EE/CprE/SE 492 WEEKLY REPORT 5

Nov 1st - Nov 14th

Group number: **20**

Project title: **GridGPT**

Client &/Advisor: **Gelli Ravikumar**

Team Members/Role:

- **Tin Ngo -> AI Integration Specialist/Backend Lead**
- **Jackson Phillips -> AI Integration Specialist**
- **Emma Heithoff -> Power Systems Specialist**
- **Eddy Andrade -> Frontend Lead**
- **Nick Doty -> Power Systems Specialist**

- **Weekly Summary**

We discussed submitting for the Texas Energy and Power Conference in our first team meeting this period. Our next team meeting consisted of improvements to be made during development this upcoming week. Improvements mentioned were branching and merging code development, presenting our contributions to the overall design, and clarifying questions on the roadblocks each of the three project categories came across.

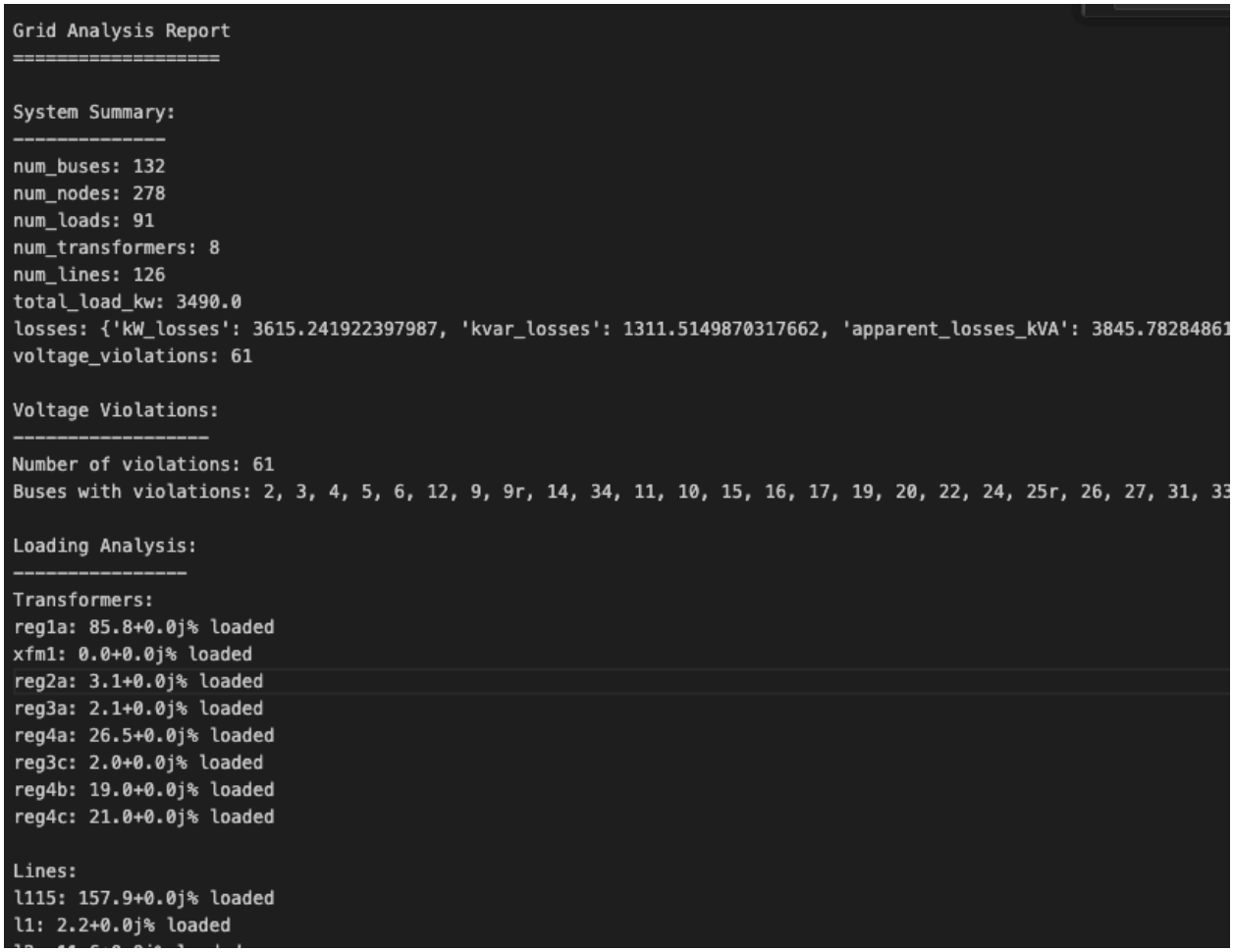
- **Past week accomplishments**
  - Tin Ngo: While working with Eddy Andrade-Robles, we found that my implementation was not what was expected. Initially I designed it to take the directory of a file but after collaboration, we needed to make something new that would handle a specific file only. I created that functionality where the user can specify the specific file and give it a prompt.

```
(venv) tinngo@tinngo sddec24-20 % curl -X POST http://0.0.0.0:3004/openai_generate_text_single_dss_file \
-H "Content-Type: application/json" \
-d '{
  "prompt": "Is Load S1A a single phase or three phase load?",
  "model": "gpt-3.5-turbo-0125",
  "context_directory": "grid_applications/distribution_grid/grid_dss/src/systems/123Bus",
  "file_name": "IEEE123Loads.dss"
}'
{
    "response": "Based on the extracted information, Load S1A is a single-phase load."
}
```

