

```
import streamlit as st
import numpy as np
import cv2
from PIL import Image
import pillow_heif

# Page Setup
st.set_page_config(page_title="Image Transformer", layout="wide")

# Theme
st.markdown("""
<style>
    .stApp {
        background: radial-gradient(circle at 20% 20%, #1a1a1f, #0c0c0f 90%);
        font-family: 'Segoe UI', sans-serif;
        color: #e6e6fa;
    }

    h1 {
        text-align: center;
        font-size: 2.5rem;
        color: #c084fc;
        text-shadow: 0 0 15px #c084fc;
    }

    h3, h4, label, p {
        color: #d8b4fe;
    }

    /* Glowing Button */
    div[data-testid="stButton"] > button {
        background-color: #7e22ce;
        color: white;
        font-weight: 600;
        border: none;
        border-radius: 12px;
        padding: 12px 25px;
        box-shadow: 0 0 15px #a855f7;
        transition: 0.3s;
    }

    div[data-testid="stButton"] > button:hover {

```

```

        background-color: #9333ea;
        box-shadow: 0 0 25px #c084fc;
        transform: scale(1.05);
    }

.img-box {
    background-color: rgba(255,255,255,0.03);
    padding: 10px;
    border-radius: 10px;
    box-shadow: 0 0 12px rgba(192,132,252,0.08);
}

hr { border: 1px solid #a855f7; margin-top: 25px; margin-bottom: 25px; }

img {
    max-width: 85% !important;
    height: auto !important;
    display: block;
    margin-left: auto;
    margin-right: auto;
    border-radius: 10px;
}

```

</style>

```

"""", unsafe_allow_html=True)

# HEADER
st.markdown("<h1>Image Transformer Studio</h1>", unsafe_allow_html=True)
st.write("Upload your image → Convert to Grayscale → Apply transformations like rotation, scaling, and translation.")

# Upload
uploaded_file = st.file_uploader("🌟 Upload your image (JPG, PNG, HEIC)", type=["jpg", "jpeg", "png", "heic"])

if "gray_resized" not in st.session_state:
    st.session_state.gray_resized = None

if uploaded_file:
    filename = uploaded_file.name.lower()
    if filename.endswith(".heic"):
        heif_file = pillow_heif.read_heif(uploaded_file.read())
        image = Image.frombytes(heif_file.mode, heif_file.size, heif_file.data)

```

```

else:
    image = Image.open(uploaded_file)

    st.session_state.original_image = image

# Three-column layout: Original | Button | Grayscale
col1, col2, col3 = st.columns([3, 1, 3])

with col1:
    st.subheader("🖼️ Original Image")
    st.image(image, caption="Original", width=450, clamp=True)

with col2:
    st.write("") # spacing
    st.write("") # spacing
    st.write("") # spacing
    if st.button("🎨 Convert to Grayscale"):
        img_np = np.array(image.convert("RGB"))
        gray = cv2.cvtColor(img_np, cv2.COLOR_RGB2GRAY)
        h, w = gray.shape
        new_w = 600
        new_h = int((new_w / w) * h)
        gray_resized = cv2.resize(gray, (new_w, new_h))
        st.session_state.gray_resized = gray_resized

with col3:
    if st.session_state.gray_resized is not None:
        st.subheader("之心 Grayscale Image")
        st.image(st.session_state.gray_resized, caption="Converted to Grayscale",
width=450, clamp=True)

# Transformations (only when grayscale exists)
if st.session_state.gray_resized is not None:
    gray_resized = st.session_state.gray_resized

    st.markdown("<hr>", unsafe_allow_html=True)
    st.subheader("⚙️ Choose Transformation Type")
    choice = st.radio(
        "Select operation:",
        ["Rotation", "Scaling", "Translation"],
        horizontal=True
    )

```

```

colA, colB = st.columns(2)

# ROTATION
if choice == "Rotation":
    angle = st.slider("Rotation Angle (degrees)", -360, 360, 0)
    (h, w) = gray_resized.shape
    center = (w // 2, h // 2)
    M = cv2.getRotationMatrix2D(center, angle, 1.0)
    rotated = cv2.warpAffine(gray_resized, M, (w, h))

    with colA:
        st.image(gray_resized, caption="Original", clamp=True, width=450)
    with colB:
        st.image(rotated, caption=f"Rotated ({angle}°)", clamp=True, width=450)

# SCALING (with separate X and Y sliders)
elif choice == "Scaling":
    sx = st.slider("Scale X (Width)", 0.1, 3.0, 1.0)
    sy = st.slider("Scale Y (Height)", 0.1, 3.0, 1.0)
    scaled = cv2.resize(gray_resized, None, fx=sx, fy=sy,
interpolation=cv2.INTER_AREA)

    with colA:
        st.image(gray_resized, caption="Original", clamp=True, width=450)
    with colB:
        st.image(scaled, caption=f"Scaled (X={sx}, Y={sy})", clamp=True,
width=450)

# TRANSLATION
elif choice == "Translation":
    tx = st.slider("Translate X (pixels)", -300, 300, 0)
    ty = st.slider("Translate Y (pixels)", -300, 300, 0)
    M = np.float32([[1, 0, tx], [0, 1, ty]])
    translated = cv2.warpAffine(gray_resized, M, (gray_resized.shape[1],
gray_resized.shape[0]))

    with colA:
        st.image(gray_resized, caption="Original", clamp=True, width=450)
    with colB:
        st.image(translated, caption=f"Translated (X={tx}, Y={ty})",
clamp=True, width=450)

```

```
else:  
    st.info("👉 Upload an image to begin transforming it!")
```