Leader email: kprakash@iastate.edu
Faculty Advisor and Client: Dr. Ravikumar Gelli

GridAI: PIRM Meeting 1

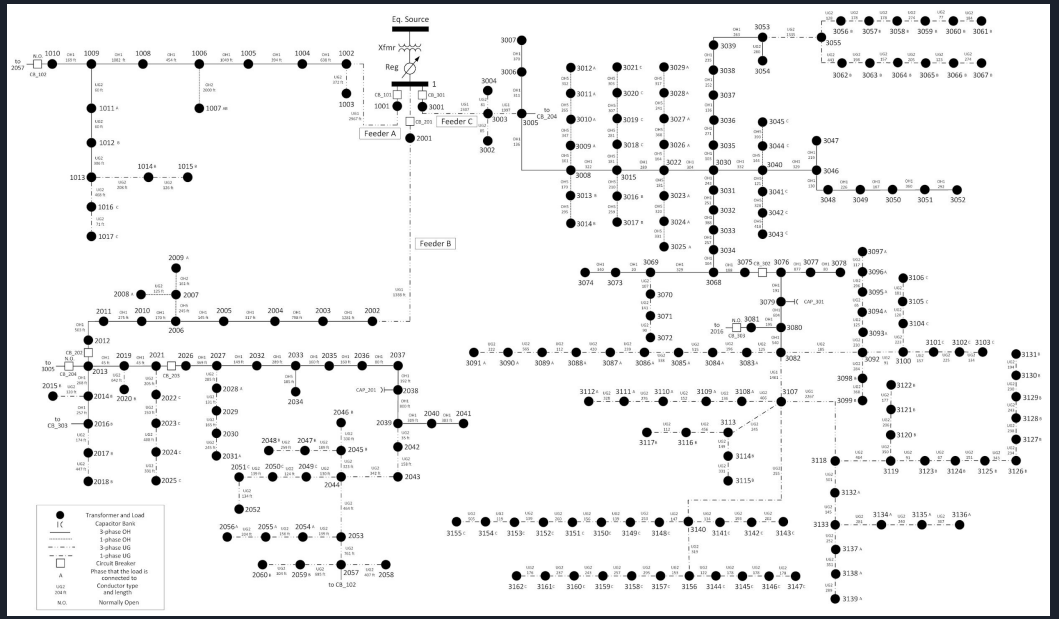
sdmay21-23:

Team Members

Abir Mojumder
Karthik Prakash
Justin Merkel
Patrick Wenzel
Abhilash Tripathy

Project Context

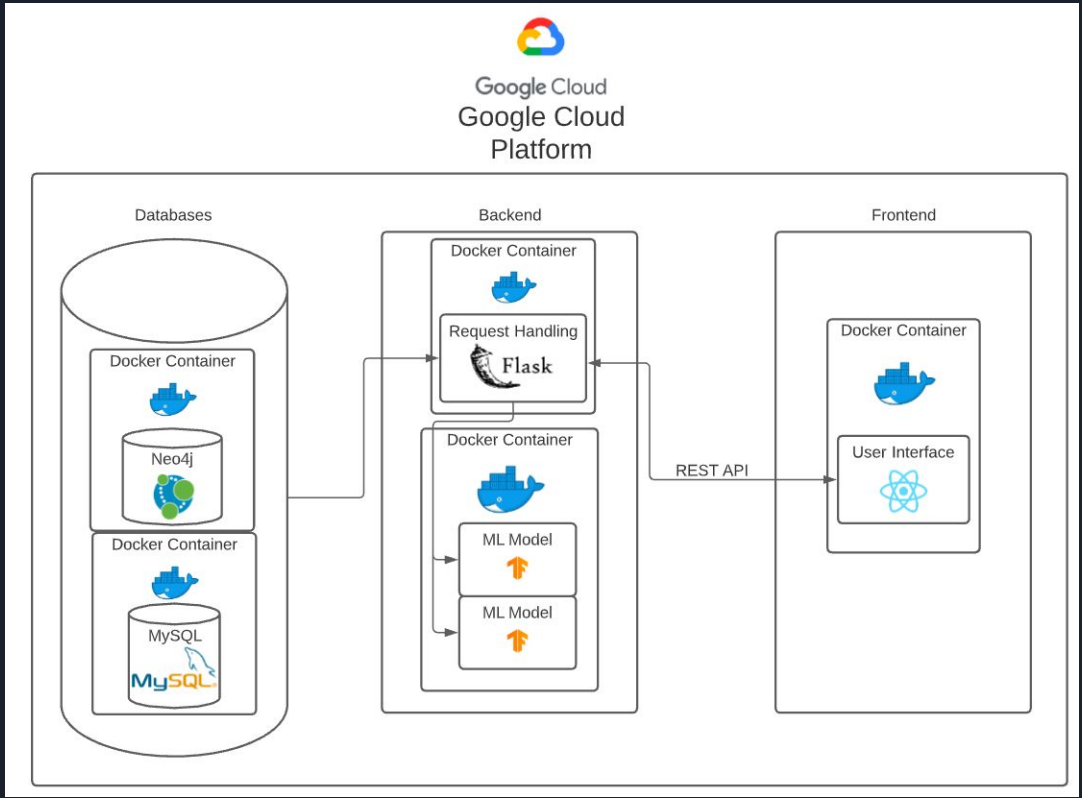
- Use Machine Learning on a simulated power grid to provide analytics and anomaly detection
  - Every node has some power output data associated
  - Static electrical properties
  - Location and connections in network



Goals

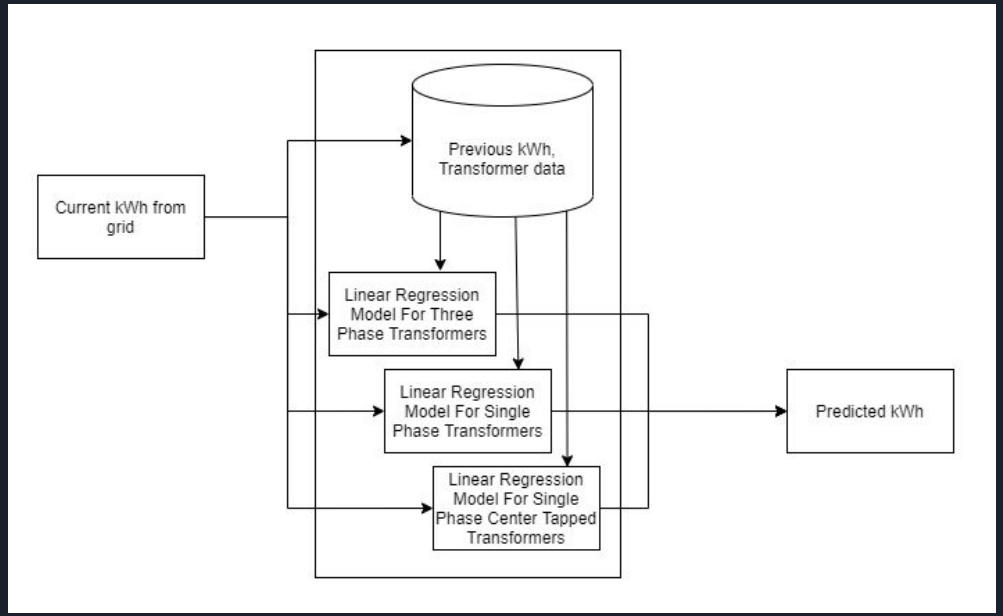
- Frontend:
  - Display the 240-node grid through the React webapp.
  - Provide a good UI for displaying predictions and anomalies.
- Machine Learning:
  - Predict the future kWh output value for nodes
  - Present the chance of a kWh anomaly
- Backend:
  - Provide real-time feedback and data processing
  - Represent grid data as graph

High-level Design



ML Design: Linear Regression

- Implement 3 prediction models (one for each transformer type).
  - Linear Regression is used to predict a continuous value
  - Adding DNN layers to effectively map feature space to higher dimension
  - Decided on MAE as our loss function to limit the effect of outliers
- Feature set currently includes static transformer information, timestamp, the previous bus in the chain, and past kWh output.