While working with Eddy Andrade-Robles to work on a End to End test to modify a file similarly to GitHub Copilot. I developed a endpoint and I leveraged prompt engineering to rewrite the file with the removed/additions. This worked and was shown during our first proof-of-concept demo.

Week 2

I worked with Emma and Nick to develop more altDss scripts to run power analysis simulations. A lot of the time was spent understanding what the expected outputs were. We ended up modifying multiple dss files and comparing simulations before and after the changes and we were able to see valuable differences that we're going to take action on. I worked further to create a grid analyzer class that uses more functionality from the altDSS library. Here is what an output of a run looks like:



- Jackson Phillips: I have been working on more hugging face functionality this week. I implemented some of tin's work on the openai functionality such as the text chunking and file access. Most of my time was spent trying to get good responses out of the Llama 3.2 1B model. I added some different configs in the generate_text function but I am still getting repetitive responses.

```python
def generate_text(self, prompt, max_length=200, temperature=0.7, top_k=50, top_p=0.9, repetition_penalty=1.15):
    """
    Generate text based on the given prompt, with added controls to reduce repetition.

    Args:
        prompt (str): The input text prompt.
        max_length (int): The maximum length of the generated text.
        temperature (float): Controls randomness in predictions (default 0.7).
        top_k (int): Limits sampling to the top-k tokens (default 50).
        top_p (float): Nucleus sampling, taking top tokens with cumulative probability up to
        top_p (default 0.9). Returns: str: The generated text.
        repetition_penalty (float): Discourages repetition (defailt 1.15)"""
    # Tokenize input text
    inputs = self.tokenizer(prompt, return_tensors="pt", return_attention_mask=False)
    inputs = {k: v.to(self.device) for k, v in inputs.items()} # Move input tensors to the device

    # Generate output with specified generation parameters
    outputs = self.model.generate(
        **inputs,
        max_length=max_length,
        pad_token_id=self.tokenizer.pad_token_id,
        temperature=temperature,
        top_k=top_k,
        top_p=top_p
    )
    text = self.tokenizer.decode(outputs[0], skip_special_tokens=True)
    return text
```

- Emma Heithoff: I clarified my new and realistic milestones in an email to the team and Professor Gelli. Tin, Nick, and I met to begin coding the concepts I have been researching for simulating datasets where grid stabilization outcomes are mapped to input changes in the grid. I explained the purpose of the code we are generating line by line as Tin used code that would run successfully. I use my concept of the dataset to have a conversation with ChatGPT on a vague idea for scripting my design. It helped me visualize the planning I had done before taking a break from school for health. I have included my conceptual planning from the last week as the first section of code. Next, the generated code is commented with the documentation fixes I will implement with Nick and Tin.

