SAMPLE CODE

import streamlit as st

import google.generativeai as genai

import openai

import json

import time

from datetime import datetime

import plotly.graph\_objects as go

import plotly.express as px

import pandas as pd

import math

# Configure your API keys

GEMINI\_API\_KEY = "AIzaSyDyDevueMFuFYoZ8QC798a\_fwB4jZS9DPQ"

OPENAI\_API\_KEY = "sk-proj-mdupEQoktuRtkW8XsicmRGsCOtYvIuIV8xNY4o1Ia59INuTX8anmE40It\_iYQTBBHZRcKm24h4T3BlbkFJayejM-4Vq\_ci5km75t\_cjfMlio03kHeQaSf43qKwDWCr3xgIZw6GRnGJ1TBAeAY40Cbf3cfGcA"

genai.configure(api\_key=GEMINI\_API\_KEY)

openai.api\_key = OPENAI\_API\_KEY

# Page configuration

st.set\_page\_config(

page\_title="🌿 Eco-Design Carbon Analyzer",

page\_icon="🌱",

layout="wide",

initial\_sidebar\_state="expanded"

)

# Initialize session state

if 'design\_data' not in st.session\_state:

st.session\_state.design\_data = None

# Custom CSS

st.markdown("""

<style>

.main-header {

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

padding: 2rem;

border-radius: 15px;

margin-bottom: 2rem;

text-align: center;

color: white;

box-shadow: 0 4px 6px rgba(0,0,0,0.1);

}

.carbon-card {

background: linear-gradient(135deg, #ff6b6b, #ee5a6f);

padding: 1.5rem;

border-radius: 12px;

color: white;

margin: 1rem 0;

box-shadow: 0 4px 8px rgba(0,0,0,0.15);

}

.benefit-card {

background: linear-gradient(135deg, #4ecdc4, #44a08d);

padding: 1.5rem;

border-radius: 12px;

color: white;

margin: 1rem 0;

box-shadow: 0 4px 8px rgba(0,0,0,0.15);

}

.material-card {

background: linear-gradient(135deg, #f093fb, #f5576c);

padding: 1.5rem;

border-radius: 12px;

color: white;

margin: 1rem 0;

box-shadow: 0 4px 8px rgba(0,0,0,0.15);

}

.design-card {

background: linear-gradient(135deg, #ffecd2, #fcb69f);

padding: 1.5rem;

border-radius: 12px;

color: #333;

margin: 1rem 0;

box-shadow: 0 4px 8px rgba(0,0,0,0.15);

}

.metric-box {

background: white;

padding: 1rem;

border-radius: 8px;

border-left: 4px solid #4CAF50;

margin: 0.5rem 0;

box-shadow: 0 2px 4px rgba(0,0,0,0.1);

}

.success-message {

background: linear-gradient(135deg, #00c851, #007e33);

padding: 1rem;

border-radius: 8px;

color: white;

margin: 1rem 0;

text-align: center;

font-weight: bold;

}

.warning-message {

background: linear-gradient(135deg, #ffbb33, #ff8800);

padding: 1rem;

border-radius: 8px;

color: white;

margin: 1rem 0;

text-align: center;

font-weight: bold;

}

</style>

""", unsafe\_allow\_html=True)

# Header

st.markdown("""

<div class="main-header">

<h1>🌿 Eco-Product Carbon & Design Analyzer</h1>

<p>Analyze Carbon Footprint • Material Benefits • Design Process • 2D Diagrams</p>

</div>

""", unsafe\_allow\_html=True)

# Sidebar

with st.sidebar:

st.header("🎯 Analysis Settings")

analysis\_depth = st.selectbox(

"📊 Analysis Depth",

["Standard", "Detailed", "Comprehensive"]

)

include\_comparisons = st.checkbox("📈 Include Traditional Product Comparison", True)

include\_lifecycle = st.checkbox("🔄 Include Lifecycle Analysis", True)

include\_cost\_analysis = st.checkbox("💰 Include Cost Analysis", True)

st.divider()

st.markdown("### 🌍 Carbon Calculation Factors")

production\_factor = st.slider("🏭 Production Impact", 0.5, 2.0, 1.0, 0.1)

transport\_factor = st.slider("🚚 Transport Impact", 0.5, 2.0, 1.0, 0.1)

usage\_factor = st.slider("🔋 Usage Impact", 0.5, 2.0, 1.0, 0.1)

