1. (Figure 1) Load the housing_price.csv file and create the following map visualization with Folium.

a. The housing_price.csv spreadsheet include four columns: latitude, longitude, streetname, streetno, and price2014.

b. Place markers on a base map (OpenStreetMap). Each marker represents one house based on its latitude and longitude. Each marker should be a circle filled with red color and with a black line.

c. When the mouse cursor hovers over a marker, the streetno and streetname, and price2014 should be displayed in a tooltip.

In [1]:

```
import pandas as pd
import folium
import numpy as np
```

In [2]:

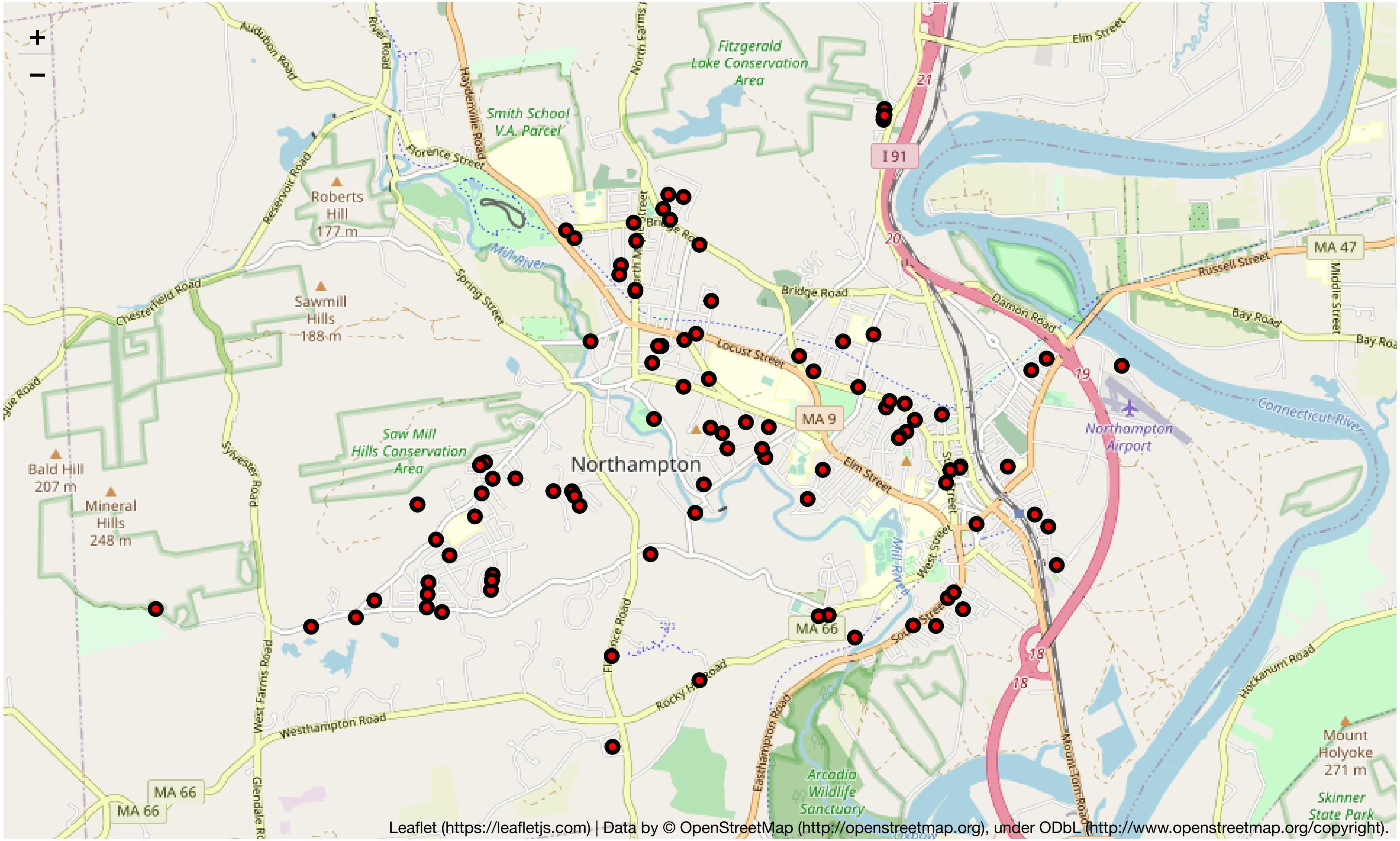
```
df1 = pd.read_csv('housing_price.csv')
df1 = df1[['latitude', 'longitude', 'streetname', 'streetno', 'price2014']]
df1.head()
```