  We expected the generated code to not reflect the functionality or syntax of AltDSS. Tin manually changed the dss file parameter values, the loads specifically, and we used a troubleshooted version of the code to utilize AltDSS analysis functionality. I reviewed the AltDSS repository documentation for the syntax to run the commands that the generated code had the right concept on but could not write syntax for successfully. I found a specific load multiplier command in the documentation that should now work. I reviewed that AltDSS is a package in the larger project of DSS-Python. DSS-Python is the backend that AltDSS connects to, which helped me understand how we are coding.

```python
#my conceptual planning from last weekly report

import altdss as altdss

# Redirect to the DSS file
altdss.altdss(f'redirect "grid_applications/distribution_grid/grid_dss/src/systems/123Bus/Run_IEEE123Bus.DSS"')

# Solve the circuit
altdss.altdss.Solution.Solve()

# Get bus voltages
voltages = altdss.altdss.BusVolts()

# Analyze voltage stability
min_voltage = 0.95  # Minimum acceptable voltage (in p.u.)
for i, voltage in enumerate(voltages):
    print(f"Bus {i+1}: Voltage = {voltage:.3f} V")
    if voltage < min_voltage:
        print(f"Warning: Bus {i+1} has a low voltage: {voltage:.3f} V")


#Generated code with future fixes commented
import json
import random
import altdss

# Initialize AltDSS and load the 13-bus system
dss = altdss.DSS()
dss.text("clear")
dss.text("redirect path/to/13bus.dss")  # Load your 13-bus system file
```