# Main content

st.header("💡 Product Input & Generation")

col1, col2 = st.columns([2, 1])

with col1:

product\_idea = st.text\_input(

"🌟 Enter Product Idea",

placeholder="e.g., bamboo smartphone case, solar water bottle, recycled plastic chair...",

help="Describe the eco-friendly product you want to analyze"

)

product\_category = st.selectbox(

"📋 Product Category",

["Electronics", "Home & Garden", "Fashion & Accessories", "Food & Beverage", "Transportation", "Personal Care", "Office Supplies"]

)

with col2:

st.markdown("### 🎯 Quick Stats")

if st.session\_state.design\_data:

st.metric("🌱 Carbon Reduction", f"{st.session\_state.design\_data.get('carbon\_reduction', 0)}%")

st.metric("♻ Recyclability", st.session\_state.design\_data.get('recyclability', 'N/A'))

st.metric("💚 Eco Score", f"{st.session\_state.design\_data.get('eco\_score', 0)}/100")

# Generate Analysis Button

if st.button("🔍 Generate Complete Analysis", type="primary", use\_container\_width=True):

if product\_idea:

with st.spinner("🔄 Analyzing carbon footprint, materials, and generating design..."):

try:

model = genai.GenerativeModel("models/gemini-1.5-flash")

# Enhanced prompt for specific analysis

analysis\_prompt = f"""

Analyze this eco-friendly product: {product\_idea} (Category: {product\_category})

Provide a comprehensive analysis with EXACTLY these sections:

1. CARBON FOOTPRINT ANALYSIS:

- Traditional product carbon footprint (kg CO2)

- Eco-friendly version carbon footprint (kg CO2)

- Carbon reduction percentage

- Lifecycle carbon analysis (production, transport, usage, disposal)

- Carbon offset potential

- Specific emission sources and reductions

2. ENVIRONMENTAL BENEFITS:

- Primary environmental benefits (list 5-7 key benefits)

- Quantified impact reductions (water, energy, waste)

- Biodiversity protection aspects

- Pollution reduction metrics

- Long-term environmental advantages

3. SUSTAINABLE MATERIALS:

- Primary materials used (with percentages)

- Material sourcing and sustainability

- Renewable vs non-renewable content

- Recyclability and end-of-life options

- Material innovation aspects

- Durability and longevity factors

4. DESIGN PROCESS:

- Step-by-step design methodology

- Key design principles applied

- Manufacturing process overview

- Quality control measures

- Assembly and construction details

- Innovation in design approach

5. TECHNICAL SPECIFICATIONS:

- Dimensions and weight

- Performance characteristics

- Durability specifications

- Maintenance requirements

- Safety features

6. 2D DIAGRAM DESCRIPTION:

- Detailed visual layout description

- Component placement and relationships

- Cross-sectional views

- Exploded view components

- Technical drawing specifications

- Annotation details for diagram

Make all data specific and quantified where possible. Include realistic numbers and percentages.

"""

response = model.generate\_content(analysis\_prompt)

analysis\_text = response.text

# Parse the response to extract specific data

sections = analysis\_text.split('\n\n')

# Extract carbon data

carbon\_reduction = 45 # Default value

recyclability = "High"

eco\_score = 78

# Try to extract actual values from the response

if 'carbon reduction' in analysis\_text.lower():

import re

carbon\_match = re.search(r'(\d+)%.carbon reduction|carbon reduction.?(\d+)%', analysis\_text.lower())

if carbon\_match:

carbon\_reduction = int(carbon\_match.group(1) or carbon\_match.group(2))

# Store analysis data

st.session\_state.design\_data = {

'product': product\_idea,

'category': product\_category,

'analysis': analysis\_text,

'carbon\_reduction': carbon\_reduction,

'recyclability': recyclability,

'eco\_score': eco\_score,

'timestamp': datetime.now()

}

# Display results in organized tabs

tab1, tab2, tab3, tab4, tab5 = st.tabs([

"🌍 Carbon Analysis",

"💚 Benefits",

"🧱 Materials",

"🔧 Design Process",

"📐 2D Diagrams"

])

with tab1:

st.markdown('<div class="carbon-card">', unsafe\_allow\_html=True)

st.markdown("## 🌍 Carbon Footprint Analysis")

# Extract carbon section

carbon\_section = ""

for section in sections:

if "CARBON FOOTPRINT" in section.upper() or "CARBON" in section.upper():

carbon\_section = section

break

if carbon\_section:

st.markdown(carbon\_section)

else:

st.markdown("### Carbon Impact Analysis")

st.markdown(analysis\_text.split('2. ENVIRONMENTAL')[0] if '2. ENVIRONMENTAL' in analysis\_text else analysis\_text[:800])

st.markdown('</div>', unsafe\_allow\_html=True)

