AI-VVO sdmay22-36 Weekly Update #7

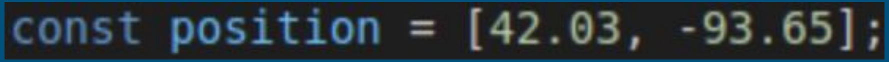
11/8/2021 - 11/15/2021

Front-end (This Week)

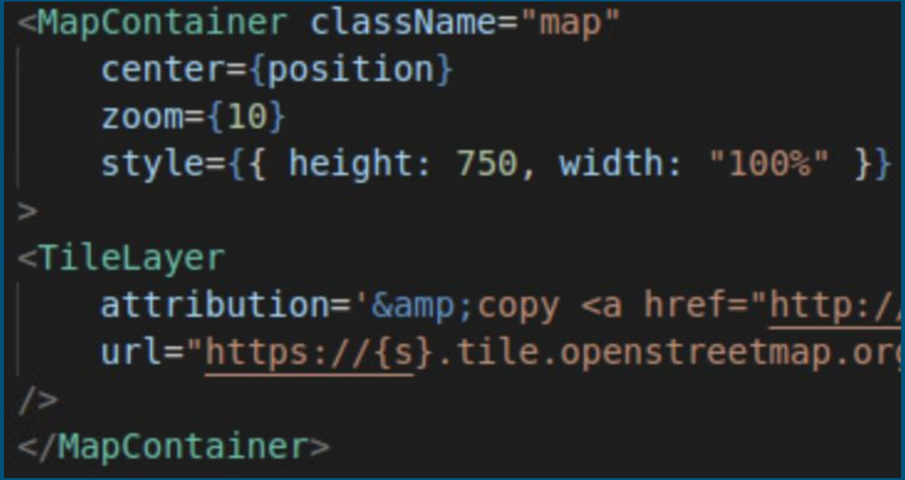
- Completed more react-leaflet tutorials
- Began implementing map features into project in our VMs
- Researched markers and custom icons
- Installed react-leaflet and leaflet into the project
- Researched using a json data file for getting node data to map

Front-end (This week)

- Implemented a Map.js file to output the map on the application
- Will eventually implement the Map.js file into the GridVis.js file for less confusion



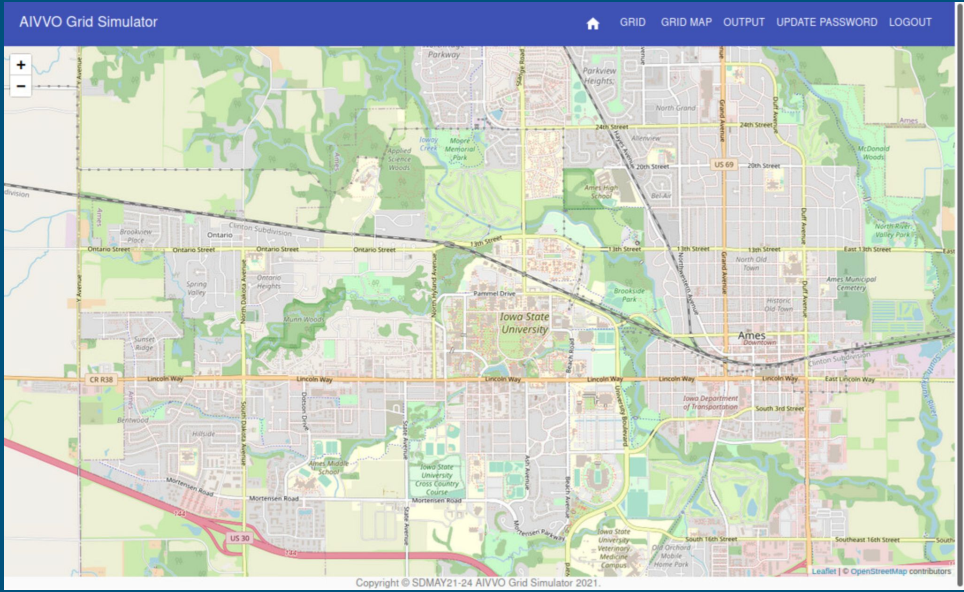
```
const position = [42.03, -93.65];
```



```
<MapContainer className="map"
    center={position}
    zoom={10}
    style={{ height: 750, width: "100%" }}
>
<TileLayer
    attribution='&amp;copy <a href="http:/
    url="https://{s}.tile.openstreetmap.or
/>
</MapContainer>
```

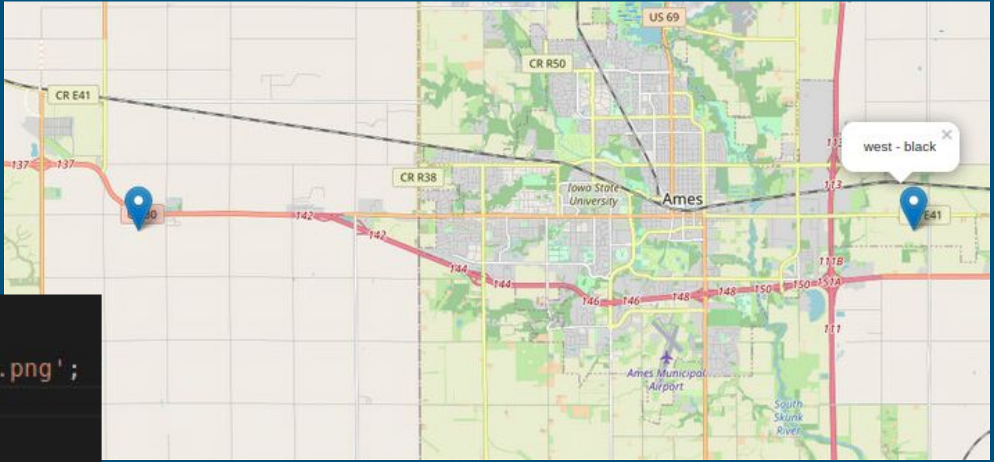
Front-end (This Week)

