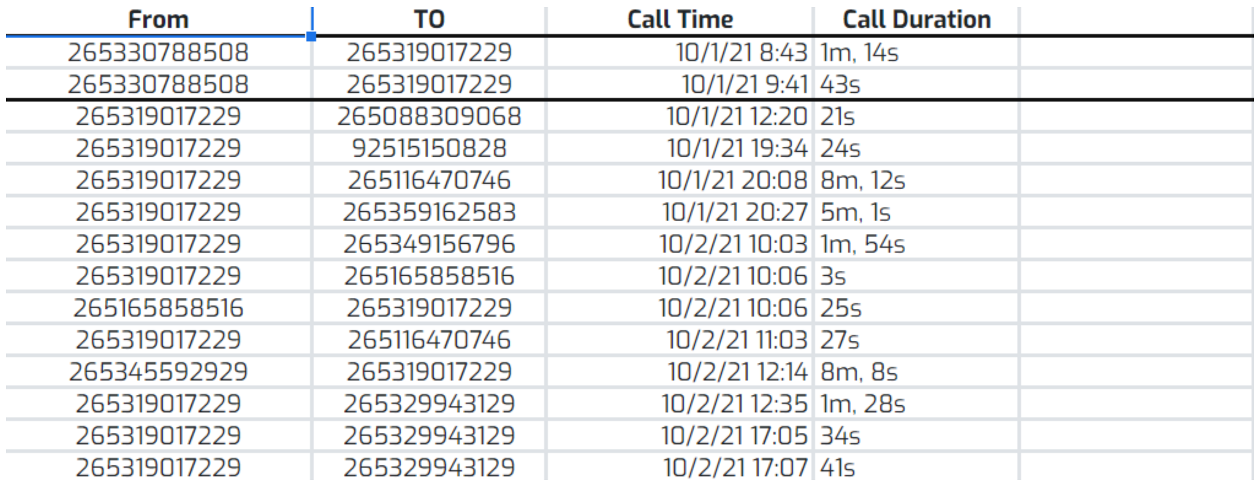
Joint-Human Machine Learning

Assignment 2

Mikias Berhanu, 2021280115

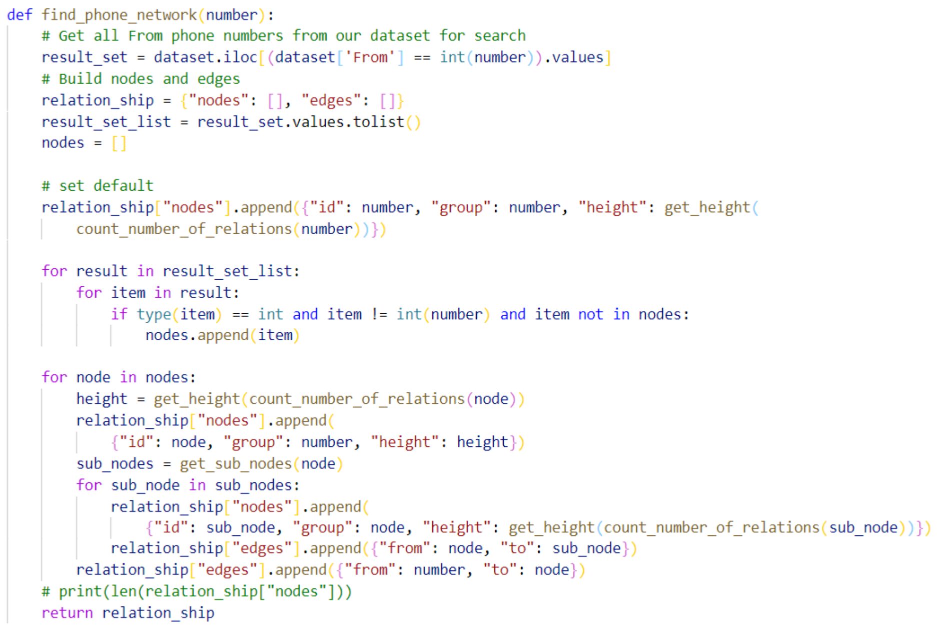
For this assignment we were asked to create a program which will display the connection of phone numbers in a network graph format based on caller and receiver information which is given as a dataset. The Network Graph visualization supports undirected and directed graph structures. This type of visualization illuminates relationships between entities. Entities are displayed as round nodes and lines show the relationships between them.