# Carbon comparison chart

st.markdown("### 📊 Carbon Footprint Comparison")

carbon\_data = {

'Stage': ['Production', 'Transport', 'Usage', 'Disposal', 'Total'],

'Traditional Product (kg CO2)': [

25 \* production\_factor,

8 \* transport\_factor,

40 \* usage\_factor,

12,

85 \* (production\_factor + transport\_factor + usage\_factor) / 3

],

'Eco-Friendly Product (kg CO2)': [

12 \* production\_factor,

5 \* transport\_factor,

15 \* usage\_factor,

3,

35 \* (production\_factor + transport\_factor + usage\_factor) / 3

]

}

df\_carbon = pd.DataFrame(carbon\_data)

fig\_carbon = px.bar(

df\_carbon.iloc[:-1], # Exclude total row for cleaner chart

x='Stage',

y=['Traditional Product (kg CO2)', 'Eco-Friendly Product (kg CO2)'],

title="🌍 Carbon Emissions by Lifecycle Stage",

color\_discrete\_sequence=['#ff6b6b', '#4ecdc4'],

barmode='group'

)

fig\_carbon.update\_layout(height=400)

st.plotly\_chart(fig\_carbon, use\_container\_width=True)

# Carbon savings metrics

col\_c1, col\_c2, col\_c3, col\_c4 = st.columns(4)

with col\_c1:

st.markdown('<div class="metric-box">', unsafe\_allow\_html=True)

st.metric("🔥 CO2 Saved", f"{carbon\_reduction}%")

st.markdown('</div>', unsafe\_allow\_html=True)

with col\_c2:

st.markdown('<div class="metric-box">', unsafe\_allow\_html=True)

st.metric("🌱 Carbon Offset", f"{carbon\_reduction \* 0.8:.1f} kg/year")

st.markdown('</div>', unsafe\_allow\_html=True)

with col\_c3:

st.markdown('<div class="metric-box">', unsafe\_allow\_html=True)

st.metric("⚡ Energy Saved", f"{carbon\_reduction \* 1.2:.1f}%")

st.markdown('</div>', unsafe\_allow\_html=True)

with col\_c4:

st.markdown('<div class="metric-box">', unsafe\_allow\_html=True)

st.metric("🌊 Water Saved", f"{carbon\_reduction \* 0.9:.1f}%")

st.markdown('</div>', unsafe\_allow\_html=True)

with tab2:

st.markdown('<div class="benefit-card">', unsafe\_allow\_html=True)

st.markdown("## 💚 Environmental Benefits")

# Extract benefits section

benefits\_section = ""

for section in sections:

if "ENVIRONMENTAL BENEFITS" in section.upper() or "BENEFITS" in section.upper():

benefits\_section = section

break

if benefits\_section:

st.markdown(benefits\_section)

else:

# Extract benefits from the analysis

if '2. ENVIRONMENTAL' in analysis\_text:

benefits\_section = analysis\_text.split('2. ENVIRONMENTAL')[1].split('3.')[0] if '3.' in analysis\_text else analysis\_text.split('2. ENVIRONMENTAL')[1]

st.markdown(benefits\_section)

st.markdown('</div>', unsafe\_allow\_html=True)

# Benefits visualization

st.markdown("### 🎯 Impact Metrics")

benefits\_data = {

'Benefit Category': ['Carbon Reduction', 'Water Conservation', 'Waste Reduction', 'Energy Efficiency', 'Biodiversity Protection', 'Pollution Prevention'],

'Impact Score': [carbon\_reduction, 65, 72, 58, 45, 68],

'Traditional Score': [0, 20, 15, 25, 10, 20]

}

df\_benefits = pd.DataFrame(benefits\_data)

fig\_benefits = px.bar(

df\_benefits,

x='Benefit Category',

y=['Impact Score', 'Traditional Score'],

title="💚 Environmental Impact Comparison",

color\_discrete\_sequence=['#4ecdc4', '#ff6b6b'],

barmode='group'

)

fig\_benefits.update\_layout(height=400)

st.plotly\_chart(fig\_benefits, use\_container\_width=True)

with tab3:

st.markdown('<div class="material-card">', unsafe\_allow\_html=True)

st.markdown("## 🧱 Sustainable Materials")

