In [ ]:

```python
import streamlit as st
import pandas as pd
import plotly.express as px
import networkx as nx
import plotly.graph_objects as go


from langchain_ollama import OllamaLLM
from langchain.agents import AgentType, initialize_agent, Tool


df = pd.read_csv("C:/Users/divye/Documents/Flight Scheduling/VABB_simulated_flights_
st.set_page_config(page_title="Flight Scheduling Dashboard + NLP", layout="wide")
st.title("Flight Scheduling Dashboard & NLP Agent")


@st.cache_resource
def load_nlp_agent(df):

    def get_busiest_hours(n_str: str = "5") -> str:
        try:
            n = int("".join(filter(str.isdigit, n_str)))
            return df.groupby('dep_hour').size().sort_values(ascending=False).head(n
        except:
            return df.groupby('dep_hour').size().sort_values(ascending=False).head(5

    def get_best_hours(n_str: str = "5") -> str:
        try:
            n = int("".join(filter(str.isdigit, n_str)))
            avg_delay = df.groupby('dep_hour')['dep_delay_min'].mean().sort_values()
            return avg_delay.head(n).to_string()
        except:
            avg_delay = df.groupby('dep_hour')['dep_delay_min'].mean().sort_values()
            return avg_delay.head(5).to_string()

    tools = [
        Tool(
            name="Get Busiest Hours",
            func=get_busiest_hours,
            description="Returns the top N busiest hours (input example: '3')."
        ),
        Tool(
            name="Get Best Hours",
            func=get_best_hours,
            description="Returns the top N best (least delayed) hours (input example
        )
    ]

    llm = OllamaLLM(model="llama3", temperature=0)
    agent = initialize_agent(
        tools,
        llm,
        agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
        verbose=True,
        handle_parsing_errors=True,
    )
    return agent

agent = load_nlp_agent(df)

st.sidebar.header("Ask the Flight AI Agent")
user_query = st.sidebar.text_input("Type your question here:")
```

```
if user_query:
    try:
        response = agent.run(user_query)
        st.sidebar.markdown(f"**Response:** {response}")
    except Exception as e:
        st.sidebar.error(f"Error: {e}")

st.subheader("Hourly Flight Load")
hourly_load = df.groupby('dep_hour').size().reset_index(name='num_flights')
fig_load = px.bar(hourly_load, x='dep_hour', y='num_flights',
                  labels={'dep_hour':'Departure Hour', 'num_flights':'Number of Flig
                  title="Number of Flights by Hour")
st.plotly_chart(fig_load, use_container_width=True)

st.subheader("Predicted vs Scheduled Delays")
fig_delay = px.scatter(df, x='scheduled_dep', y='optimized_pred_delay',
                       color='dep_delay_min',
                       labels={'scheduled_dep':'Scheduled Departure', 'optimized_pre
                       title="Predicted Delay vs Scheduled Departure")
st.plotly_chart(fig_delay, use_container_width=True)

st.subheader("Cascading Delay Impact")
G = nx.DiGraph()
for _, row in df.iterrows():
    if row['delay_propagation'] > 0:
        G.add_edge(row['flight_id'], f"{row['flight_id']}_next", weight=row['delay_p

pos = nx.spring_layout(G, seed=42)
edge_x, edge_y = [], []
for edge in G.edges():
    x0, y0 = pos[edge[0]]
    x1, y1 = pos[edge[1]]
    edge_x += [x0, x1, None]
    edge_y += [y0, y1, None]

edge_trace = go.Scatter(x=edge_x, y=edge_y, line=dict(width=1, color='#888'), hoveri
node_x, node_y = [], []
for node in G.nodes():
    x, y = pos[node]
    node_x.append(x)
    node_y.append(y)

node_trace = go.Scatter(x=node_x, y=node_y, mode='markers+text', hoverinfo='text',
                        text=[node for node in G.nodes()], marker=dict(color='skyblu
fig_net = go.Figure(data=[edge_trace, node_trace],
                    layout=go.Layout(title='Flight Cascading Network', showlegend=Fa
st.plotly_chart(fig_net, use_container_width=True)

st.subheader("Schedule Optimization: Before vs After")
fig_opt = px.bar(df.head(20), x='flight_id', y=['original_total_delay', 'optimized_t
                 labels={'value':'Total Delay (min)', 'flight_id':'Flight ID'},
                 title="Original vs Optimized Total Delay for Top 20 Flights")
st.plotly_chart(fig_opt, use_container_width=True)
```