Out[2]:

	latitude	longitude	streetname	streetno	price2014
0	42.31533	-72.6940	Acrebrook Drive	406	210.729
1	42.29856	-72.6747	Autumn Dr	57	204.171
2	42.34379	-72.6802	Bridge Road	31	338.662
3	42.34446	-72.6722	Bridge Road	200	276.250
4	42.34253	-72.6644	Bridge Road	395	169.173

```
In [3]: locations = list(zip(df1['latitude'], df1['longitude']))
        tooltips = df1['streetname'].astype(str) +', '+ df1['streetno'].astype(str) +', housePrice: '+df1['price2014'].astype(str) + '$'

        fig1 = folium.Map(
            location =[np.mean(df1['latitude']), np.mean(df1['longitude'])],
            titles = 'fig 1. House pricing',
            zoom_start = 13
        )

        for i  in range(0, len(locations)):
                        folium.Circle(
                            location = locations[i],
                            radius=50,
                            color = 'black',
                            fill_color = 'red',
                            fill=True,
                            fill_opacity=1,
                            tooltip =tooltips[i]
                    ).add_to(fig1)

        fig1
```

Out[3]:



Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

1. (Figure 2) Create a network visualization of the 2019 Women's World Cup Bracket using python-graphviz.

a. Here are some examples for your reference. You do not have to follow these formats.
i. https://www.fifa.com/womensworldcup/matches/ (https://www.fifa.com/womensworldcup/matches/) ii. http://www.espn.com/soccer/bracket/_/league/fifa.wwc (http://www.espn.com/soccer/bracket/_/league/fifa.wwc)

b. Each node should contain the names of the countries and the score. National flags are optional.