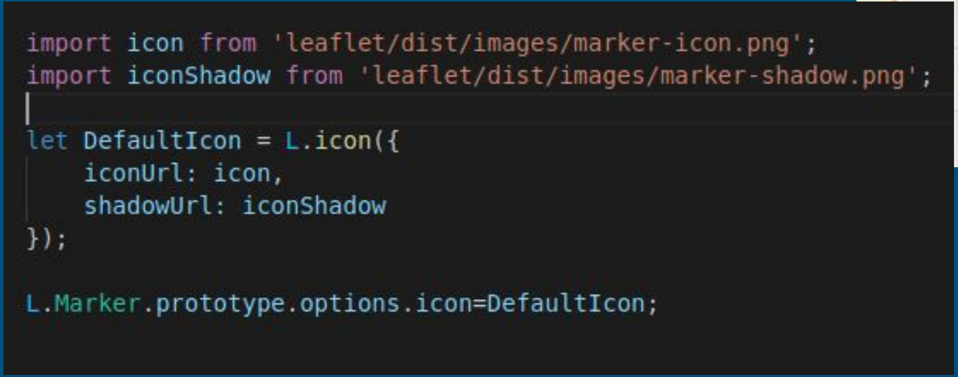
- Added Grid map button to the home screen toolbar (will eventually change it to be in the Grid screen toolbar)
- The button will display the map with its center being at a specified location


AIVVO Grid Simulator
GRID
GRID MAP
OUTPUT
UPDATE PASSWORD
LOGOUT
Iowa State University
Ames
Leaflet | © OpenStreetMap contributors
Copyright © SDMAY21-24 AIVVO Grid Simulator 2021

Front-end (This Week)

- Fixed issue with markers not displaying by importing leaflet marker icons
- Set up a default icon to use



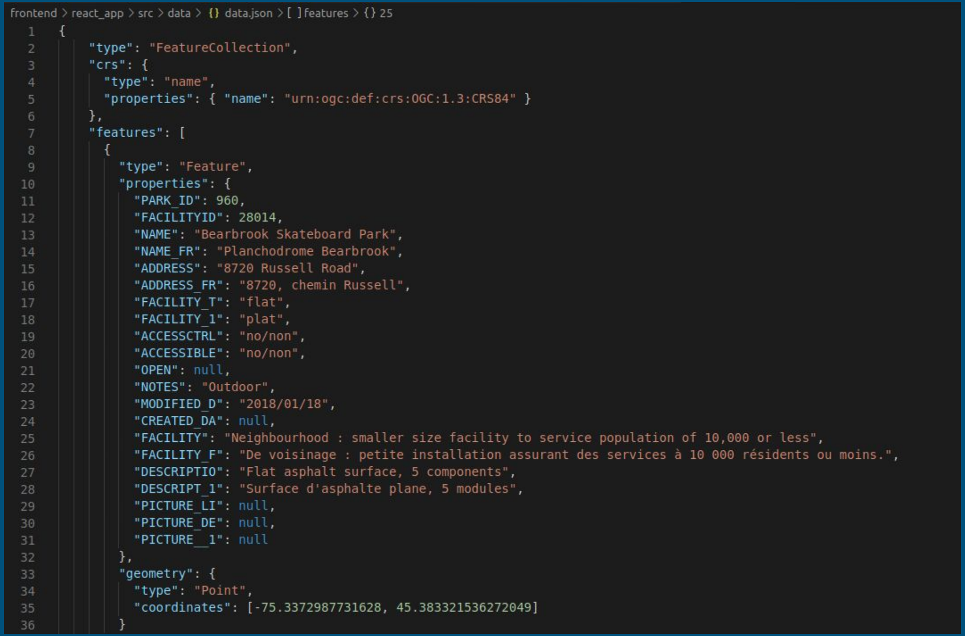


```
import icon from 'leaflet/dist/images/marker-icon.png';
import iconShadow from 'leaflet/dist/images/marker-shadow.png';

let DefaultIcon = L.icon({
    iconUrl: icon,
    shadowUrl: iconShadow
});

L.Marker.prototype.options.icon=DefaultIcon;
```

Front-end (This Week)

- Added a test “data.json” file with data from a tutorial
- JSON file will be filled with node data including:
    - Coordinates
    - Status (online, offline, etc.)
    - Last Date of Repair
    - Etc.