From	TO	Call Time	Call Duration	
265330788508	265319017229	10/1/21 8:43	1m, 14s	
265330788508	265319017229	10/1/21 9:41	43s	
265319017229	265088309068	10/1/21 12:20	21s	
265319017229	92515150828	10/1/21 19:34	24s	
265319017229	265116470746	10/1/21 20:08	8m, 12s	
265319017229	265359162583	10/1/21 20:27	5m, 1s	
265319017229	265349156796	10/2/21 10:03	1m, 54s	
265319017229	265165858516	10/2/21 10:06	3s	
265165858516	265319017229	10/2/21 10:06	25s	
265319017229	265116470746	10/2/21 11:03	27s	
265345592929	265319017229	10/2/21 12:14	8m, 8s	
265319017229	265329943129	10/2/21 12:35	1m, 28s	
265319017229	265329943129	10/2/21 17:05	34s	
265319017229	265329943129	10/2/21 17:07	41s	

Sample data from provided data

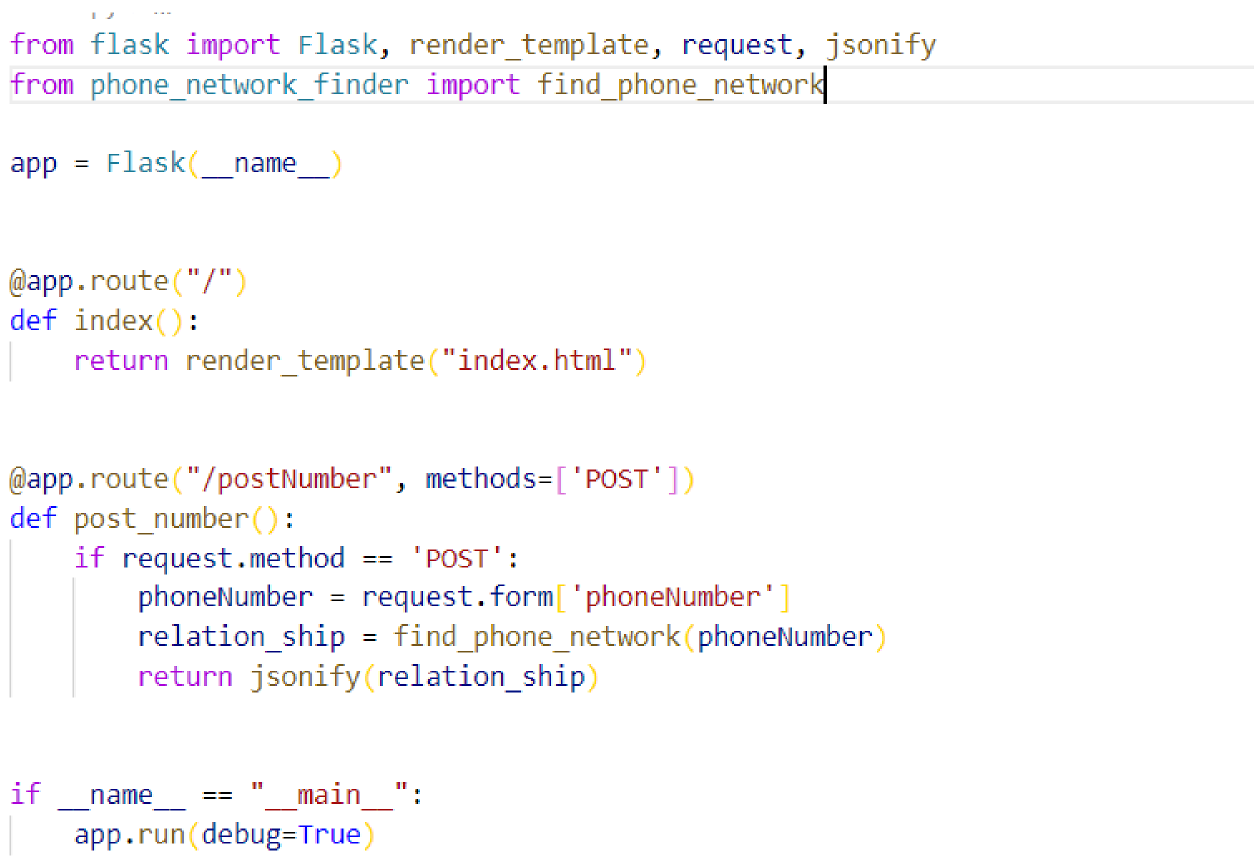
The program I made is based on Python and JavaScript to generate the network graph. I used a famous python library called flask in order to filter out the connection based on a phone number given from the user and return the data as json format which basically serves as an API. Once the backend sends data to the front end then another javascript library called anychart is used to generate the graphs.



```python
def find_phone_network(number):
    # Get all From phone numbers from our dataset for search
    result_set = dataset.iloc[(dataset['From'] == int(number)).values]
    # Build nodes and edges
    relation_ship = {"nodes": [], "edges": []}
    result_set_list = result_set.values.tolist()
    nodes = []

    # set default
    relation_ship["nodes"].append({"id": number, "group": number, "height": get_height(
        count_number_of_relations(number))})

    for result in result_set_list:
        for item in result:
            if type(item) == int and item != int(number) and item not in nodes:
                nodes.append(item)

    for node in nodes:
        height = get_height(count_number_of_relations(node))
        relation_ship["nodes"].append(
            {"id": node, "group": number, "height": height})
        sub_nodes = get_sub_nodes(node)
        for sub_node in sub_nodes:
            relation_ship["nodes"].append(
                {"id": sub_node, "group": node, "height": get_height(count_number_of_relations(sub_node))})
            relation_ship["edges"].append({"from": node, "to": sub_node})
        relation_ship["edges"].append({"from": number, "to": node})
    # print(len(relation_ship["nodes"]))
    return relation_ship
```