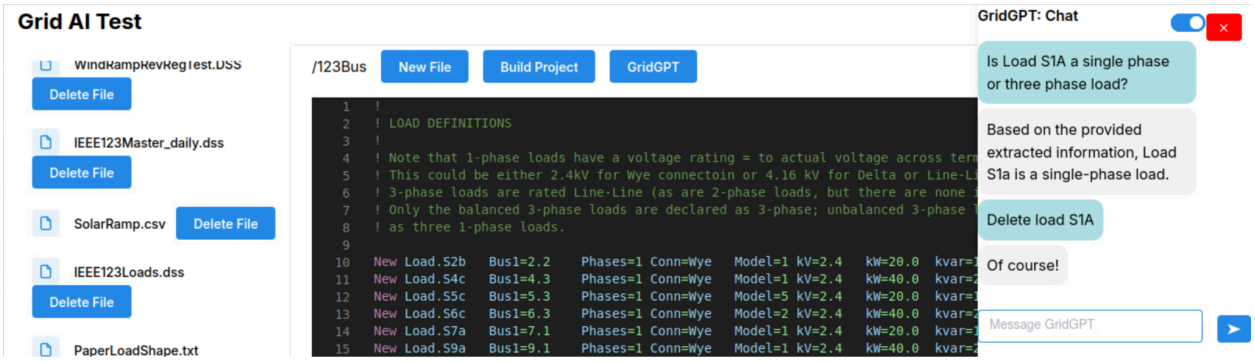
```python
# Function to simulate a single scenario and produce an input-output pair
def simulate_13bus_scenario(bus, load_increase_factor=1.0):
    dss.text(f"Set LoadMultiplier={load_increase_factor}")
    #documentation says syntax is
    #we will import SolveModes instead of using .text so the line is altdss.Solution.Solve()
    #and the prefix needs to be altdss not dss
    #we will use altdss.Solution.LoadMult = 1.3 and altdss2.Solution.Solve()
    #we use altdss2=altdss.NewContext() because the grid model is changing load values

    dss.text("Solve")

    bus_voltage = dss.circuit.buses(bus).pu_voltage()
    initial_voltage_pu = bus_voltage[0] if bus_voltage else None
    initial_reactive_power_kvar = dss.circuit.total_power().imag  # total reactive power


    if initial_voltage_pu is None:
        return None

    min_voltage_pu = min(bus_voltage)
    capacitor_activation = min_voltage_pu < 0.9  # Example threshold

    added_reactive_support_kvar = 150 if capacitor_activation else 0
    if capacitor_activation:
        dss.text(f"Edit Capacitor.bank phases=3 kvar={added_reactive_support_kvar}")
        dss.text("Solve")

    stabilized_voltage_pu = dss.circuit.buses(bus).pu_voltage()[0] if dss.circuit.buses(bus).pu_voltage() else min_voltage_pu
    time_to_stabilize_s = 15 if capacitor_activation else 5
    outcome = "voltage_stabilized" if capacitor_activation else "voltage_stable_without_action"
```

```
    return {
        "input": {
            "event_type": "reactive_power_imbalance",
            "bus": bus,
            "initial_load_kw": dss.circuit.total_power().real,
            "initial_load_kvar": initial_reactive_power_kvar,
            "initial_voltage_pu": initial_voltage_pu,
            "initial_reactive_power_support_kvar": initial_reactive_power_kvar
        },
        "output": {
            "voltage_response": {
                "min_voltage_pu": round(min_voltage_pu, 2),
                "stabilized_voltage_pu": round(stabilized_voltage_pu, 2),
                "time_to_stabilize_s": time_to_stabilize_s
            },
            "control_actions": {
                "capacitor_activation": capacitor_activation,
                "added_reactive_power_support_kvar": added_reactive_support_kvar
            },
            "outcome": outcome
        }
    }

def create_dataset(num_scenarios):
    dataset = []
    for _ in range(num_scenarios):
        bus = f"Bus{random.randint(1, 13)}"
        load_increase_factor = random.uniform(0.8, 1.5)  # Random load factor
        pair = simulate_13bus_scenario(bus, load_increase_factor)
        if pair:  # Only append valid pairs
            dataset.append(pair)

    return dataset
```

```
num_scenarios = 50
dataset = create_dataset(num_scenarios)

with open("voltage_instability_dataset_13bus.json", "w") as f:
    json.dump(dataset, f, indent=4)

print("Dataset created and saved to voltage_instability_dataset_13bus.json")
```

- Eddy Andrade: I continued to work closely with Tin Ngo on making API calls with OpenAI. We noticed that what he had originally implemented wouldn't take into consideration individual files, and instead focused on directories. We worked on implementing a new call to view individual files within a directory. After doing so, we got a response from the AI model. We began doing some simple tests by also asking Emma Heithoff and Nicholas Doty what questions to ask in the chat

(and what the correct responses are). Finally, me and Tin continued working on making changes to the file from the AI. For the time being, the AI only deletes certain things if you ask what you want to delete.