- Updated README with tutorial information and helpful links



frontend > react_app > src > data > {} data.json > [ ] features > {} 25

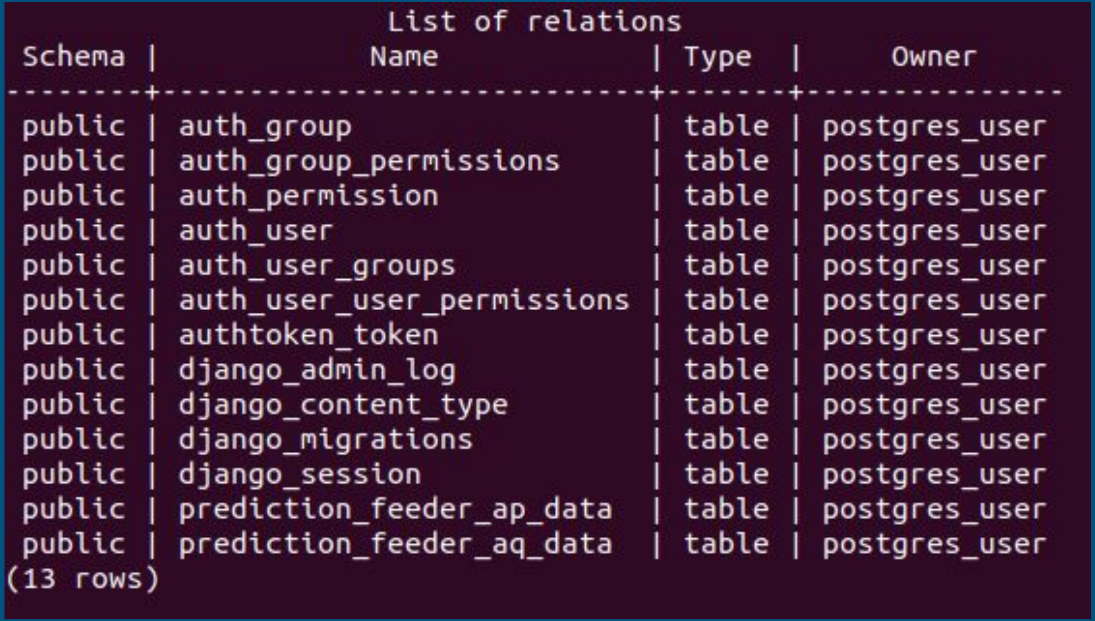
```
{
    "type": "FeatureCollection",
    "crs": {
      "type": "name",
      "properties": { "name": "urn:ogc:def:crs:OGC:1.3:CRS84" }
    },
    "features": [
      {
        "type": "Feature",
        "properties": {
          "PARK_ID": 960,
          "FACILITYID": 28014,
          "NAME": "Bearbrook Skateboard Park",
          "NAME_FR": "Planchodrome Bearbrook",
          "ADDRESS": "8720 Russell Road",
          "ADDRESS_FR": "8720, chemin Russell",
          "FACILITY_T": "flat",
          "FACILITY_1": "plat",
          "ACCESSCTRL": "no/non",
          "ACCESSIBLE": "no/non",
          "OPEN": null,
          "NOTES": "Outdoor",
          "MODIFIED_D": "2018/01/18",
          "CREATED_DA": null,
          "FACILITY": "Neighbourhood : smaller size facility to service population of 10,000 or less",
          "FACILITY_F": "De voisinage : petite installation assurant des services à 10 000 résidents ou moins.",
          "DESCRIPTIO": "Flat asphalt surface, 5 components",
          "DESCRIPT_1": "Surface d'asphalte plane, 5 modules",
          "PICTURE_LI": null,
          "PICTURE_DE": null,
          "PICTURE__1": null
        },
        "geometry": {
          "type": "Point",
          "coordinates": [-75.3372987731628, 45.383321536272049]
        }
```

Front-end (Next Week)

- Put Map and associated libraries in a Docker container
- Populate JSON file with test data based on real nodes
- Place markers according to real node locations on map
- Research how to add route highlighting for connected networks

Back-end (This Week)

- Created two new tables in the database for FeederAP and FeederAQ data

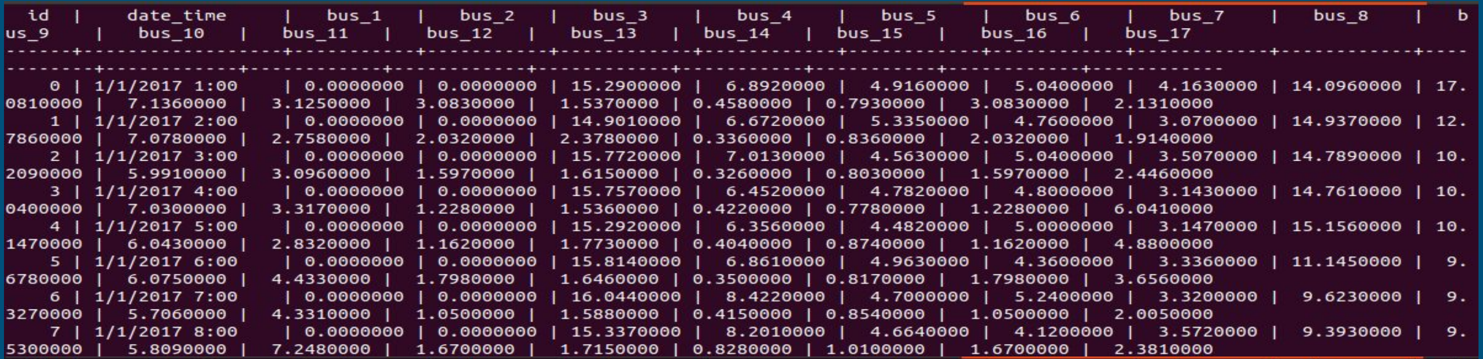


```
                     List of relations
 Schema |             Name            | Type  |     Owner
--------+-----------------------------+-------+---------------
 public | auth_group                  | table | postgres_user
 public | auth_group_permissions      | table | postgres_user
 public | auth_permission             | table | postgres_user
 public | auth_user                   | table | postgres_user
 public | auth_user_groups            | table | postgres_user
 public | auth_user_user_permissions  | table | postgres_user
 public | authtoken_token             | table | postgres_user
 public | django_admin_log            | table | postgres_user
 public | django_content_type         | table | postgres_user
 public | django_migrations           | table | postgres_user
 public | django_session              | table | postgres_user
 public | prediction_feeder_ap_data   | table | postgres_user
 public | prediction_feeder_aq_data   | table | postgres_user
(13 rows)
```

Back-end (This Week)

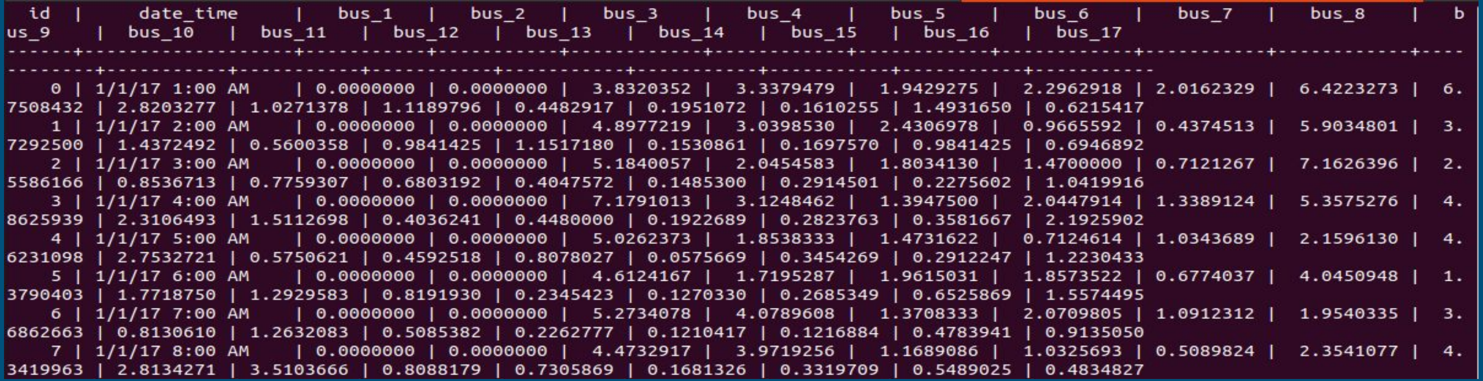
- Filled in the tables with data from csv files given