ML Design: Logistic Regression

- A continuous value does not tell us if there's an anomaly
- 3 classes of data
  - No Anomaly
  - Power Spike
  - Power Failure
- Softmax for K = 3
  - $$\sigma(\mathbf{z})_i = \frac{e^{-\beta z_i}}{\sum_{j=1}^{K} e^{-\beta z_j}} \text{ for } i = 1, \ldots, K.$$
  - Returns 3 values summing to 1
  - Probabilities for each Anomaly Class

```
1  ,kVA rating,%R1,%R2,%R3,%X12,%X13,%X23,Year,Month,Day,Hour,Current Value,Prev Node,Prev Time,Anomaly
2  0,7.9677,0.665,1.33,1.33,2.256,2.256,1.504,2017,1,1,1,3.125,17.081,0.0,0
3  1,7.9677,0.665,1.33,1.33,2.256,2.256,1.504,2017,1,1,2,2.758,12.786,3.125,0
4  2,7.9677,0.665,1.33,1.33,2.256,2.256,1.504,2017,1,1,3,0.0,10.209,2.758,1
```

Anomaly 1 = Power Failure

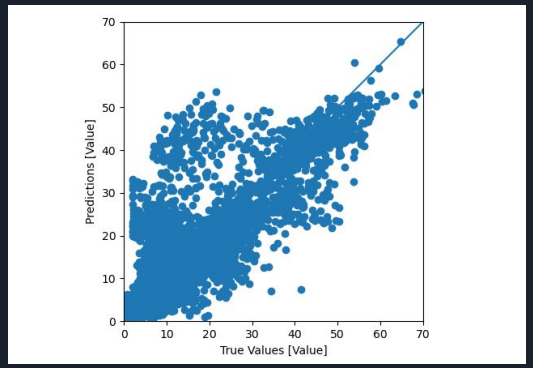
Neural Net layers and softmax final layer

```
[[7.2279609e-06 9.9999273e-01 2.7352313e-12]]
```

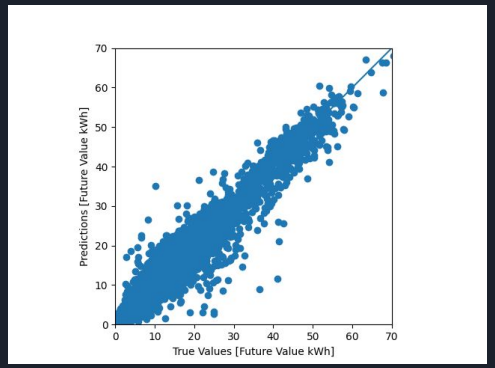
[Normal, Power Failure, Power Spike]

Machine Learning Challenges

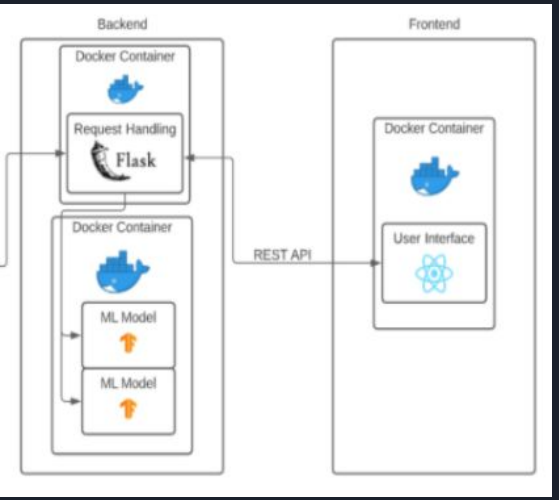
- Understanding how ML works
  - Unaware of common algorithms such as linear regression/logistic regression
  - How do we go from a spreadsheet of node data and node information into an input for a ML model to predict data/detect anomalies
- How to improve the predictability of the models
  - Originally using a feature set of static transformer data and timestamp
  - Implementing depth and temporal information
  - Tuning model parameters

