Imports

```python
import streamlit as st
import cv2
import numpy as np
from PIL import Image
import io
import os
```
- **Purpose:** Imports necessary modules for UI, image processing, array handling, image conversion, and I/O operations.

***

Streamlit Page Setup and Title

```python
st.set_page_config(layout="wide")
st.title("Image Processing & Analysis Toolkit")
```
- **Purpose:** Sets the page layout to wide and gives the app a title[1].

***

Sidebar: File Operations

```python
with st.sidebar:
    st.header("File")
    uploaded_file = st.file_uploader("Open: Upload an image", type=["jpg", "jpeg", "png", "bmp"])
    save_btn = st.button("Save Processed Image")
    exit_btn = st.button("Exit App")
```
- **Purpose:** Handles image upload, save, and exit operations from the sidebar[1].

***

Load Image

```python
if uploaded_file is not None:
    file_bytes = np.asarray(bytearray(uploaded_file.read()), dtype=np.uint8)
    image = cv2.imdecode(file_bytes, 1)
    orig_pil = Image.open(io.BytesIO(uploaded_file.getvalue()))
    file_format = orig_pil.format
    file_size = uploaded_file.size
    dpi = orig_pil.info.get("dpi", (72, 72))
    shape = image.shape
```
- **Purpose:** Loads and decodes the uploaded image, extracts metadata such as format, size, DPI, and shape[1].

***

Sidebar: Operation Selection and Parameters

```python
st.sidebar.header("Operations")
op_category = st.sidebar.selectbox(
    "Select Category",
    ("Image Info", "Color Conversions", "Transformations", "Filtering & Morphology", "Enhancement", "Edge Detection", "Compression")
)
operation = None
params = {}

# Category and individual operations, plus parameter selection, such as angles and scaling
# ... (contains multiple nested if/elif statements for operation selection and parameter sliders)
```
- **Purpose:** Lets the user choose a category and a specific image processing operation with adjustable parameters[1].

***

Image Processing Logic

```python
processed = image.copy()
info_text = ""

if operation == "info":
```

```
    # Gather image info for display
elif operation == "RGB ↔ BGR":
    processed = cv2.cvtColor(processed, cv2.COLOR_BGR2RGB)
# ... (similar blocks for each operation, handling conversion, transformations, filters, enhancement, edge detection, and compression)
```

- **Purpose:** Performs the selected operation on the image, storing results in the `processed` variable, or prepares metadata text[1].

***

File Compression and Download

```python
elif operation == "Save as JPG":
    _, buf = cv2.imencode('.jpg', processed)
    st.sidebar.download_button("Download JPG", buf.tobytes(), file_name="processed.jpg", mime="image/jpeg")
# ... (similar for PNG and BMP)
```

- **Purpose:** Allows saving the processed image in different file formats (JPG, PNG, BMP) via sidebar download buttons[1].

***

Display Area (Dual Panel)

```python
col1, col2 = st.columns(2)

with col1:
    st.subheader("Original Image")
    st.image(cv2.cvtColor(image, cv2.COLOR_BGR2RGB), use_column_width=True)

with col2:
    st.subheader("Processed Image")
    if operation == "info":
        st.info(info_text)
        st.image(cv2.cvtColor(image, cv2.COLOR_BGR2RGB), use_column_width=True)
    elif operation and len(processed.shape) == 2:
        st.image(processed, use_column_width=True, channels="GRAY")
    elif operation:
```

```
        st.image(cv2.cvtColor(processed, cv2.COLOR_BGR2RGB),
use_column_width=True)
    else:
        st.image(cv2.cvtColor(image, cv2.COLOR_BGR2RGB),
use_column_width=True)
```
- **Purpose:** Shows the original and processed images side by side in separate columns. Handles display for grayscale and colored outputs[1].

***

Status Bar and Metadata

```python
st.markdown("---")
st.markdown(
    f"**Dimensions:** {shape[1]} x {shape} "
    f"**Channels:** {shape if len(shape) == 3 else 1} "
    f"**DPI:** {dpi} "
    f"**Format:** {file_format} "
    f"**File Size:** {file_size/1024:.2f} KB"
)
```
- **Purpose:** Displays metadata about the image at the bottom of the app[1].

***

Exit Button and App End

```python
if exit_btn:
    st.stop()
else:
    st.info("Please upload an image to begin.")
.
```