Ap data



```
  id  |      date_time      |   bus_1   |   bus_2   |   bus_3    |    bus_4    |    bus_5    |    bus_6    |    bus_7    |    bus_8    |   b
us_9     |   bus_10    |   bus_11    |   bus_12    |   bus_13    |  bus_14   |  bus_15   |   bus_16    |   bus_17
------+-------------------+-----------+-----------+------------+-------------+-------------+-------------+-------------+-------------+----
--------+-------------+-------------+-------------+-------------+-----------+-----------+-------------+-------------
     0 | 1/1/2017 1:00     | 0.0000000 | 0.0000000 | 15.2900000 |   6.8920000 |   4.9160000 |   5.0400000 |   4.1630000 | 14.0960000 | 17.
0810000 |   7.1360000 |   3.1250000 |   3.0830000 |   1.5370000 | 0.4580000 | 0.7930000 |   3.0830000 |   2.1310000
     1 | 1/1/2017 2:00     | 0.0000000 | 0.0000000 | 14.9010000 |   6.6720000 |   5.3350000 |   4.7600000 |   3.0700000 | 14.9370000 | 12.
7860000 |   7.0780000 |   2.7580000 |   2.0320000 |   2.3780000 | 0.3360000 | 0.8360000 |   2.0320000 |   1.9140000
     2 | 1/1/2017 3:00     | 0.0000000 | 0.0000000 | 15.7720000 |   7.0130000 |   4.5630000 |   5.0400000 |   3.5070000 | 14.7890000 | 10.
2090000 |   5.9910000 |   3.0960000 |   1.5970000 |   1.6150000 | 0.3260000 | 0.8030000 |   1.5970000 |   2.4460000
     3 | 1/1/2017 4:00     | 0.0000000 | 0.0000000 | 15.7570000 |   6.4520000 |   4.7820000 |   4.8000000 |   3.1430000 | 14.7610000 | 10.
0400000 |   7.0300000 |   3.3170000 |   1.2280000 |   1.5360000 | 0.4220000 | 0.7780000 |   1.2280000 |   6.0410000
     4 | 1/1/2017 5:00     | 0.0000000 | 0.0000000 | 15.2920000 |   6.3560000 |   4.4820000 |   5.0000000 |   3.1470000 | 15.1560000 | 10.
1470000 |   6.0430000 |   2.8320000 |   1.1620000 |   1.7730000 | 0.4040000 | 0.8740000 |   1.1620000 |   4.8800000
     5 | 1/1/2017 6:00     | 0.0000000 | 0.0000000 | 15.8140000 |   6.8610000 |   4.9630000 |   4.3600000 |   3.3360000 | 11.1450000 |  9.
6780000 |   6.0750000 |   4.4330000 |   1.7980000 |   1.6460000 | 0.3500000 | 0.8170000 |   1.7980000 |   3.6560000
     6 | 1/1/2017 7:00     | 0.0000000 | 0.0000000 | 16.0440000 |   8.4220000 |   4.7000000 |   5.2400000 |   3.3200000 |  9.6230000 |  9.
3270000 |   5.7060000 |   4.3310000 |   1.0500000 |   1.5880000 | 0.4150000 | 0.8540000 |   1.0500000 |   2.0050000
     7 | 1/1/2017 8:00     | 0.0000000 | 0.0000000 | 15.3370000 |   8.2010000 |   4.6640000 |   4.1200000 |   3.5720000 |  9.3930000 |  9.
5300000 |   5.8090000 |   7.2480000 |   1.6700000 |   1.7150000 | 0.8280000 | 1.0100000 |   1.6700000 |   2.3810000
```

