```python
# streamlit_app.py

import streamlit as st
import matplotlib.pyplot as plt
from genovate_backend import load_data, train_model, predict_method, find_pam_sites

# Load and train model
data = load_data()
model, le_mut, le_org, le_method = train_model(data)

# Streamlit app setup
st.set_page_config(page_title="Genovate: CRISPR Delivery Predictor", layout="centered")
st.title("🧬 Genovate: CRISPR/Cas9 Delivery Simulation")
st.markdown("""
Welcome to **Genovate**, a predictive simulation tool to identify the optimal CRISPR delivery method
for treating gene mutations like **PKD1**, **PKD2**, and **PKHD1**.
""")

# Input section
st.header("1️⃣ Input Your Case")
organ = st.selectbox("Select Target Organ:", ["Kidney", "Liver"])
mutation = st.selectbox("Select Gene Mutation:", ["PKD1", "PKD2", "PKHD1"])
therapy_type = st.radio("Therapy Type:", ["Ex vivo", "In vivo"])

st.subheader("Clinical Parameters")
eff = st.slider("Estimated Editing Efficiency (%)", 60, 100, 75) / 100.0
off = st.slider("Estimated Off-target Risk (%)", 0, 20, 9) / 100.0
viability = st.slider("Cell Viability Post-Delivery (%)", 50, 100, 90) / 100.0
cost = st.select_slider("Cost & Scalability (1=Low Cost, 5=High Cost)", options=[1, 2, 3, 4, 5], value=3)

if st.button("🔍 Predict Best Delivery Method"):
    recommendation = predict_method(model, le_mut, le_org, le_method, mutation, organ, eff, off, viability, cost)
    st.success(f"🚀 Recommended Delivery Method: **{recommendation}**")

    # Display a basic bar chart comparing values
    methods = ["Lipid Nanoparticles", "Electroporation"]
    if recommendation == "LNP":
        values = [eff, 0.85]
        risks = [off, 0.12]
        viability_vals = [viability, 0.75]
    else:
        values = [0.72, eff]
        risks = [0.07, off]
        viability_vals = [0.92, viability]

    st.subheader("📊 Comparison Chart")
    fig, ax = plt.subplots()
    bar_width = 0.25
    x = range(len(methods))
    ax.bar(x, values, bar_width, label="Efficiency")
    ax.bar([i + bar_width for i in x], risks, bar_width, label="Off-Target Risk")
    ax.bar([i + 2*bar_width for i in x], viability_vals, bar_width, label="Viability")
    ax.set_xticks([i + bar_width for i in x])
    ax.set_xticklabels(methods)
    ax.set_ylim(0, 1.2)
    ax.legend()
    st.pyplot(fig)

# PAM Finder
st.header("🧬 Optional: Find PAM Sequences in Your DNA")
dna_input = st.text_area("Enter a DNA sequence (use only A, T, G, C):",
"AGGTCGTTACCGGTAGCGGTACCGTAGGGTAGGCTAGGGTACCGGTAG")
if st.button("🔎 Find PAM Sites"):
    pam_sites = find_pam_sites(dna_input.upper())
    if pam_sites:
        st.success(f"✅ Found {len(pam_sites)} PAM site(s). First 10:")
        st.write(pam_sites[:10])
    else:
        st.warning("❌ No PAM sites (NGG) found in the input sequence.")

# Footer
st.markdown("---")
st.caption("Developed by Raksheet Gummakonda for Genovate")
```