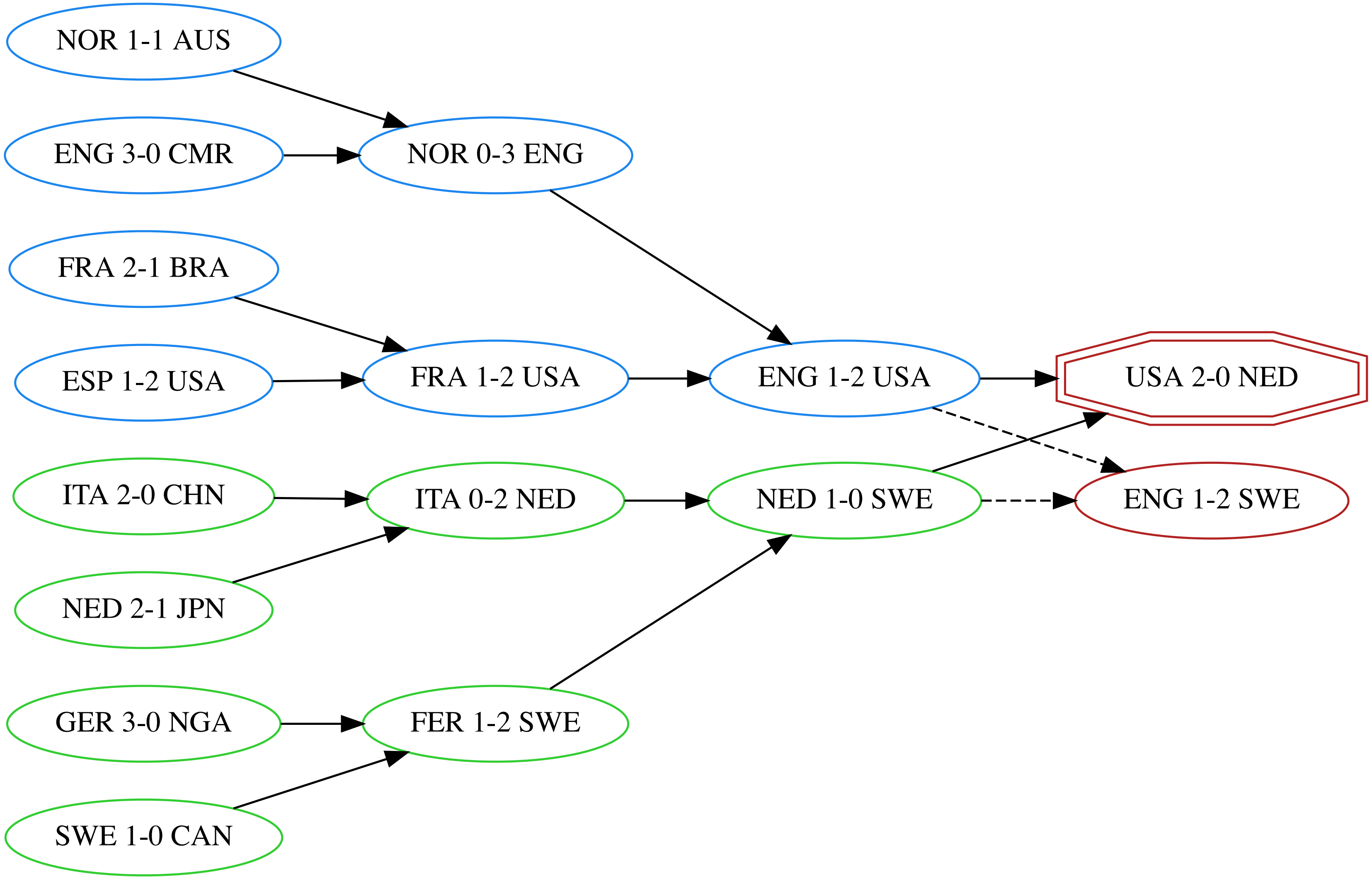
c. This must be a directional graph. There must be edges with arrows pointing from one stage to the next.

In [4]:

```
# % conda install python-graphviz
import graphviz
d = graphviz.Digraph(name = "2019 Women's world cup bracket", filename='worldCup.gv')
d.attr(rankdir = 'LR')

d.node(name = 'A', label = 'NOR 1-1 AUS', color = "dodgerblue2")
d.node(name = 'B', label = 'ENG 3-0 CMR', color = "dodgerblue2")
d.node(name = 'C', label = 'FRA 2-1 BRA', color = "dodgerblue2")
d.node(name = 'D', label = 'ESP 1-2 USA', color = "dodgerblue2")

d.node(name = 'E', label = 'NOR 0-3 ENG', color = "dodgerblue2")
d.node(name = 'F', label = 'FRA 1-2 USA', color = "dodgerblue2")

d.node(name = 'G', label = 'ENG 1-2 USA', color = "dodgerblue2")

d.node(name = 'H', label = 'USA 2-0 NED', color = "firebrick", shape = "doubleoctagon")
d.node(name = 'I', label = 'ENG 1-2 SWE', color = "firebrick")

d.node(name = 'J', label = 'NED 1-0 SWE', color = "limegreen")

d.node(name = 'K', label = 'ITA 0-2 NED', color = "limegreen")
d.node(name = 'L', label = 'FER 1-2 SWE', color = "limegreen")

d.node(name = 'M', label = 'ITA 2-0 CHN', color = "limegreen")
d.node(name = 'N', label = 'NED 2-1 JPN', color = "limegreen")
d.node(name = 'O', label = 'GER 3-0 NGA', color = "limegreen")
d.node(name = 'P', label = 'SWE 1-0 CAN', color = "limegreen")

d.edge('A', 'E')
d.edge('B', 'E')
d.edge('C', 'F')
d.edge('D', 'F')

d.edge('E', 'G')
d.edge('F', 'G')

d.edge('G', 'H')
d.edge('J', 'H')
d.edge('G', 'I', style = 'dashed')
d.edge('J', 'I', style = 'dashed')


d.edge('K', 'J')
```

```
d.edge('L', 'J')

d.edge('M', 'K')
d.edge('N', 'K')
d.edge('O', 'L')
d.edge('P', 'L')

d
```