Aq data

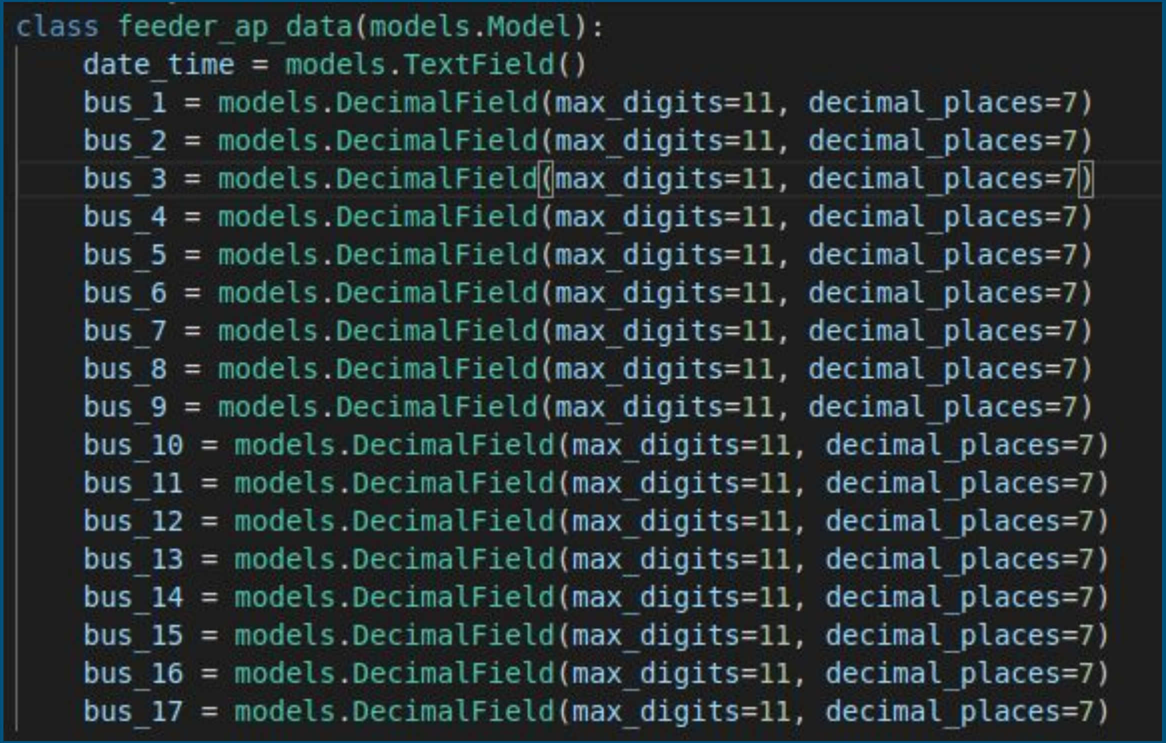


```
  id  |        date_time        |   bus_1   |   bus_2   |   bus_3   |   bus_4   |   bus_5   |   bus_6   |   bus_7   |   bus_8   |   b
us_9     |  bus_10   |  bus_11   |  bus_12   |  bus_13   |  bus_14   |  bus_15   |  bus_16   |  bus_17
------+-----------------------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+----
--------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------
     0 | 1/1/17 1:00 AM        | 0.0000000 | 0.0000000 |  3.8320352 |  3.3379479 |  1.9429275 |  2.2962918 | 2.0162329 |  6.4223273 |  6.
7508432 | 2.8203277 | 1.0271378 | 1.1189796 | 0.4482917 | 0.1951072 | 0.1610255 | 1.4931650 | 0.6215417
     1 | 1/1/17 2:00 AM        | 0.0000000 | 0.0000000 |  4.8977219 |  3.0398530 |  2.4306978 |  0.9665592 | 0.4374513 |  5.9034801 |  3.
7292500 | 1.4372492 | 0.5600358 | 0.9841425 | 1.1517180 | 0.1530861 | 0.1697570 | 0.9841425 | 0.6946892
     2 | 1/1/17 3:00 AM        | 0.0000000 | 0.0000000 |  5.1840057 |  2.0454583 |  1.8034130 |  1.4700000 | 0.7121267 |  7.1626396 |  2.
5586166 | 0.8536713 | 0.7759307 | 0.6803192 | 0.4047572 | 0.1485300 | 0.2914501 | 0.2275602 | 1.0419916
     3 | 1/1/17 4:00 AM        | 0.0000000 | 0.0000000 |  7.1791013 |  3.1248462 |  1.3947500 |  2.0447914 | 1.3389124 |  5.3575276 |  4.
8625939 | 2.3106493 | 1.5112698 | 0.4036241 | 0.4480000 | 0.1922689 | 0.2823763 | 0.3581667 | 2.1925902
     4 | 1/1/17 5:00 AM        | 0.0000000 | 0.0000000 |  5.0262373 |  1.8538333 |  1.4731622 |  0.7124614 | 1.0343689 |  2.1596130 |  4.
6231098 | 2.7532721 | 0.5750621 | 0.4592518 | 0.8078027 | 0.0575669 | 0.3454269 | 0.2912247 | 1.2230433
     5 | 1/1/17 6:00 AM        | 0.0000000 | 0.0000000 |  4.6124167 |  1.7195287 |  1.9615031 |  1.8573522 | 0.6774037 |  4.0450948 |  1.
3790403 | 1.7718750 | 1.2929583 | 0.8191930 | 0.2345423 | 0.1270330 | 0.2685349 | 0.6525869 | 1.5574495
     6 | 1/1/17 7:00 AM        | 0.0000000 | 0.0000000 |  5.2734078 |  4.0789608 |  1.3708333 |  2.0709805 | 1.0912312 |  1.9540335 |  3.
6862663 | 0.8130610 | 1.2632083 | 0.5085382 | 0.2262777 | 0.1210417 | 0.1216884 | 0.4783941 | 0.9135050
     7 | 1/1/17 8:00 AM        | 0.0000000 | 0.0000000 |  4.4732917 |  3.9719256 |  1.1689086 |  1.0325693 | 0.5089824 |  2.3541077 |  4.
3419963 | 2.8134271 | 3.5103666 | 0.8088179 | 0.7305869 | 0.1681326 | 0.3319709 | 0.5489025 | 0.4834827
```

Back-end (This Week)