# Extract materials section

materials\_section = ""

for section in sections:

if "SUSTAINABLE MATERIALS" in section.upper() or "MATERIALS" in section.upper():

materials\_section = section

break

if materials\_section:

st.markdown(materials\_section)

else:

if '3. SUSTAINABLE' in analysis\_text:

materials\_section = analysis\_text.split('3. SUSTAINABLE')[1].split('4.')[0] if '4.' in analysis\_text else analysis\_text.split('3. SUSTAINABLE')[1]

st.markdown(materials\_section)

st.markdown('</div>', unsafe\_allow\_html=True)

# Materials composition chart

st.markdown("### 📊 Material Composition")

materials\_data = {

'Material': ['Recycled Plastic', 'Bamboo Fiber', 'Organic Cotton', 'Bioplastic', 'Recycled Metal', 'Other Sustainable'],

'Percentage': [35, 25, 15, 12, 8, 5],

'Sustainability Score': [85, 95, 80, 90, 75, 85]

}

df\_materials = pd.DataFrame(materials\_data)

col\_m1, col\_m2 = st.columns(2)

with col\_m1:

fig\_pie = px.pie(

df\_materials,

values='Percentage',

names='Material',

title="🥧 Material Composition",

color\_discrete\_sequence=px.colors.qualitative.Set3

)

st.plotly\_chart(fig\_pie, use\_container\_width=True)

with col\_m2:

fig\_sustain = px.bar(

df\_materials,

x='Material',

y='Sustainability Score',

title="🌱 Material Sustainability Scores",

color='Sustainability Score',

color\_continuous\_scale='Greens'

)

fig\_sustain.update\_layout(xaxis\_tickangle=-45)

st.plotly\_chart(fig\_sustain, use\_container\_width=True)

with tab4:

st.markdown('<div class="design-card">', unsafe\_allow\_html=True)

st.markdown("## 🔧 Design Process & Methodology")

# Extract design process section

design\_section = ""

for section in sections:

if "DESIGN PROCESS" in section.upper() or "METHODOLOGY" in section.upper():

design\_section = section

break

if design\_section:

st.markdown(design\_section)

else:

if '4. DESIGN' in analysis\_text:

design\_section = analysis\_text.split('4. DESIGN')[1].split('5.')[0] if '5.' in analysis\_text else analysis\_text.split('4. DESIGN')[1]

st.markdown(design\_section)

st.markdown('</div>', unsafe\_allow\_html=True)

# Design process flowchart

st.markdown("### 🔄 Design Process Flow")

process\_steps = [

"Concept & Research",

"Material Selection",

"Sustainable Design",

"Prototyping",

"Testing & Validation",

"Manufacturing",

"Quality Control",

"Distribution"

]

# Create a process flow diagram

fig\_process = go.Figure()

# Add process steps as connected nodes

x\_positions = list(range(len(process\_steps)))

y\_positions = [0] \* len(process\_steps)

# Add lines connecting the steps

for i in range(len(process\_steps) - 1):

fig\_process.add\_trace(go.Scatter(

x=[x\_positions[i], x\_positions[i+1]],

y=[y\_positions[i], y\_positions[i+1]],

mode='lines',

line=dict(color='#4CAF50', width=3),

showlegend=False

))

# Add process step nodes

fig\_process.add\_trace(go.Scatter(

x=x\_positions,

y=y\_positions,

mode='markers+text',

marker=dict(

size=60,

color=['#4CAF50', '#2196F3', '#FF9800', '#9C27B0', '#00BCD4', '#FFC107', '#795548', '#607D8B'],

opacity=0.8

),

text=process\_steps,

textposition="bottom center",

textfont=dict(size=10, color='black'),

showlegend=False

))

fig\_process.update\_layout(

title="🔄 Eco-Friendly Design Process",

xaxis=dict(showgrid=False, zeroline=False, showticklabels=False),

yaxis=dict(showgrid=False, zeroline=False, showticklabels=False),

height=300,

showlegend=False

)

st.plotly\_chart(fig\_process, use\_container\_width=True)

with tab5:

st.markdown("## 📐 2D Technical Diagrams")

# Extract 2D diagram description

diagram\_section = ""

for section in sections:

if "2D DIAGRAM" in section.upper() or "DIAGRAM" in section.upper():

diagram\_section = section

break

if diagram\_section:

st.markdown("### 📋 Diagram Specifications")

st.markdown(diagram\_section)

col\_d1, col\_d2 = st.columns(2)

with col\_d1:

st.markdown("### 🎨 AI-Generated Product Visualization")