Out[4]:



1. (Figure 3) Create a music collaboration network visualization using NetworkX and Plotly.

a. Select 10 singers/musicians. For each musician, identify at least two collaborators and the songs they collaborated on. The more artists you include in your visualization the better. i. For example, Drake collaborated with Rihanna on “Work” and “What’s My Name?”, with Lil Wayne on “She Will”, etc.

b. Create a network visualization of the collaborations. i. Each node represents a musician with the name of the musician displayed. ii. Each edge represents a collaboration between two musicians. The name of the song should be displayed next to the edge.
iii. If more than two musicians collaborated in a project, each musician should be connected to every other musician. iv. If two musicians collaborated more than once, create multiple edges between them.
v. Pictures are optional. vi. Here is an example of Jazz music collaboration network visualization:

c. Create THREE visualizations with three different layout algorithms. i. Some layout may not look good. It’s OK. The goal is to let you experiment with different layouts and learn how to adjust layout parameters.

d. You can choose the style of the visualization.

e. You decide how to handle the data. You may hard code the data in the Python program or create a spreadsheet and load it into your program. i. If you use a spreadsheet, make sure you submit the spreadsheet with your code and PDF file.

f. You can find musicians, songs, and their collaborators at https://www.billboard.com/charts/artist-100 (https://www.billboard.com/charts/artist-100). Click on an artist and look at his/her Chart History. Or use your own source of information.

g. Visualize your network with Plotly.

In [5]:

```
import plotly.graph_objects as go
import networkx as nx
```

In [6]:
```
df3 = pd.read_csv('singer.csv')
df3.head()
```

Out[6]:

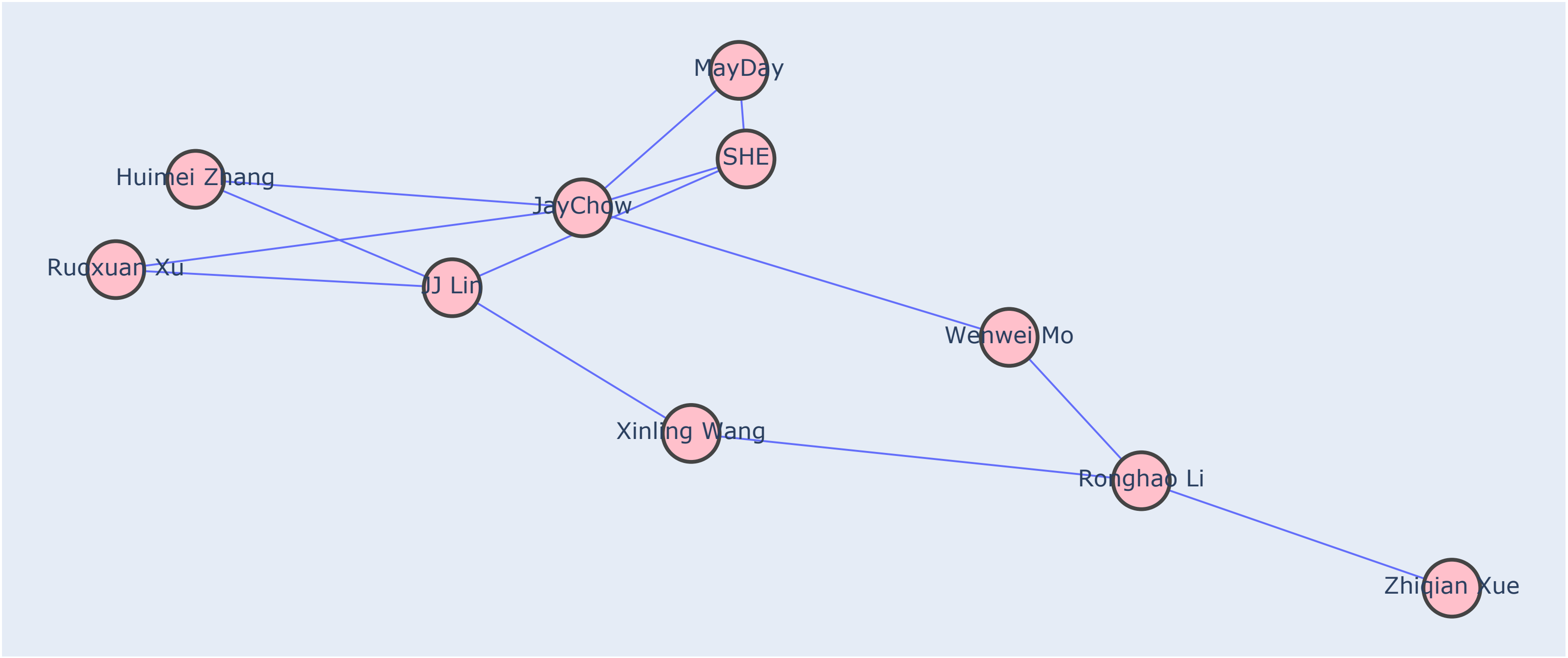
	name1	name2	song
0	JayChow	MayDay	ShuoHaoBuKu
1	JayChow	SHE	HouNiao
2	JayChow	Ruoxuan Xu	MianJu
3	Ruoxuan Xu	JJ Lin	AiXiaoDeYanJing
4	Xinling Wang	JJ Lin	DangNi

In [7]:
```
G = nx.Graph()
G.add_nodes_from(list(set().union(df3.name1, df3.name2)))
G.add_edges_from(list(zip(df3.name1, df3.name2)))
```

In [8]:

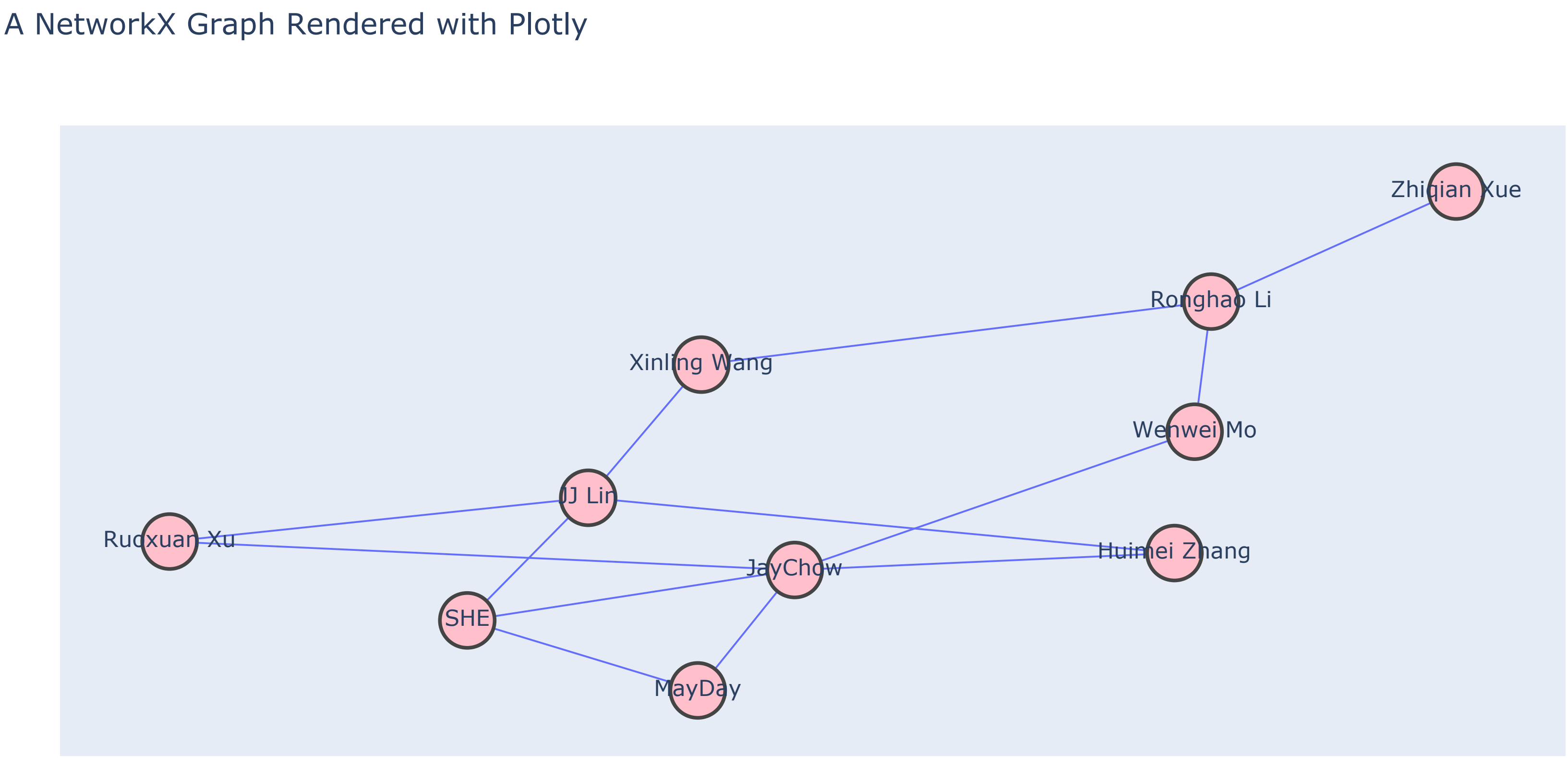
```
pos = nx.spring_layout(G)

def netWorkDraw():

    edge_x = []
    edge_y = []
    for edge in G.edges():
        x0 = pos[edge[0]][0]
        y0 = pos[edge[0]][1]
        x1 = pos[edge[1]][0]
        y1 = pos[edge[1]][1]
        edge_x.append(x0)
        edge_x.append(x1)
        edge_x.append(None)
        edge_y.append(y0)
        edge_y.append(y1)
        edge_y.append(None)

    # Create a line plot to draw all the edges.
    edge_trace = go.Scatter(
        x=edge_x,
        y=edge_y,
        mode='lines',
        line = dict(width = 1))

    # Create a node list
    node_x = []
    node_y = []
    for node in G.nodes():
        # Saving node coordinates to the node list.
        x = pos[node][0]
        y = pos[node][1]
        node_x.append(x)
        node_y.append(y)

    # Create a scatter plot to draw all the nodes.
    node_trace = go.Scatter(
        x=node_x,
        y=node_y,
        mode="markers + text",
        text = list(G.nodes),
        textposition = "middle center",
        hoverinfo = "text",
        marker=dict(
            size=30,
            color= "Pink",
            line_width=2))

    fig = go.Figure(data=[edge_trace, node_trace],
                 layout=go.Layout(
                    title="A NetworkX Graph Rendered with Plotly",
                    titlefont_size=16,
                    showlegend=False,
                    xaxis=dict(showgrid=False, zeroline=False, showticklabels=False),
                    yaxis=dict(showgrid=False, zeroline=False, showticklabels=False))
                   )
    return fig.show()

netWorkDraw()
```

A NetworkX Graph Rendered with Plotly



In [9]:

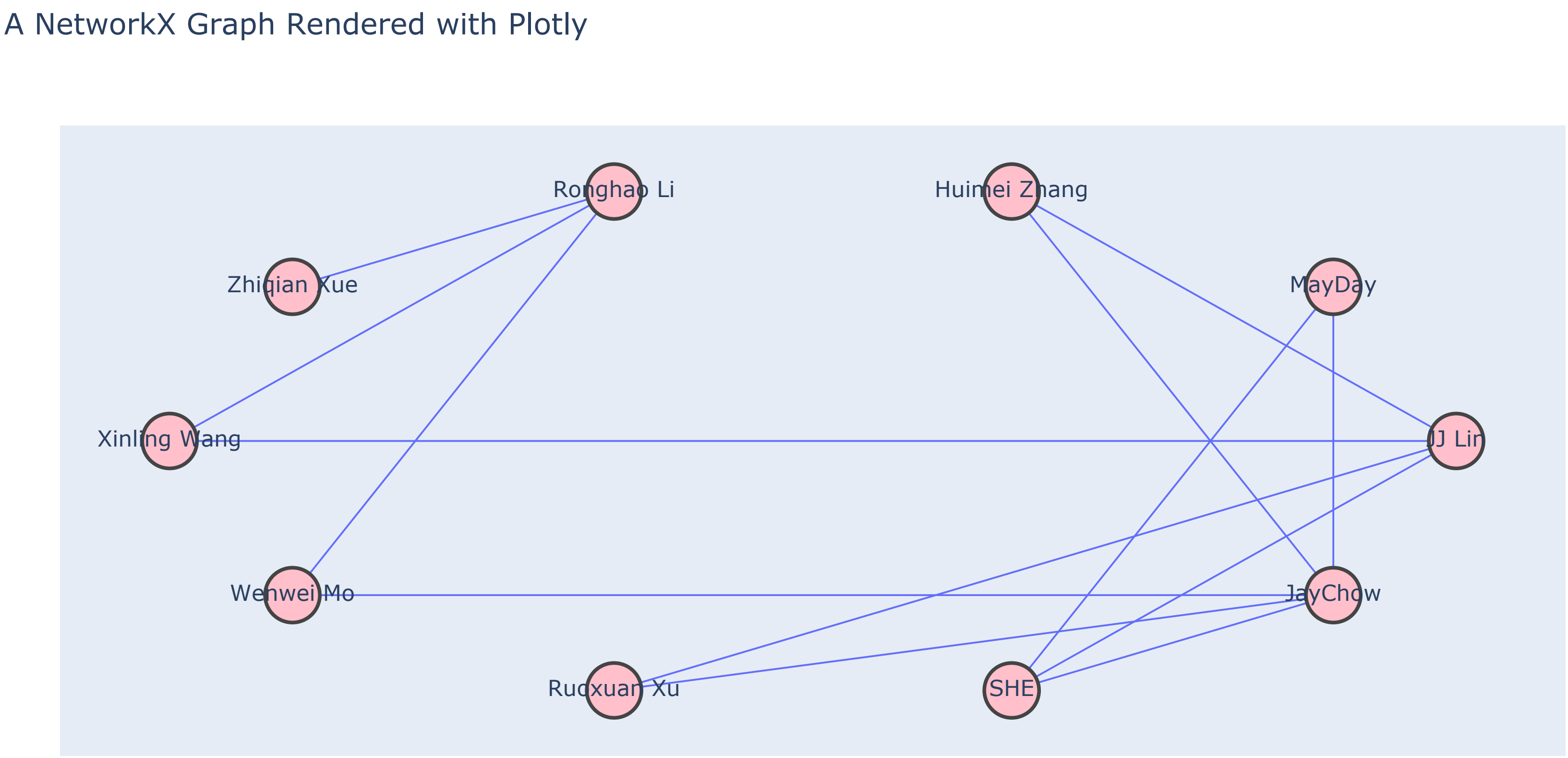
```
pos = nx.fruchterman_reingold_layout(G)

netWorkDraw()
```



A NetworkX Graph Rendered with Plotly

In [10]:

```
pos = nx.shell_layout(G)

netWorkDraw()
```



In [ ]:

```
```