Django model for the tables

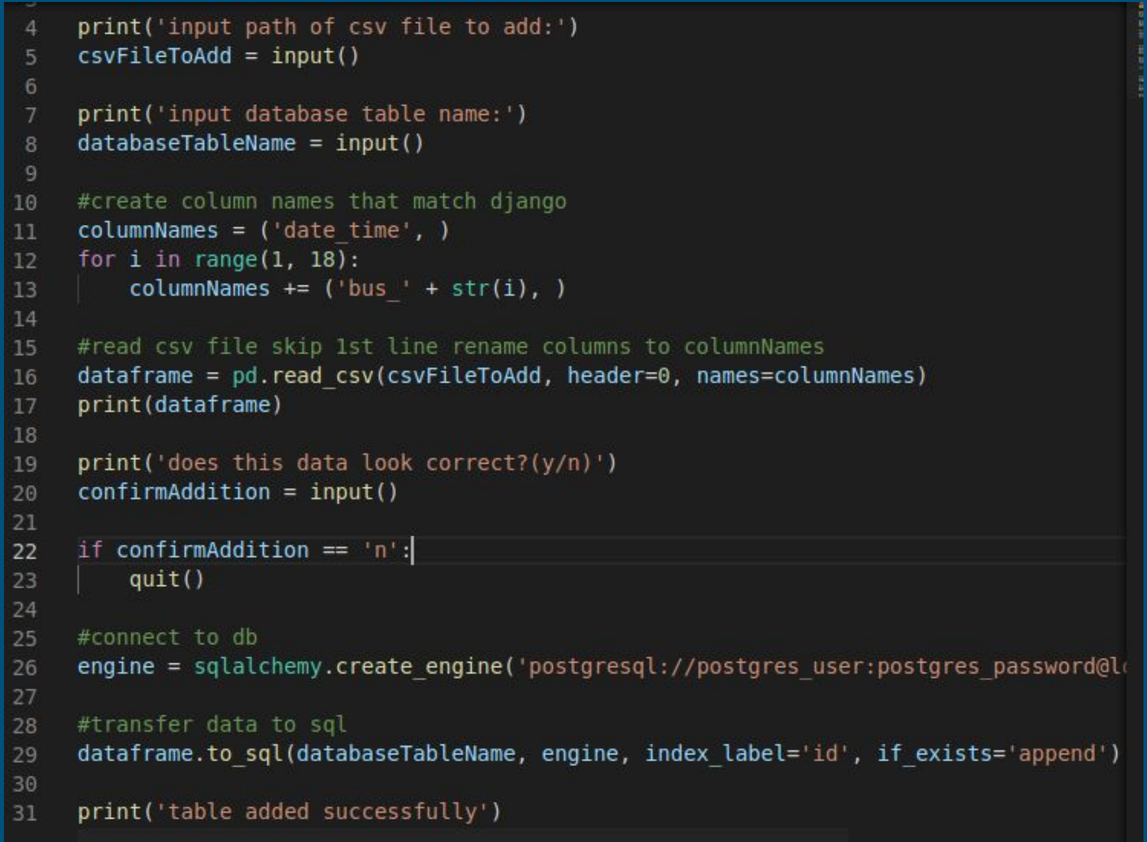
Aq model looks the same



```
class feeder_ap_data(models.Model):
    date_time = models.TextField()
    bus_1 = models.DecimalField(max_digits=11, decimal_places=7)
    bus_2 = models.DecimalField(max_digits=11, decimal_places=7)
    bus_3 = models.DecimalField(max_digits=11, decimal_places=7)
    bus_4 = models.DecimalField(max_digits=11, decimal_places=7)
    bus_5 = models.DecimalField(max_digits=11, decimal_places=7)
    bus_6 = models.DecimalField(max_digits=11, decimal_places=7)
    bus_7 = models.DecimalField(max_digits=11, decimal_places=7)
    bus_8 = models.DecimalField(max_digits=11, decimal_places=7)
    bus_9 = models.DecimalField(max_digits=11, decimal_places=7)
    bus_10 = models.DecimalField(max_digits=11, decimal_places=7)
    bus_11 = models.DecimalField(max_digits=11, decimal_places=7)
    bus_12 = models.DecimalField(max_digits=11, decimal_places=7)
    bus_13 = models.DecimalField(max_digits=11, decimal_places=7)
    bus_14 = models.DecimalField(max_digits=11, decimal_places=7)
    bus_15 = models.DecimalField(max_digits=11, decimal_places=7)
    bus_16 = models.DecimalField(max_digits=11, decimal_places=7)
    bus_17 = models.DecimalField(max_digits=11, decimal_places=7)
```

Back-end (This Week)

- Created a CSV parser script that takes the csv file and directly inputs it into the postgreSQL database



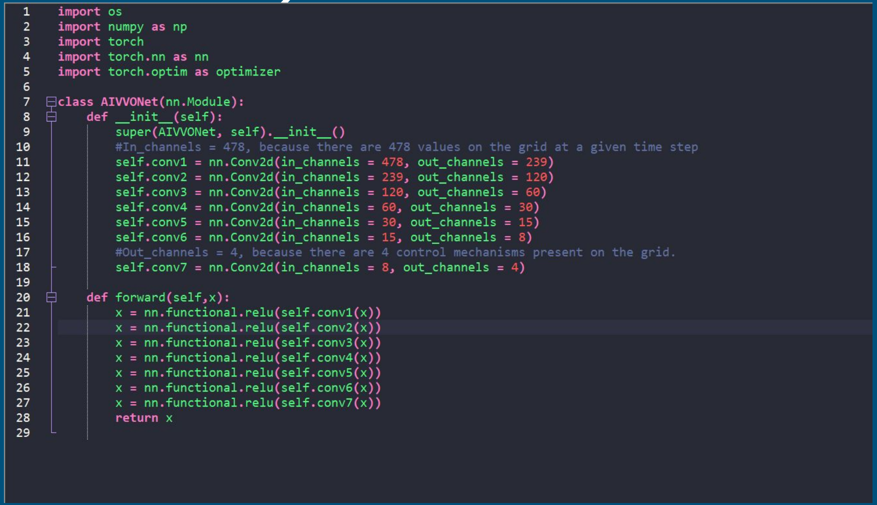
```python
print('input path of csv file to add:')
csvFileToAdd = input()

print('input database table name:')
databaseTableName = input()

#create column names that match django
columnNames = ('date_time', )
for i in range(1, 18):
    columnNames += ('bus_' + str(i), )

#read csv file skip 1st line rename columns to columnNames
dataframe = pd.read_csv(csvFileToAdd, header=0, names=columnNames)
print(dataframe)

print('does this data look correct?(y/n)')
confirmAddition = input()

if confirmAddition == 'n':
    quit()

#connect to db
engine = sqlalchemy.create_engine('postgresql://postgres_user:postgres_password@l

#transfer data to sql
dataframe.to_sql(databaseTableName, engine, index_label='id', if_exists='append')

print('table added successfully')
```

Back-end (Next Week)