# Generate 2D diagram using DALL-E

try:

diagram\_prompt = f"Technical 2D engineering diagram of {product\_idea}, clean line drawing, white background, detailed cross-section view, labeled components, professional technical illustration style, blueprint aesthetic"

dalle\_response = openai.Image.create(

prompt=diagram\_prompt,

n=1,

size="512x512"

)

image\_url = dalle\_response["data"][0]["url"]

st.image(image\_url, caption="Technical 2D Diagram", use\_column\_width=True)

except Exception as e:

st.error(f"Image generation failed: {e}")

# Fallback: Create a simple diagram using Plotly

st.info("Generating alternative diagram...")

fig\_diagram = go.Figure()

# Create a simple product outline

# Rectangle for main body

fig\_diagram.add\_shape(

type="rect",

x0=1, y0=1, x1=4, y1=3,

line=dict(color="black", width=2),

fillcolor="lightblue",

opacity=0.3

)

# Add components

components = [

{"name": "Eco Material Core", "x": 2.5, "y": 2, "color": "#4CAF50"},

{"name": "Recyclable Shell", "x": 1.5, "y": 2.5, "color": "#2196F3"},

{"name": "Sustainable Coating", "x": 3.5, "y": 1.5, "color": "#FF9800"},

]

for comp in components:

fig\_diagram.add\_trace(go.Scatter(

x=[comp["x"]],

y=[comp["y"]],

mode='markers+text',

marker=dict(size=20, color=comp["color"]),

text=comp["name"],

textposition="top center",

showlegend=False

))

fig\_diagram.update\_layout(

title=f"2D Diagram: {product\_idea}",

xaxis=dict(range=[0, 5], showgrid=True),

yaxis=dict(range=[0, 4], showgrid=True),

height=400,

showlegend=False

)

st.plotly\_chart(fig\_diagram, use\_container\_width=True)

with col\_d2:

st.markdown("### 🔍 Component Breakdown")

# Component analysis diagram

fig\_components = go.Figure()

# Create a hierarchical component structure

components = [

{"name": "Main Structure", "level": 0, "x": 2, "y": 3, "size": 40, "color": "#4CAF50"},

{"name": "Eco Shell", "level": 1, "x": 1, "y": 2, "size": 30, "color": "#2196F3"},

{"name": "Core Material", "level": 1, "x": 3, "y": 2, "size": 30, "color": "#FF9800"},

{"name": "Sustainable Coating", "level": 2, "x": 0.5, "y": 1, "size": 20, "color": "#9C27B0"},

{"name": "Recycled Components", "level": 2, "x": 1.5, "y": 1, "size": 20, "color": "#00BCD4"},

{"name": "Bio-based Elements", "level": 2, "x": 2.5, "y": 1, "size": 20, "color": "#FFC107"},

{"name": "Renewable Parts", "level": 2, "x": 3.5, "y": 1, "size": 20, "color": "#795548"},

]

# Add connection lines

connections = [

(2, 3, 1, 2), (2, 3, 3, 2), # Main to level 1

(1, 2, 0.5, 1), (1, 2, 1.5, 1), # Shell to level 2

(3, 2, 2.5, 1), (3, 2, 3.5, 1), # Core to level 2

]

for x1, y1, x2, y2 in connections:

fig\_components.add\_trace(go.Scatter(

x=[x1, x2], y=[y1, y2],

mode='lines',

line=dict(color='gray', width=1),

showlegend=False

))

# Add component nodes

for comp in components:

fig\_components.add\_trace(go.Scatter(

x=[comp["x"]], y=[comp["y"]],

mode='markers+text',

marker=dict(size=comp["size"], color=comp["color"], opacity=0.7),

text=comp["name"],

textposition="middle center",

textfont=dict(size=8, color='white'),

showlegend=False

))

fig\_components.update\_layout(

title="🔧 Component Hierarchy",

xaxis=dict(range=[-0.5, 4], showgrid=False, showticklabels=False),

yaxis=dict(range=[0.5, 3.5], showgrid=False, showticklabels=False),

height=400,

showlegend=False

)

st.plotly\_chart(fig\_components, use\_container\_width=True)

# Technical specifications table

st.markdown("### 📊 Technical Specifications")

specs\_data = {

'Specification': ['Dimensions', 'Weight', 'Material Composition', 'Recyclability', 'Durability', 'Energy Efficiency'],

'Value': ['25x15x5 cm', '150g', '85% Sustainable', '95% Recyclable', '10+ years', f'{carbon\_reduction}% improved'],

'Standard Comparison': ['30x20x8 cm', '300g', '20% Sustainable', '30% Recyclable', '3-5 years', 'Baseline']