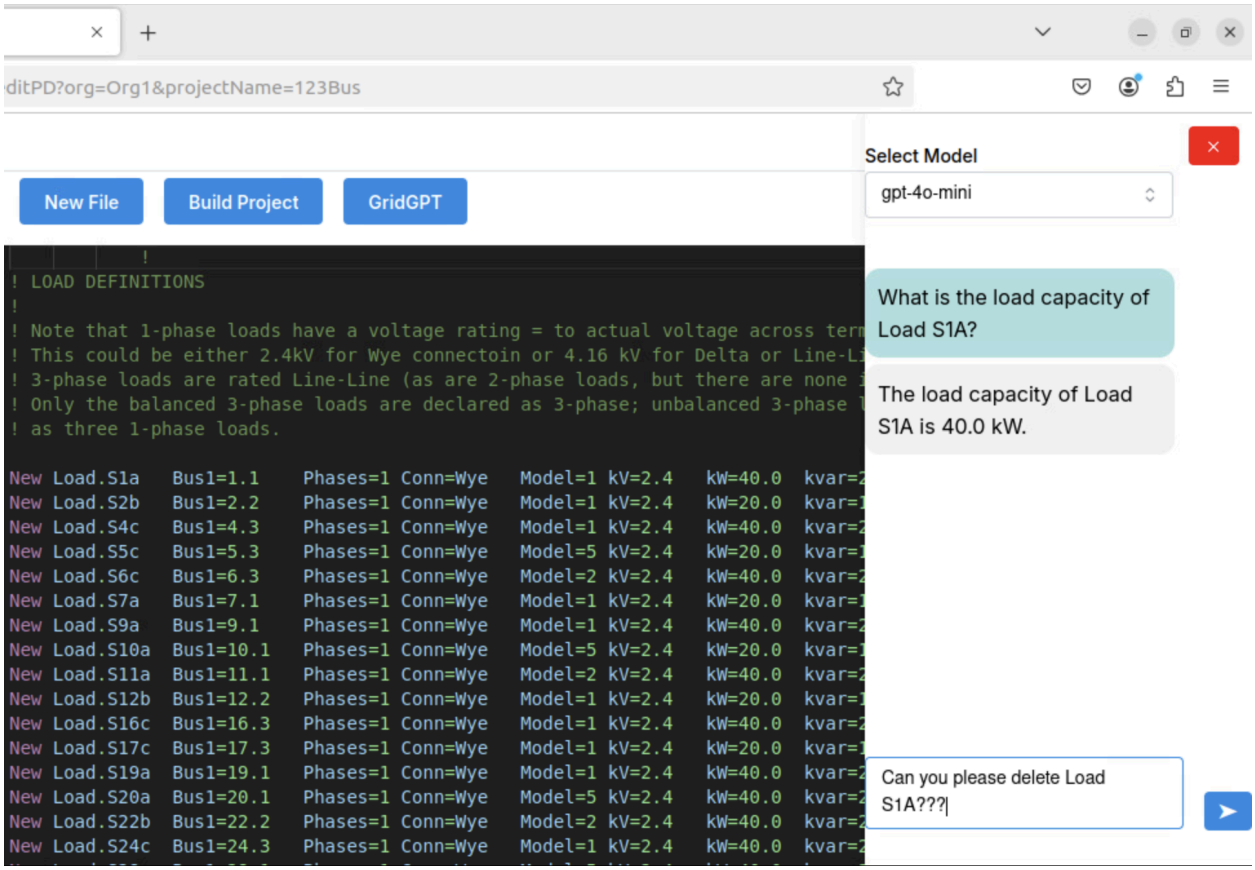


The first message asks the AI if Load S1A is a single-phase or three-phase load, to which it views the file and gives a response. Finally, I asked it to delete Load S1A. It did take some time for the AI to do so, but it eventually updated the entire file with the changes.