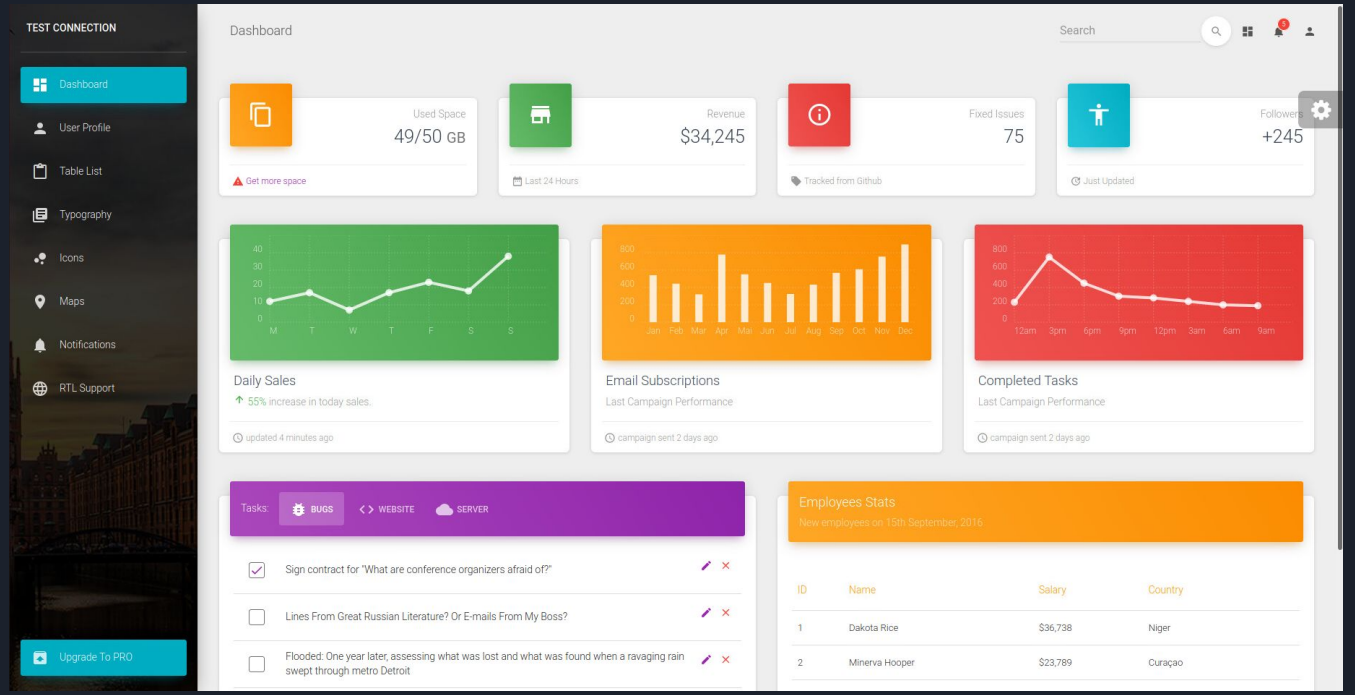


Optimizations



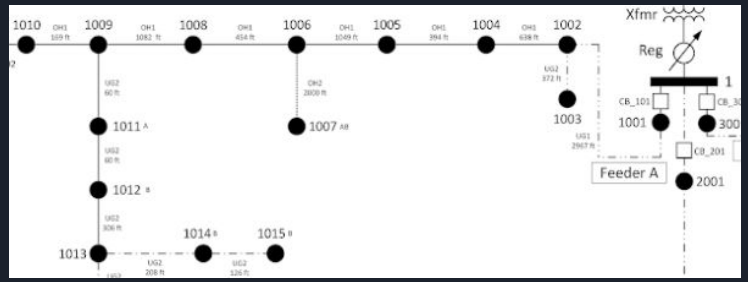
Frontend Design

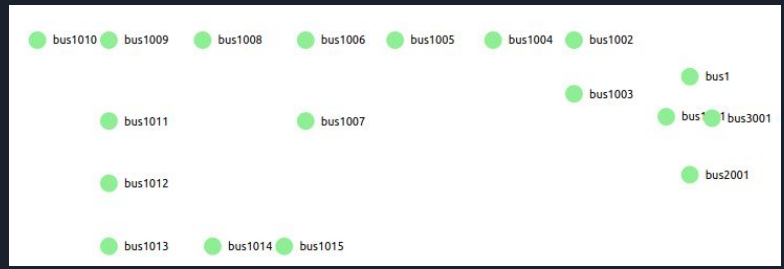




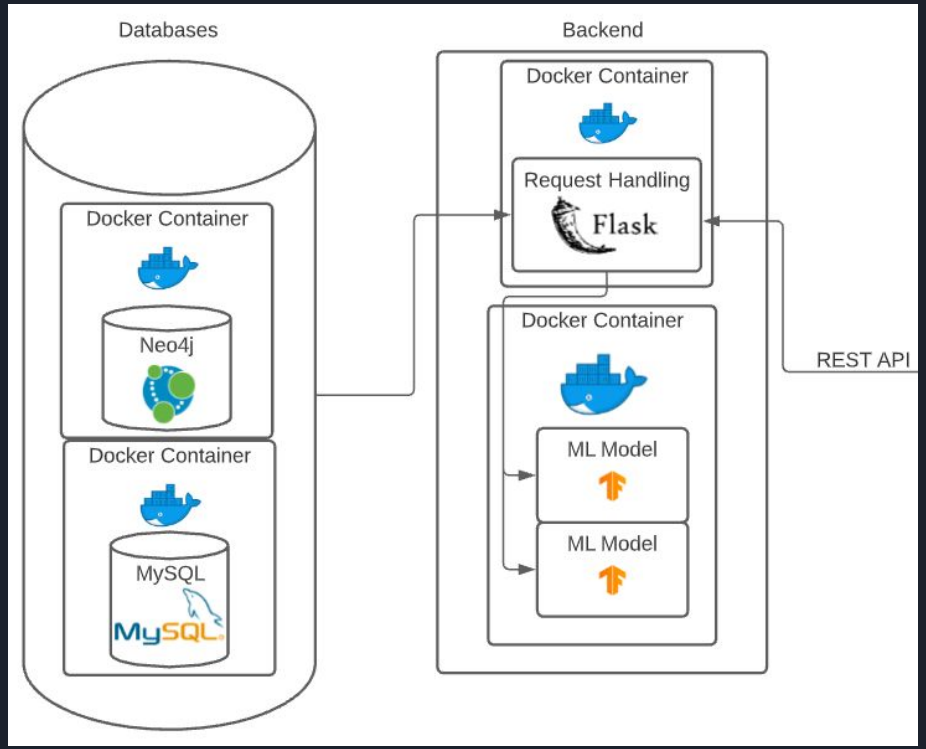
Frontend Challenges

- Dockerizing the frontend
  - Getting our Flask backend routing in a Docker container
  - Getting our ReactJS frontend dashboard in a Docker container
- Using React D3 Component for graphing
  - Using the correct/required configurations (static or loose graph, etc)
  - Making line connections between Bus Nodes from data retrieved from database.
  - Different line connection types (dotted/solid/hybrid) based on phase config.
- Real time updating components
  - Graph and table information update interval (modifying d3 node data/other tabular data at the same time)