}

df\_specs = pd.DataFrame(specs\_data)

st.dataframe(df\_specs, use\_container\_width=True)

st.markdown('<div class="success-message">✅ Complete Analysis Generated Successfully!</div>', unsafe\_allow\_html=True)

except Exception as e:

st.error(f"❌ Error generating analysis: {e}")

st.info("💡 Please check your API keys and try again")

else:

st.markdown('<div class="warning-message">⚠ Please enter a product idea first!</div>', unsafe\_allow\_html=True)

# Export section

if st.session\_state.design\_data:

st.markdown("---")

st.header("📄 Export Analysis")

col\_exp1, col\_exp2, col\_exp3 = st.columns(3)

with col\_exp1:

# JSON export

export\_data = {

'product': st.session\_state.design\_data['product'],

'category': st.session\_state.design\_data['category'],

'carbon\_reduction': st.session\_state.design\_data['carbon\_reduction'],

'eco\_score': st.session\_state.design\_data['eco\_score'],

'recyclability': st.session\_state.design\_data['recyclability'],

'analysis': st.session\_state.design\_data['analysis'],

'generated\_date': st.session\_state.design\_data['timestamp'].isoformat()

}

with col\_exp2:

# PDF Export functionality

if st.button("📄 Export to PDF", use\_container\_width=True):

try:

from reportlab.lib.pagesizes import letter, A4

from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer

from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle

from reportlab.lib.units import inch

import io

buffer = io.BytesIO()

doc = SimpleDocTemplate(buffer, pagesize=A4)

styles = getSampleStyleSheet()

story = []

# Title

title\_style = ParagraphStyle(

'CustomTitle',

parent=styles['Heading1'],

fontSize=24,

spaceAfter=30,

textColor='darkblue'

)

story.append(Paragraph(f"🌿 Eco-Design Analysis Report", title\_style))

story.append(Spacer(1, 12))

# Product Info

story.append(Paragraph(f"<b>Product:</b> {st.session\_state.design\_data['product']}", styles['Normal']))

story.append(Paragraph(f"<b>Category:</b> {st.session\_state.design\_data['category']}", styles['Normal']))

story.append(Paragraph(f"<b>Analysis Date:</b> {st.session\_state.design\_data['timestamp'].strftime('%Y-%m-%d %H:%M')}", styles['Normal']))

story.append(Spacer(1, 12))

# Key Metrics

story.append(Paragraph("<b>Key Environmental Metrics:</b>", styles['Heading2']))

story.append(Paragraph(f"• Carbon Reduction: {st.session\_state.design\_data['carbon\_reduction']}%", styles['Normal']))

story.append(Paragraph(f"• Eco Score: {st.session\_state.design\_data['eco\_score']}/100", styles['Normal']))

story.append(Paragraph(f"• Recyclability: {st.session\_state.design\_data['recyclability']}", styles['Normal']))

story.append(Spacer(1, 12))

# Analysis Content

story.append(Paragraph("<b>Detailed Analysis:</b>", styles['Heading2']))

analysis\_text = st.session\_state.design\_data['analysis'].replace('\n', '<br/>')

story.append(Paragraph(analysis\_text, styles['Normal']))

doc.build(story)

buffer.seek(0)

st.download\_button(

label="📥 Download PDF Report",

data=buffer.getvalue(),

file\_name=f"eco\_analysis\_{st.session\_state.design\_data['product'].replace(' ', '\_')}.pdf",

mime="application/pdf"

)

except ImportError:

st.error("📄 PDF export requires reportlab. Install with: pip install reportlab")

except Exception as e:

st.error(f"PDF export failed: {e}")

with col\_exp3:

# CSV Export for data analysis

if st.button("📊 Export Data to CSV", use\_container\_width=True):

# Create comprehensive data for CSV export

export\_csv\_data = {

'Metric': [

'Product Name', 'Category', 'Carbon Reduction (%)',

'Eco Score', 'Recyclability', 'Analysis Date',

'Production Carbon (Traditional)', 'Production Carbon (Eco)',

'Transport Carbon (Traditional)', 'Transport Carbon (Eco)',

'Usage Carbon (Traditional)', 'Usage Carbon (Eco)',

'Total Carbon (Traditional)', 'Total Carbon (Eco)'

