import plotly.graph\_objects as go

# Ensure transition weights are represented in the Sankey diagram

if len(journey\_edges) == 0:

print(f"No journey data to plot for User {user\_to\_visualize}.")

else:

# Create lists for Sankey Diagram

sources = []

targets = []

weights = []

labels = set()

# Add edges with weights (transition counts)

for edge in journey\_edges:

source, target = edge[0], edge[1]

if len(edge) > 2: # Check if weight is included in edge

weight = edge[2]

else:

weight = 1 # Default weight if not specified

sources.append(source)

targets.append(target)

weights.append(weight)

labels.add(source)

labels.add(target)

# Convert labels to list and create a mapping for indices

labels = list(labels)

label\_to\_index = {label: i for i, label in enumerate(labels)}

# Sort nodes for better visualization (optional custom sorting logic)

labels.sort(key=lambda x: ('Exit' in x, x)) # Example: place "Exit" nodes at the end

# Convert sources and targets to indices

sources\_indices = [label\_to\_index[source] for source in sources]

targets\_indices = [label\_to\_index[target] for target in targets]

# Generate labels for links (arrows) with transition weights

link\_labels = [f"{weight} transitions" for weight in weights]

# Create the Sankey Diagram

fig = go.Figure(go.Sankey(

arrangement="snap", # Arrange nodes more neatly

node=dict(

pad=20, # Increase padding between nodes

thickness=20,

line=dict(color="black", width=0.5),

label=labels,

color=['#66c2a5' if 'Exit' not in label else '#fc8d62' for label in labels] # Example color scheme

),

link=dict(

source=sources\_indices,

target=targets\_indices,

value=weights,

customdata=link\_labels,

hovertemplate="From %{source.label} to %{target.label}<br>%{value} transitions<extra></extra>",

color='rgba(44, 160, 44, 0.6)' # Semi-transparent green for links

)

))

fig.update\_layout(

title\_text=f"Sankey Diagram for User ID: {user\_to\_visualize}",

font\_size=12,

title\_font\_size=16

)

fig.show()

# Print the ordered journey

ordered\_journey = " -> ".join([edge[0] for edge in journey\_edges] + [journey\_edges[-1][1]])

print(f"Ordered Journey for User {user\_to\_visualize}:")

print(ordered\_journey)