Function find_phone_network filters out connections between phone numbers

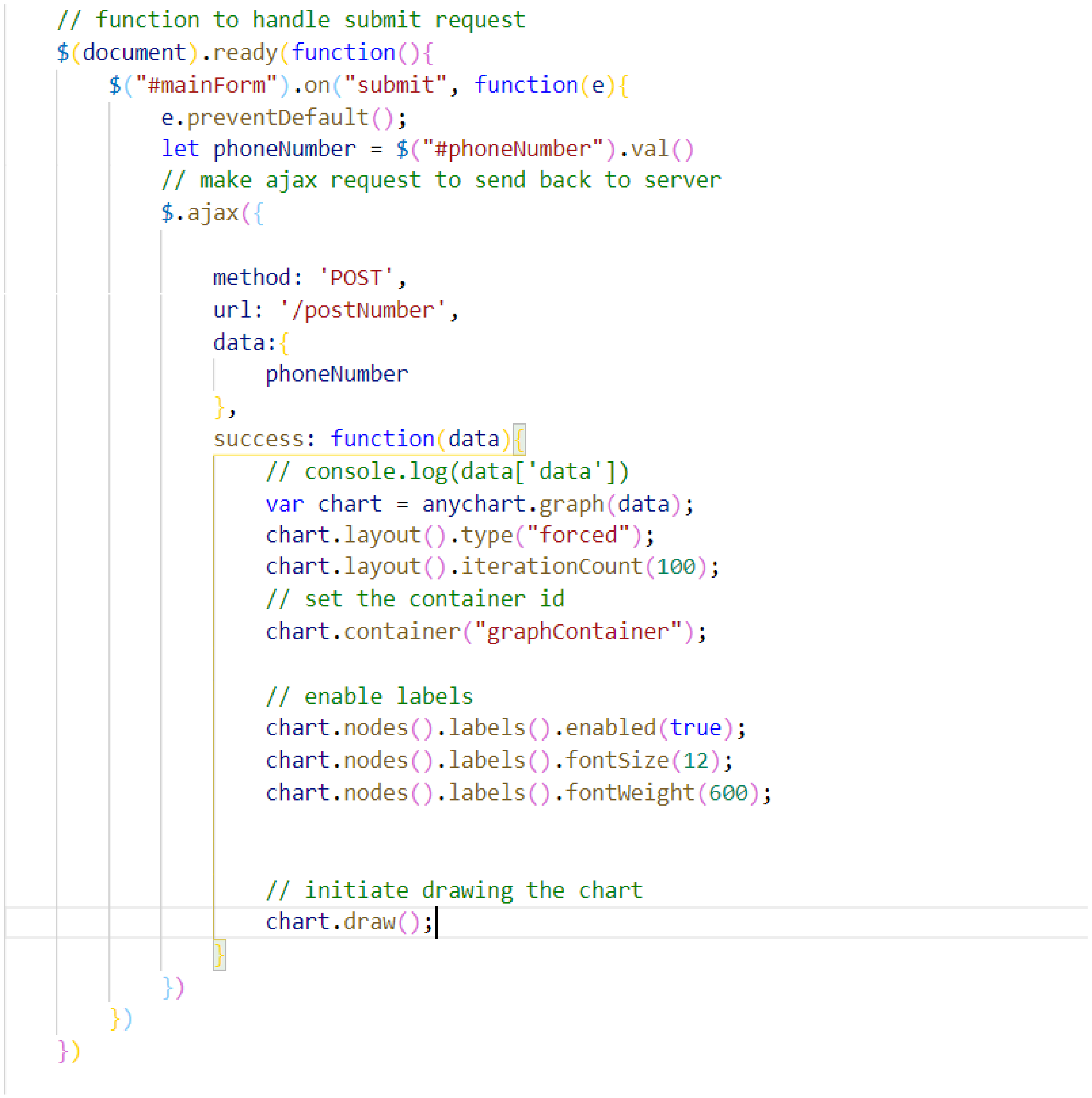
The above function filters out connections between the given number from the user and other phone numbers in our dataset. It will also call get_sub_nodes to generate other connections which are not part of the other nodes connected with the input phone number but not the same group. These nodes are saved in a dictionary called relationship which stores the nodes of the network and the relationship they have in the form of from and to patten so that later it can be rendered using javascript.



```python
from flask import Flask, render_template, request, jsonify
from phone_network_finder import find_phone_network

app = Flask(__name__)


@app.route("/")
def index():
    return render_template("index.html")


@app.route("/postNumber", methods=['POST'])
def post_number():
    if request.method == 'POST':
        phoneNumber = request.form['phoneNumber']
        relation_ship = find_phone_network(phoneNumber)
        return jsonify(relation_ship)


if __name__ == "__main__":
    app.run(debug=True)
```