],

'Value': [

st.session\_state.design\_data['product'],

st.session\_state.design\_data['category'],

st.session\_state.design\_data['carbon\_reduction'],

st.session\_state.design\_data['eco\_score'],

st.session\_state.design\_data['recyclability'],

st.session\_state.design\_data['timestamp'].strftime('%Y-%m-%d %H:%M'),

f"{25 \* production\_factor:.1f} kg CO2",

f"{12 \* production\_factor:.1f} kg CO2",

f"{8 \* transport\_factor:.1f} kg CO2",

f"{5 \* transport\_factor:.1f} kg CO2",

f"{40 \* usage\_factor:.1f} kg CO2",

f"{15 \* usage\_factor:.1f} kg CO2",

f"{85 \* (production\_factor + transport\_factor + usage\_factor) / 3:.1f} kg CO2",

f"{35 \* (production\_factor + transport\_factor + usage\_factor) / 3:.1f} kg CO2"

]

}

df\_export = pd.DataFrame(export\_csv\_data)

csv\_buffer = df\_export.to\_csv(index=False)

st.download\_button(

label="📥 Download CSV Data",

data=csv\_buffer,

file\_name=f"eco\_data\_{st.session\_state.design\_data['product'].replace(' ', '\_')}.csv",

mime="text/csv"

)

# Analysis History Section

st.markdown("---")

st.header("📚 Analysis History")

if st.session\_state.analysis\_history:

for i, analysis in enumerate(reversed(st.session\_state.analysis\_history[-5:])): # Show last 5

with st.expander(f"🌱 {analysis['product']} - {analysis['timestamp'].strftime('%Y-%m-%d %H:%M')}"):

col\_h1, col\_h2, col\_h3 = st.columns(3)

with col\_h1:

st.metric("Carbon Reduction", f"{analysis['carbon\_reduction']}%")

with col\_h2:

st.metric("Eco Score", f"{analysis['eco\_score']}/100")

with col\_h3:

st.metric("Category", analysis['category'])

if st.button(f"🔄 Reload Analysis", key=f"reload\_{i}"):

st.session\_state.design\_data = analysis

st.experimental\_rerun()

else:

st.info("🌱 No previous analyses found. Generate your first eco-product analysis above!")

# Advanced Features Section

st.markdown("---")

st.header("🔬 Advanced Features")

col\_adv1, col\_adv2 = st.columns(2)

with col\_adv1:

st.subheader("🎯 Batch Analysis")

uploaded\_file = st.file\_uploader(

"📁 Upload CSV with Product Ideas",

type=['csv'],

help="Upload a CSV file with 'product\_idea' and 'category' columns for batch analysis"

)

if uploaded\_file is not None:

try:

batch\_df = pd.read\_csv(uploaded\_file)

st.write("📊 Preview of uploaded data:")

st.dataframe(batch\_df.head())

if st.button("🚀 Run Batch Analysis"):

batch\_results = []

progress\_bar = st.progress(0)

for idx, row in batch\_df.iterrows():

if 'product\_idea' in row and 'category' in row:

# Simulate analysis (replace with actual API calls)

progress\_bar.progress((idx + 1) / len(batch\_df))

# Mock results for demonstration

result = {

'product': row['product\_idea'],

'category': row['category'],

'carbon\_reduction': 30 + (idx \* 5) % 50,

'eco\_score': 60 + (idx \* 7) % 35,

'recyclability': ['High', 'Medium', 'Very High'][idx % 3]

}

batch\_results.append(result)

time.sleep(0.1) # Simulate processing time

# Display batch results

st.success(f"✅ Batch analysis completed for {len(batch\_results)} products!")

results\_df = pd.DataFrame(batch\_results)

st.dataframe(results\_df)

# Download batch results

csv\_results = results\_df.to\_csv(index=False)

st.download\_button(

"📥 Download Batch Results",

csv\_results,

"batch\_eco\_analysis.csv",

"text/csv"

)

except Exception as e:

st.error(f"Error processing file: {e}")

with col\_adv2:

st.subheader("🎨 Custom Visualization")

if st.session\_state.design\_data:

viz\_type = st.selectbox(

"📊 Choose Visualization Type",

["Carbon Footprint Radar", "Sustainability Scorecard", "Material Impact Bubble Chart", "Lifecycle Assessment Flow"]

)

if viz\_type == "Carbon Footprint Radar":

# Create radar chart

categories = ['Production', 'Transport', 'Usage', 'Disposal', 'Recycling', 'Innovation']

traditional\_values = [85, 70, 90, 60, 20, 30]

eco\_values = [45, 35, 40, 85, 95, 90]

fig\_radar = go.Figure()

fig\_radar.add\_trace(go.Scatterpolar(

r=traditional\_values,

theta=categories,

fill='toself',

name='Traditional Product',

line\_color='red',

fillcolor='rgba(255, 0, 0, 0.1)'