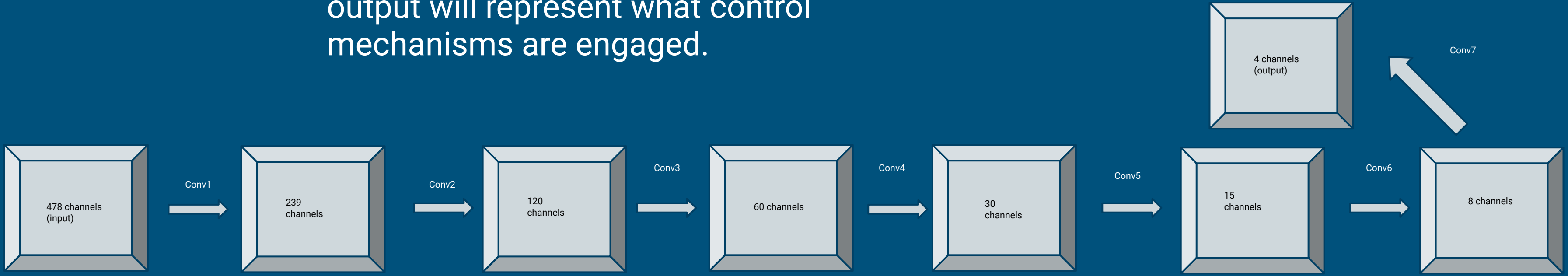
- Decide if the feeder data works well in the database table. And if it does add the Feeders B and C into datatables
- Work on the connection between ML algorithm and backend api that will send the data to the front end

Machine Learning (This Week)

- Implemented AIVVONet class on the PowerCyberTestBed and pushed class file to Gitlab
- Extended nn.Module to construct AIVVONet
- Structure of neural network is as follows:
    - 478 inputs per time step
    - 7 convolution layers with each layer reducing the size of the problem by a factor of 2
    - 4 Control Mechanisms, so the output will represent what control mechanisms are engaged.

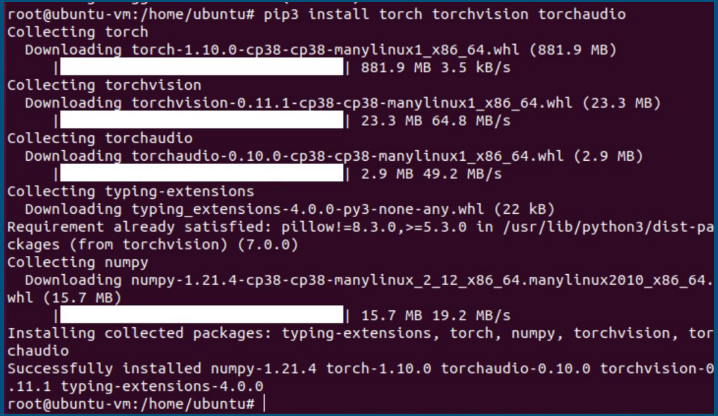


```python
import os
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optimizer

class AIVVONet(nn.Module):
    def __init__(self):
        super(AIVVONet, self).__init__()
        #In_channels = 478, because there are 478 values on the grid at a given time step
        self.conv1 = nn.Conv2d(in_channels = 478, out_channels = 239)
        self.conv2 = nn.Conv2d(in_channels = 239, out_channels = 120)
        self.conv3 = nn.Conv2d(in_channels = 120, out_channels = 60)
        self.conv4 = nn.Conv2d(in_channels = 60, out_channels = 30)
        self.conv5 = nn.Conv2d(in_channels = 30, out_channels = 15)
        self.conv6 = nn.Conv2d(in_channels = 15, out_channels = 8)
        #Out_channels = 4, because there are 4 control mechanisms present on the grid.
        self.conv7 = nn.Conv2d(in_channels = 8, out_channels = 4)

    def forward(self,x):
        x = nn.functional.relu(self.conv1(x))
        x = nn.functional.relu(self.conv2(x))
        x = nn.functional.relu(self.conv3(x))
        x = nn.functional.relu(self.conv4(x))
        x = nn.functional.relu(self.conv5(x))
        x = nn.functional.relu(self.conv6(x))
        x = nn.functional.relu(self.conv7(x))
        return x
```



Machine Learning (This Week)

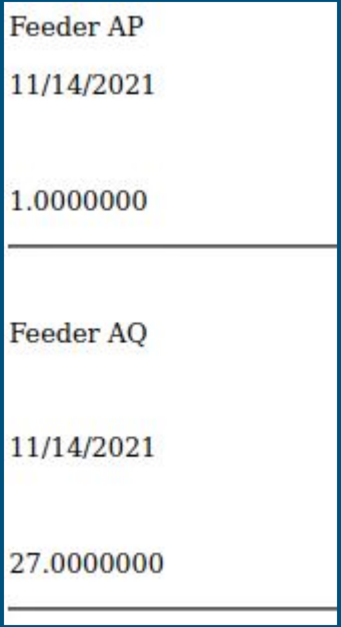
- Installed PyTorch on the PowerCyberTestbed VM
  - Installed pip3 with “apt install python3-pip”
  - Installed PyTorch dependencies with “pip3 install torch torchvision torchaudio”



```
root@ubuntu-vm:/home/ubuntu# pip3 install torch torchvision torchaudio
Collecting torch
  Downloading torch-1.10.0-cp38-cp38-manylinux1_x86_64.whl (881.9 MB)
     |████████████████████████████████| 881.9 MB 3.5 kB/s
Collecting torchvision
  Downloading torchvision-0.11.1-cp38-cp38-manylinux1_x86_64.whl (23.3 MB)
     |████████████████████████████████| 23.3 MB 64.8 MB/s
Collecting torchaudio
  Downloading torchaudio-0.10.0-cp38-cp38-manylinux1_x86_64.whl (2.9 MB)
     |████████████████████████████████| 2.9 MB 49.2 MB/s
Collecting typing-extensions
  Downloading typing_extensions-4.0.0-py3-none-any.whl (22 kB)
Requirement already satisfied: pillow!=8.3.0,>=5.3.0 in /usr/lib/python3/dist-pa
ckages (from torchvision) (7.0.0)
Collecting numpy
  Downloading numpy-1.21.4-cp38-cp38-manylinux_2_12_x86_64.manylinux2010_x86_64.
whl (15.7 MB)
     |████████████████████████████████| 15.7 MB 19.2 MB/s
Installing collected packages: typing-extensions, torch, numpy, torchvision, tor
chaudio
Successfully installed numpy-1.21.4 torch-1.10.0 torchaudio-0.10.0 torchvision-0
.11.1 typing-extensions-4.0.0
root@ubuntu-vm:/home/ubuntu# |
```