Backend Design



Backend Challenges

- REST API
  - Updating database with time-series data
- Database
  - Importing data
  - Learning new query language

Three-phase Secondary Distribution Transformers

Name	Connection	Bus of Winding 1	Bus of Winding 2	Primary voltage rating (kV)	Secondary voltage rating (kV)	kVA rating (kVA)	%R	%X
T_1003	Grd. Y - Grd. Y	bus1003	T_bus1003_L	13.8	0.208	112.5	2.43	3.87
T_1004	Grd. Y - Grd. Y	bus1004	T_bus1004_L	13.8	0.208	75	2.27	1.91
T_1005	Grd. Y - Grd. Y	bus1005	T_bus1005_L	13.8	0.208	75	2.27	1.91
T_1008	Grd. Y - Grd. Y	bus1008	T_bus1008_L	13.8	0.208	45	2.52	1.73
T_1009	Grd. Y - Grd. Y	bus1009	T_bus1009_L	13.8	0.208	75	2.27	1.91
T_1010	Grd. Y - Grd. Y	bus1010	T_bus1010_L	13.8	0.208	45	2.52	1.73
T_1013	Grd. Y - Grd. Y	bus1013	T_bus1013_L	13.8	0.208	45	2.52	1.73
T_2002	Grd. Y - Grd. Y	bus2002	T_bus2002_L	13.8	0.208	300	1.8	4.5
T_2003	Grd. Y - Grd. Y	bus2003	T_bus2003_L	13.8	0.208	75	2.27	1.91
T_2005	Grd. Y - Grd. Y	bus2005	T_bus2005_L	13.8	0.208	45	2.52	1.73

Line Segment Data (Feeder A)

Name	Bus A	Bus B	Length(ft.)	Phases	Config.
L_1001_1002	1001	1002	2967	ABC	UG_3p_type1
L_1002_1003	1002	1003	372	ABC	UG_3p_type2
L_1002_1004	1002	1004	638	ABC	OH_3p_type1
L_1004_1005	1004	1005	394	ABC	OH_3p_type1
L_1005_1006	1005	1006	1049	ABC	OH_3p_type1
L_1006_1007	1006	1007	2000	AB	OH_2p_type2
L_1006_1008	1006	1008	454	ABC	OH_3p_type1

Time \ Bus Name — Energy Consumtpion (kWh)	Bus 1001	Bus 1002	Bus 1003	Bus 1004	Bus 1005	Bus 1006	Bus 1007	Bus 1008	Bus 1009	Bus 1010	Bus 1011	Bus 1012	Bus 1013	Bus 1014	Bus 1015	Bus 1016	Bus 1017
1/1/17 1:00 AM	0	0	15.29	6.892	4.916	5.04	4.163	14.096	17.081	7.136	3.125	3.083	1.537	0.458	0.793	3.083	2.131
1/1/17 2:00 AM	0	0	14.901	6.672	5.335	4.76	3.07	14.937	12.786	7.078	2.758	2.032	2.378	0.336	0.836	2.032	1.914
1/1/17 3:00 AM	0	0	15.772	7.013	4.563	5.04	3.507	14.789	10.209	5.991	3.096	1.597	1.615	0.326	0.803	1.597	2.446
1/1/17 4:00 AM	0	0	15.757	6.452	4.782	4.8	3.143	14.761	10.04	7.03	3.317	1.228	1.536	0.422	0.778	1.228	6.041
1/1/17 5:00 AM	0	0	15.292	6.356	4.482	5	3.147	15.156	10.147	6.043	2.832	1.162	1.773	0.404	0.874	1.162	4.88
1/1/17 6:00 AM	0	0	15.814	6.861	4.963	4.36	3.336	11.145	9.678	6.075	4.433	1.798	1.646	0.35	0.817	1.798	3.656
1/1/17 7:00 AM	0	0	16.044	8.422	4.7	5.24	3.32	9.623	9.327	5.706	4.331	1.05	1.588	0.415	0.854	1.05	2.005
1/1/17 8:00 AM	0	0	15.337	8.201	4.664	4.12	3.572	9.393	9.53	5.809	7.248	1.67	1.715	0.828	1.01	1.67	2.381