))

fig\_radar.add\_trace(go.Scatterpolar(

r=eco\_values,

theta=categories,

fill='toself',

name='Eco-Friendly Product',

line\_color='green',

fillcolor='rgba(0, 255, 0, 0.1)'

))

fig\_radar.update\_layout(

polar=dict(

radialaxis=dict(

visible=True,

range=[0, 100]

)),

showlegend=True,

title="🎯 Environmental Impact Radar Chart"

)

st.plotly\_chart(fig\_radar, use\_container\_width=True)

elif viz\_type == "Sustainability Scorecard":

# Create comprehensive scorecard

scorecard\_data = {

'Category': ['Carbon Footprint', 'Material Sustainability', 'Energy Efficiency',

'Water Conservation', 'Waste Reduction', 'Social Impact', 'Innovation'],

'Score': [st.session\_state.design\_data['carbon\_reduction'], 85, 78, 82, 75, 70, 88],

'Target': [80, 90, 85, 85, 80, 75, 85],

'Industry Average': [45, 60, 55, 50, 45, 40, 50]

}

df\_scorecard = pd.DataFrame(scorecard\_data)

fig\_scorecard = go.Figure()

fig\_scorecard.add\_trace(go.Bar(

name='Current Score',

x=df\_scorecard['Category'],

y=df\_scorecard['Score'],

marker\_color='lightgreen'

))

fig\_scorecard.add\_trace(go.Bar(

name='Target',

x=df\_scorecard['Category'],

y=df\_scorecard['Target'],

marker\_color='gold'

))

fig\_scorecard.add\_trace(go.Bar(

name='Industry Average',

x=df\_scorecard['Category'],

y=df\_scorecard['Industry Average'],

marker\_color='lightcoral'

))

fig\_scorecard.update\_layout(

title='🏆 Sustainability Scorecard',

xaxis\_tickangle=-45,

barmode='group',

height=500

)

st.plotly\_chart(fig\_scorecard, use\_container\_width=True)

# Collaboration Features

st.markdown("---")

st.header("🤝 Collaboration & Sharing")

col\_collab1, col\_collab2 = st.columns(2)

with col\_collab1:

st.subheader("💬 Analysis Comments")

if 'comments' not in st.session\_state:

st.session\_state.comments = []

new\_comment = st.text\_area("💭 Add your insights or suggestions:", height=100)

if st.button("💬 Add Comment"):

if new\_comment:

comment\_data = {

'text': new\_comment,

'timestamp': datetime.now(),

'user': 'Analyst' # In real app, get from authentication

}

st.session\_state.comments.append(comment\_data)

st.success("✅ Comment added!")

# Display comments

if st.session\_state.comments:

st.markdown("### 💬 Previous Comments")

for comment in reversed(st.session\_state.comments[-5:]):

st.markdown(f"""

<div style="background-color: #f0f0f0; padding: 10px; border-radius: 5px; margin: 5px 0;">

<small><b>{comment['user']}</b> - {comment['timestamp'].strftime('%Y-%m-%d %H:%M')}</small><br>

{comment['text']}

</div>

""", unsafe\_allow\_html=True)

with col\_collab2:

st.subheader("🔗 Share Analysis")

if st.session\_state.design\_data:

# Generate shareable link (mock)

share\_id = f"eco\_analysis\_{hash(st.session\_state.design\_data['product'])}"

share\_url = f"https://eco-analyzer.app/shared/{share\_id}"

st.code(share\_url, language=None)

if st.button("📋 Copy Share Link"):

st.success("🔗 Link copied to clipboard! (simulated)")

# Social sharing buttons (mock)

col\_s1, col\_s2, col\_s3 = st.columns(3)

with col\_s1:

st.button("📘 Share on LinkedIn")

with col\_s2:

st.button("🐦 Share on Twitter")

with col\_s3:

st.button("📧 Email Report")

# Footer

st.markdown("---")

st.markdown("""

<div style="text-align: center; padding: 2rem; background: linear-gradient(135deg, #667eea 0%, #764ba2 100%); border-radius: 10px; color: white; margin-top: 2rem;">

<h3>🌿 Eco-Design Carbon Analyzer</h3>

<p>Making sustainable design decisions through data-driven analysis</p>

<p><small>Built with ❤ for a greener future | Powered by AI & Environmental Science</small></p>

</div>

""", unsafe\_allow\_html=True)