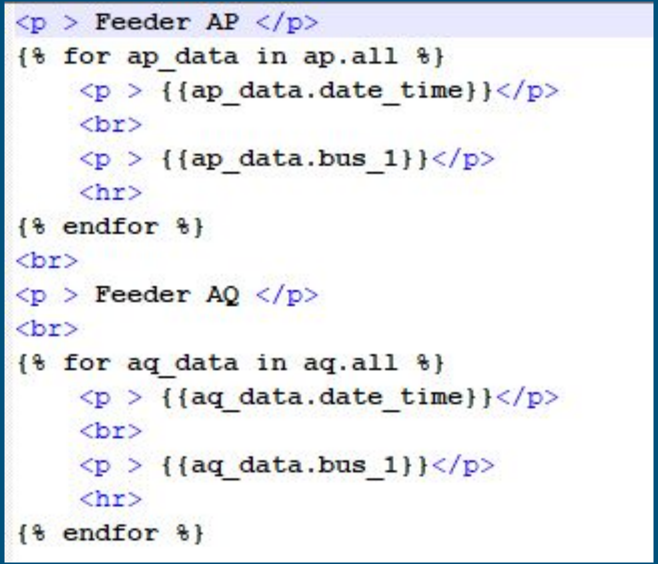
Machine Learning (This Week)

- Installed all required components of Django on VM-3
- Worked with Django to develop an understanding of how it interacts with PostgreSQL to store and retrieve data
- Create a simple test “view” to read and display data from out tables.
- Added our two models to the Django Admin page to view and manipulate for easy testing.

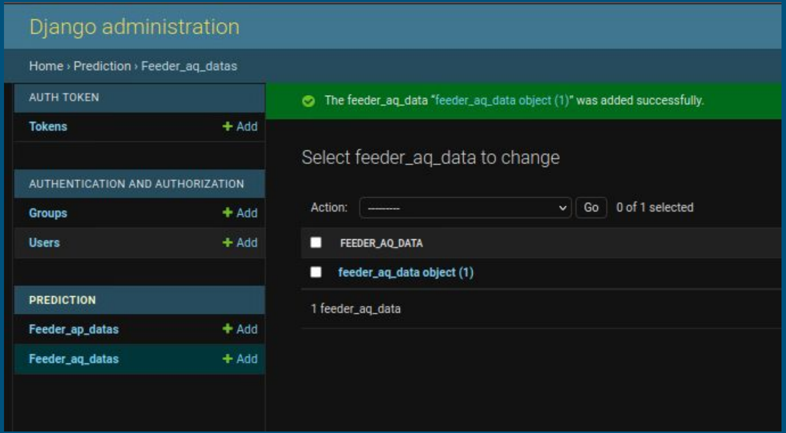


```
Feeder AP
11/14/2021

1.0000000
_______________

Feeder AQ

11/14/2021

27.0000000
_______________
```

Machine Learning (This Week)



```
<p > Feeder AP </p>
{% for ap_data in ap.all %}
    <p > {{ap_data.date_time}}</p>
    <br>
    <p > {{ap_data.bus_1}}</p>
    <hr>
{% endfor %}
<br>
<p > Feeder AQ </p>
<br>
{% for aq_data in aq.all %}
    <p > {{aq_data.date_time}}</p>
    <br>
    <p > {{aq_data.bus_1}}</p>
    <hr>
{% endfor %}
```



Django administration

Home › Prediction › Feeder_aq_datas

AUTH TOKEN
Tokens + Add

AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add

PREDICTION
Feeder_ap_datas + Add
Feeder_aq_datas + Add

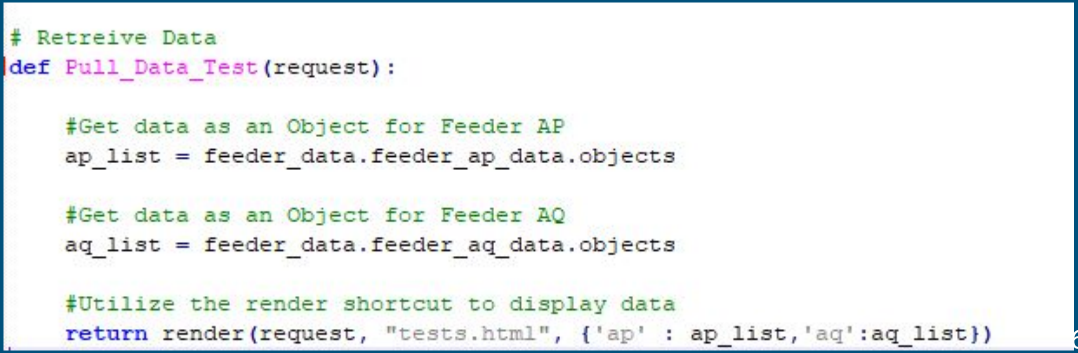
The feeder_aq_data "feeder_aq_data object (1)" was added successfully.

Select feeder_aq_data to change

Action: --------- Go 0 of 1 selected

FEEDER_AQ_DATA
feeder_aq_data object (1)

1 feeder_aq_data



```
# Retreive Data
def Pull_Data_Test(request):

    #Get data as an Object for Feeder AP
    ap_list = feeder_data.feeder_ap_data.objects

    #Get data as an Object for Feeder AQ
    aq_list = feeder_data.feeder_aq_data.objects

    #Utilize the render shortcut to display data
    return render(request, "tests.html", {'ap' : ap_list,'aq':aq_list})
```

Machine Learning (Next Week)

- GridController agent class will be implemented
- Main learning loop utilizing GridController and AIVVONet will be constructed
- Add PyTorch and apt install python3-pip to the Docker install files
- Modify the test view to be a more complete data retrieval function
- Use the data retrieval function to provide data to the main Machine Learning Algorithm.