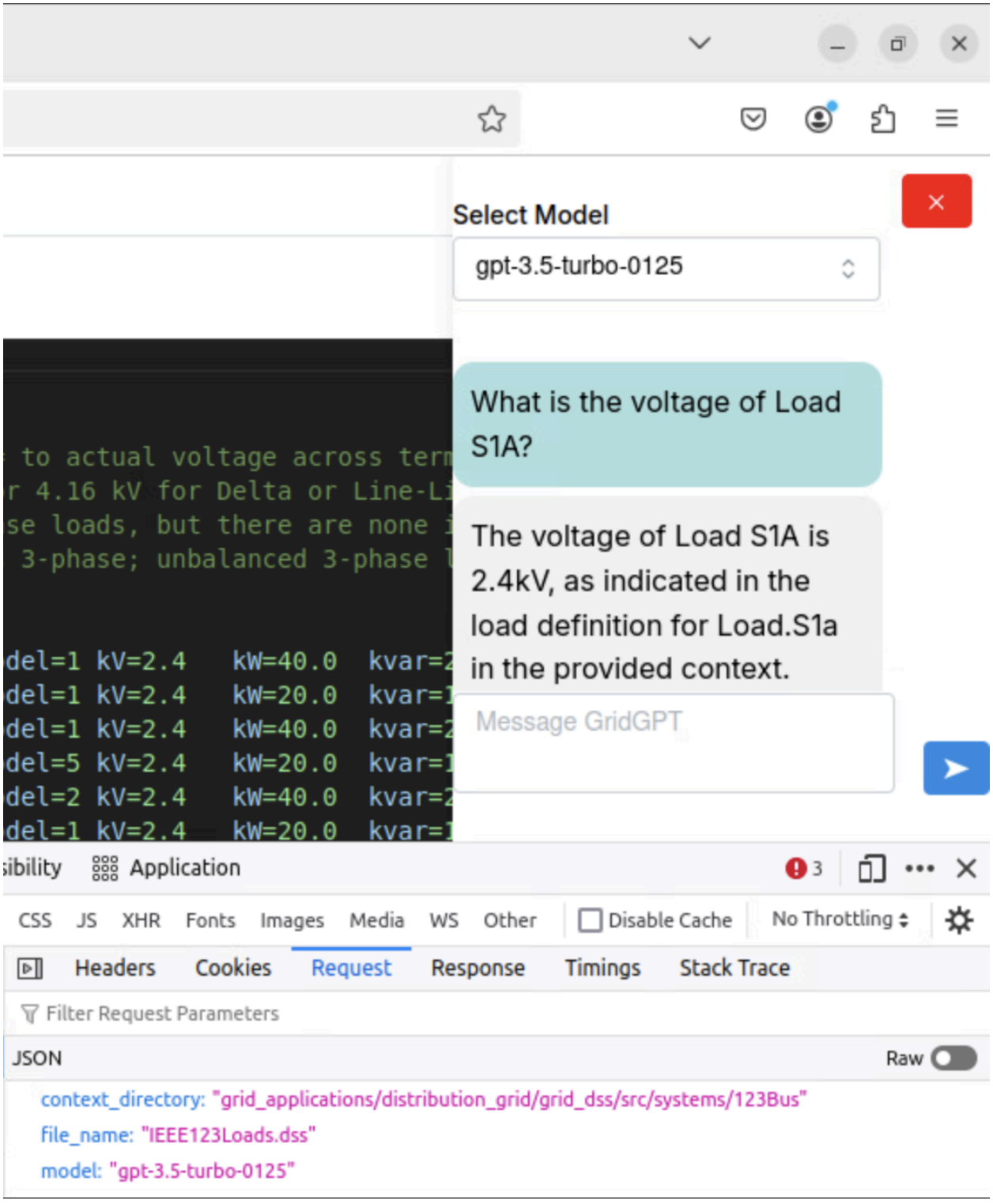
Week 2

- Haven't got much work done due to being under the weather for a good chunk of the week, but I decided to take a look at the Python script used to make the API calls. In there, I made changes to dynamically call on all the endpoints based on the context of the prompts provided by the user. I created a new endpoint to run through the prompt, and depending on the context of it, make a call to different endpoints.

  Finally, I made changes to the layout of the chat window. The user can now select which OpenAI model to use (the default is gpt-4o-mini) if desired. As for the chat bubbles, they will no longer overlap that model selection dropdown. I also changed the text input into a Textarea to allow the user's message to break up into multiple rows instead of one single line.



