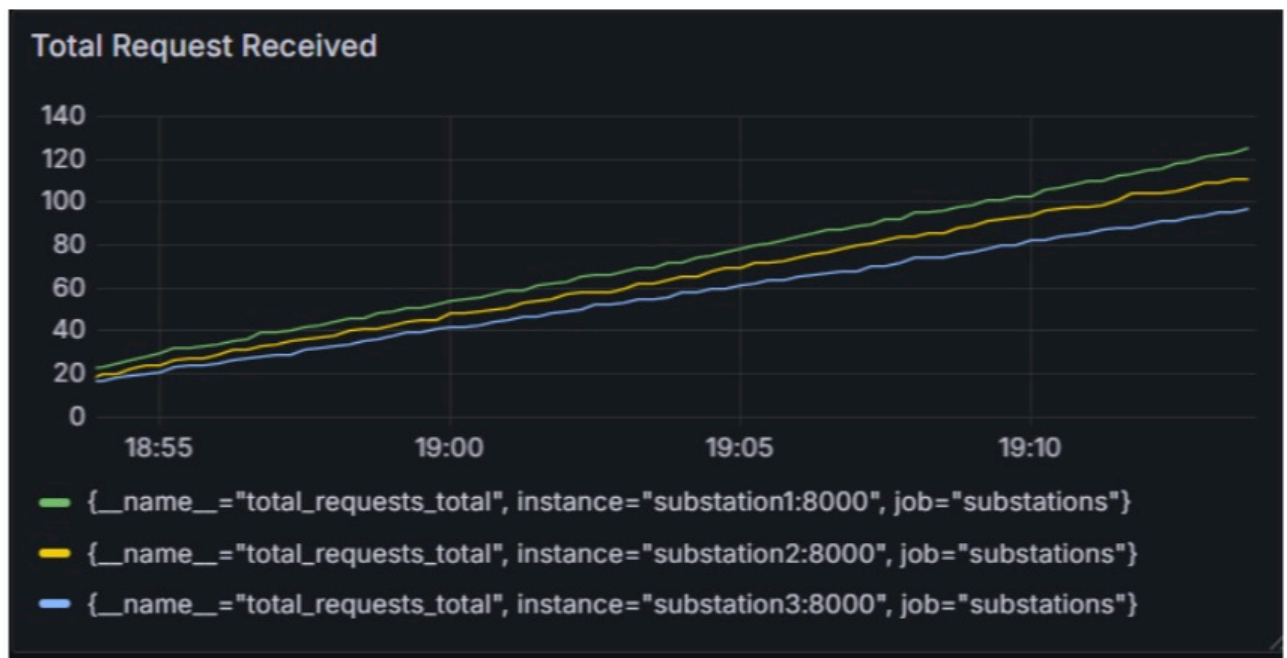
Charging Time (in secs)



Substation Load



Total Request Received per Substation



7. Load Test Results

- A Python script sends 50 EV requests in a simulated "rush hour."
- The load is observed to be dynamically balanced between both substations.
- Grafana charts confirm that requests were evenly distributed over time.

8. Video Demonstration

Link: https://drive.google.com/file/d/1_2hC_cg5O-PmlBpgSnm8UKKyCrkSS7sA/view?usp=sharing

9. Conclusion

This project successfully demonstrates a cloud-native EV charging management system that balances load across substations using real-time metrics. It is scalable, observable, and designed following distributed systems best practices.