Rest API

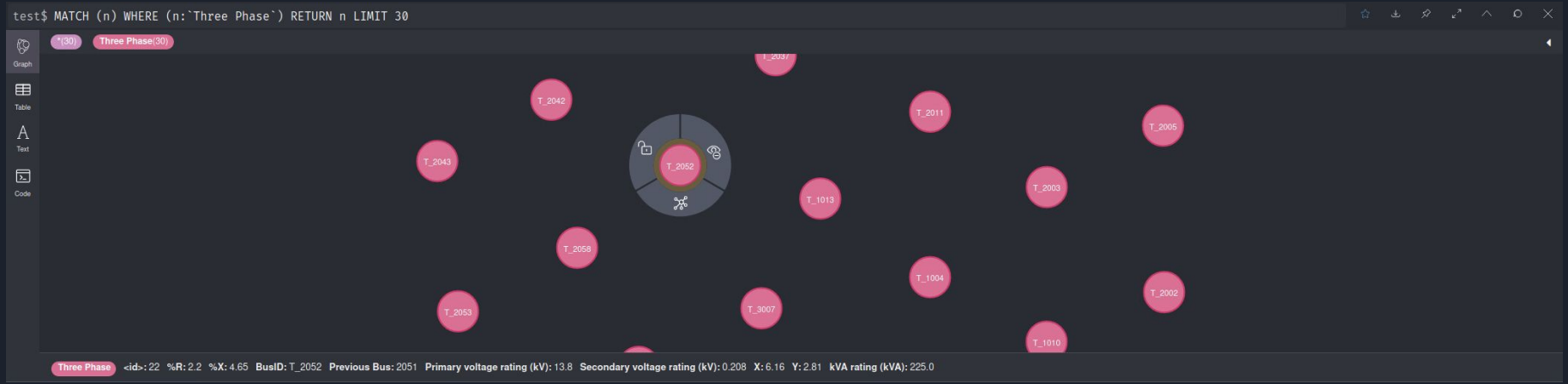
- Request Handling
  - Flask
    - Lightweight
  - JSON data structure
- Challenges
  - Connecting to database
  - Using Cypher Query Language to pull relevant data

```
from neo4j import GraphDatabase
import csv

with open("creds.txt") as file1:
    data=csv.reader(file1,delimiter=",")
    for row in data:
        username=row[0]
        passw=row[1]
        uri=row[2]

driver=GraphDatabase.driver(uri=uri,auth=(username,passw))
session=driver.session()
```

```
@api.route("/create/<string:bus>&<string:value>",methods=["GET","POST"])
def create_node(bus,value):
    query="""
    merge (n:Node{BUS:$bus,VALUE:$value})
    """

    map={"bus":bus,"value":value}
    try:
        session.run(query,map)
        return(f"Node created with bus={bus},value={value}")
    except Exception as e:
        return(str(e))

#def create_node_multi(id,bus):

@api.route("/showall",methods=["GET","POST"])
def return_nodes():
    q2="MATCH (n) return n.BUS as Node ,n.VALUE as KWH"
    output=session.run(q2)
    return(jsonify(output.data()))
```

Database

- Store node parameters and time-series data
- Challenges
  - Neo4j
    - Docker folder permissions
    - Importing data
    - Formatting node data into a reasonable graph structure
    - Learning Cypher Query Language



Questions?