Updated layout (request made with gpt-4o-mini model)



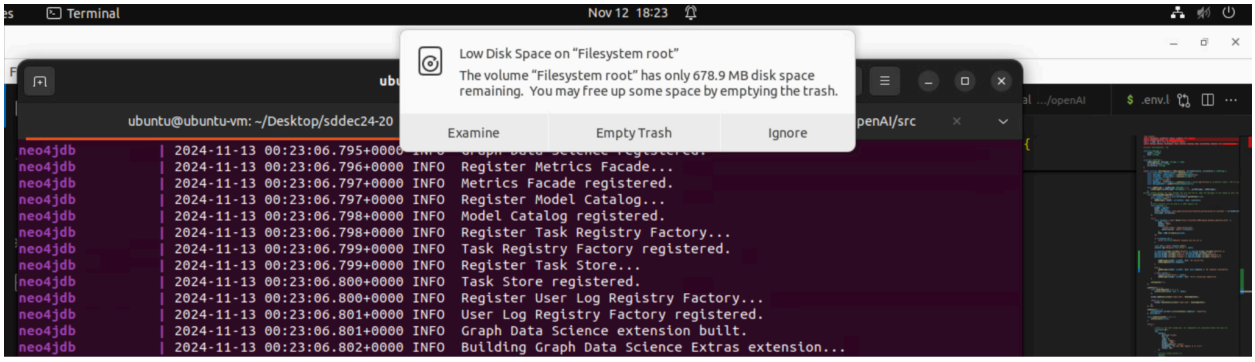
Screenshot shows that changing the OpenAI model works with each request you make

- Nick Doty: This week I worked with Tin and Emma to begin to generate an altDSS script that will run power analysis simulations with the help of the altDSS python documentaion. Additionally, when fully functional, it will change the values in a csv file using a load multiplier. For the script we used the IEEE123bus.

```
import json
from altdss import altdss

IEEE13_PATH = "../grid_applications/distribution_grid/grid_dss/src/systems/123Bus/Run_IEEE123Bus.DSS"

altdss(f'''
    Clear
    Redirect "{IEEE13_PATH}"
''')

# Run Simulation Solve

altdss.Solution.Solve()

load_objects = altdss.Load()

print(load_objects)



# Apply changes to the system
altdss.Solution.Solve()



# apply action if needed - stabalize voltage - apply capacity control

# Run another simulation to see how the load changes the voltages
```

- **<u>Pending issues</u>**
  - Tin Ngo: N/A
  - Jackson Phillips: I am confused on how to stop the model from giving repetitive responses. I also do not have a dataset yet so no progress has been made on training.
  - Emma Heithoff: Not yet successfully creating data for model training when my timeline has been pushed back is not ideal to support the rest of the team.

- Eddy Andrade: When working on the project, I cam across this message on the VM about having low disk space. I am not sure if it is something major but thought it is worth bringing up



- Nick Doty: N/A

○ Individual contributions

NAME	**Individual Contributions** *(Quick list of contributions. This should be short.)*	**Hours this week**	**HOURS cumulative**
Tin Ngo	Front End POC. AltDss Simulations	20	177
Jackson Phillips	Huggingface functionality	6	121
Emma Heithoff	Explained plan for simulations and troubleshooting	7	121
Eddy Andrade	Frontend work, Backend	13	124
Nick Doty	Dataset work	6	123

- **Plans for the upcoming week**
    - Tin Ngo: Work more on altDSS simulations.
    - Jackson Phillips: Work on huggingface functionality if I can't get hands on with a dataset for training. If I can get a dataset I will work on training.
    - Emma Heithoff: Overall goal is to be efficient with the dataset generation. This requires working individually and together on finding AltDSS commands from the documentation that reflect the conversations we've had on making event and outcome (input/output) pairs for teaching AI to understand processes of grid stabilization. Reflecting on what has been done in 491/492 so far and what work gaps there are simultaneously.
    - Eddy Andrade: I plan on making tabs that keeps a history of previous messages with GridGPT. I'll keep finding ways to refine the Frontend aside from that
    - Nick Doty: Continue to work on the script with Emma and Tin along with anything else that needs to get done.