```python
import streamlit as st
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from PIL import Image

# Set page configuration
st.set_page_config(page_title="Decision Intelligence Dashboard", layout="wide")

# Load Data
@st.cache_data
def load_data():
    forecast_df = pd.read_csv('revenue_forecast_scenarios.csv')
    roi_df = pd.read_csv('roi_simulation_results.csv')
    segment_df = pd.read_csv('segment_decision_summary.csv')

    # Preprocessing
    forecast_df['Date'] = pd.to_datetime(forecast_df['Date'], dayfirst=True)
    return forecast_df, roi_df, segment_df

forecast_df, roi_df, segment_df = load_data()

# --- HEADER WITH LOGO ---
col_logo, col_title = st.columns([1, 5])
with col_logo:
    try:
        logo = Image.open('Mu_sigma_logo.jpg')
        st.image(logo, width=150)
    except:
        st.warning("Logo file not found.")

with col_title:
    st.title("Decision Intelligence Dashboard for Revenue Growth & Risk Management")

st.markdown("---")

# Sidebar Navigation
st.sidebar.title("Navigation")
page = st.sidebar.radio("Go to", ["Executive Summary", "Customer Segmentation", "Revenue Forecasting", "ROI Analysis"])

# --- PAGE 1: EXECUTIVE SUMMARY ---
if page == "Executive Summary":
    st.header("рџ“Љ Executive Summary")

    # Top Row Metrics
    col1, col2, col3, col4 = st.columns(4)
    total_customers = segment_df['Customer_Count'].sum()
    total_gain = roi_df['Projected_Gain'].sum()
    avg_roi = roi_df['ROI'].mean()
    total_investment = roi_df['Investment'].sum()

    col1.metric("Total Customers", f"{total_customers:,}")
    col2.metric("Projected Gain", f"${total_gain:,.2f}")
    col3.metric("Avg ROI", f"{avg_roi:,.1f}x")
    col4.metric("Total Investment", f"${total_investment:,.0f}")

    st.markdown("---")

    # Quick View: Segmentation vs ROI
    c1, c2 = st.columns(2)
    with c1:
        st.subheader("Customer Distribution by Segment")
        fig_seg = px.pie(segment_df, values='Customer_Count', names='Decision_Action', hole=0.4,
                         color_discrete_sequence=px.colors.qualitative.Pastel)
        st.plotly_chart(fig_seg, use_container_width=True)

    with c2:
        st.subheader("ROI by Segment")
        fig_roi = px.bar(roi_df.sort_values('ROI', ascending=False), x='Segment', y='ROI',
                         color='ROI', color_continuous_scale='Viridis')
        st.plotly_chart(fig_roi, use_container_width=True)

# --- PAGE 2: CUSTOMER SEGMENTATION ---
elif page == "Customer Segmentation":
    st.header("рџ‘Ґ Customer Segmentation Analysis")

    st.dataframe(segment_df.style.background_gradient(subset=['Avg_Monetary', 'Customer_Count'], cmap='Greens'), use_container_width=True)

    col1, col2 = st.columns(2)
    with col1:
        st.subheader("Recency vs Frequency")
        fig = px.scatter(segment_df, x='Avg_Recency', y='Avg_Frequency',
                         size='Customer_Count', color='Decision_Action',
                         hover_name='Cluster', text='Cluster')
        st.plotly_chart(fig, use_container_width=True)

    with col2:
        st.subheader("Monetary Value by Cluster")
        fig2 = px.bar(segment_df, x='Cluster', y='Avg_Monetary', color='Decision_Action')
        st.plotly_chart(fig2, use_container_width=True)

# --- PAGE 3: REVENUE FORECASTING ---
elif page == "Revenue Forecasting":
    st.header("рџ“€ Revenue Forecasting Scenarios")

    scenarios = st.multiselect("Select Scenarios to Display",
                               ['Base_Forecast', 'Best_Case', 'Worst_Case'],
                               default=['Base_Forecast', 'Best_Case', 'Worst_Case'])

    fig = go.Figure()

    # Add Shaded Confidence Interval
    fig.add_trace(go.Scatter(
        x=forecast_df['Date'].tolist() + forecast_df['Date'].tolist()[::-1],
        y=forecast_df['Upper_CI'].tolist() + forecast_df['Lower_CI'].tolist()[::-1],
        fill='toself',
        fillcolor='rgba(0,100,80,0.1)',
        line_color='rgba(255,255,255,0)',
        name='95% Confidence Interval',
    ))

    # Add lines
    colors = {'Base_Forecast': '#1f77b4', 'Best_Case': '#2ca02c', 'Worst_Case': '#d62728'}
    for scenario in scenarios:
        fig.add_trace(go.Scatter(x=forecast_df['Date'], y=forecast_df[scenario],
                                 name=scenario, line=dict(color=colors[scenario], width=3)))

    fig.update_layout(title="6-Month Revenue Projection", xaxis_title="Date", yaxis_title="Revenue ($)", hovermode="x unified")
    st.plotly_chart(fig, use_container_width=True)

    st.info("The shaded area represents the 95% Confidence Interval for the Base Forecast.")

# --- PAGE 4: ROI ANALYSIS ---
```

```
elif page == "ROI Analysis":
    st.header("pµ'° Investment & ROI Simulation")

    st.subheader("Investment vs. Projected Gain")
    fig = go.Figure(data=[
        go.Bar(name='Investment', x=roi_df['Segment'], y=roi_df['Investment'], marker_color='#FFA15A'),
        go.Bar(name='Projected Gain', x=roi_df['Segment'], y=roi_df['Projected_Gain'], marker_color='#19D3F3')
    ])
    fig.update_layout(barmode='group', xaxis_title="Segment", yaxis_title="USD ($)")
    st.plotly_chart(fig, use_container_width=True)

    st.markdown("---")
    st.subheader("Segment Deep Dive")
    selected_segment = st.selectbox("Select Segment for Detailed Metrics", roi_df['Segment'].unique())
    seg_data = roi_df[roi_df['Segment'] == selected_segment].iloc[0]

    c1, c2, c3 = st.columns(3)
    c1.metric("ROI", f"{seg_data['ROI']:.2f}x")
    c2.metric("Break-Even Target", f"${seg_data['BreakEven_Revenue']:,.2f}")
    c3.metric("Efficiency Score", f"{(seg_data['Projected_Gain']/seg_data['Investment']):.1f}")
```