Route Handling and API response handler

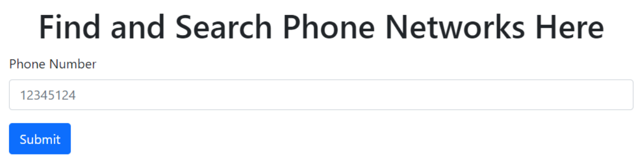
The main application has two main routes index and post_number which are responsible for handling user requests. The index function/route renders the index page which takes user input as well as displays the graph generated by the user input. The post_number function/route take the user input from the post request and calls find_phone_network function to generate an API response which is then sent back to the client side to be processed using JavaScript.

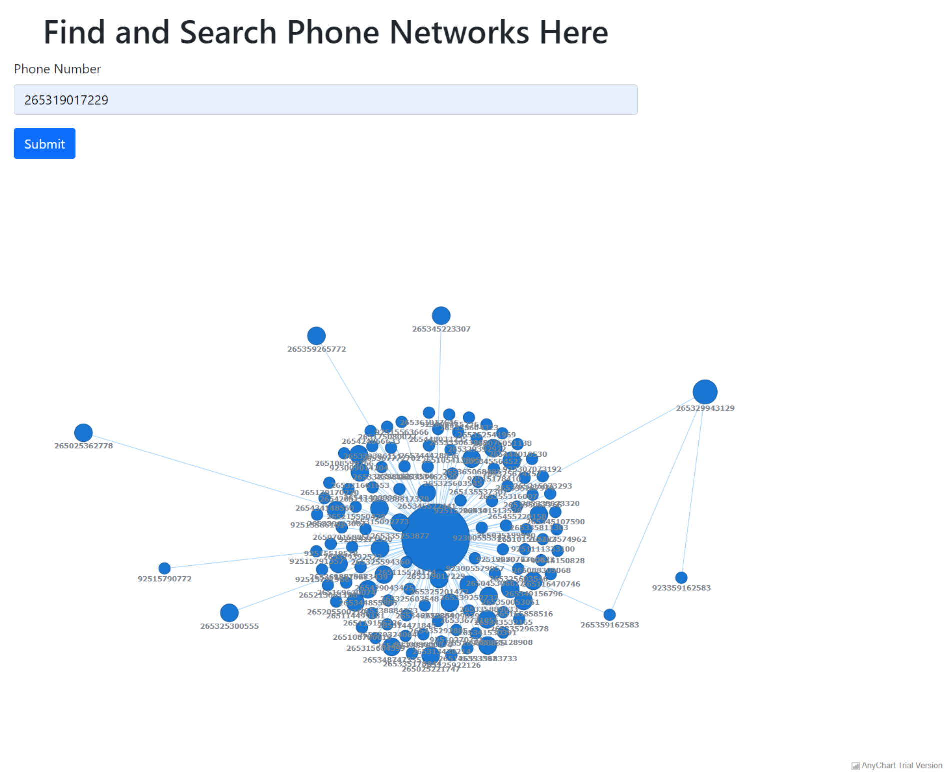


```
// function to handle submit request
$(document).ready(function(){
    $("#mainForm").on("submit", function(e){
        e.preventDefault();
        let phoneNumber = $("#phoneNumber").val()
        // make ajax request to send back to server
        $.ajax({

            method: 'POST',
            url: '/postNumber',
            data:{
                phoneNumber
            },
            success: function(data){
                // console.log(data['data'])
                var chart = anychart.graph(data);
                chart.layout().type("forced");
                chart.layout().iterationCount(100);
                // set the container id
                chart.container("graphContainer");

                // enable labels
                chart.nodes().labels().enabled(true);
                chart.nodes().labels().fontSize(12);
                chart.nodes().labels().fontWeight(600);


                // initiate drawing the chart
                chart.draw();
            }
        })
    })
})
```

Java Script code snippet for sending and receiving API requests using AJAX

The Javascript code takes user input from the HTML form and sends post request API call to the server. Once the response is sent back from the server it uses anychart library to generate network graphs based on the data received. When we put all these components together will get the results show below.





Find and Search Phone Networks Here

Phone Number

265319017229

Submit

The code and other details have been linked [here]()