Python project for your **Image Converter application** with all the requested features:

✅ Features Implemented:

- Convert between JPG, PNG, BMP, GIF, etc.
- Batch image conversion
- Resizing and compression options
- GUI with **Tkinter**
- Error handling and status updates

📁 Folder Structure

```
image_converter/
├── main.py
├── requirements.txt
```

📄 `requirements.txt`

```
Pillow
opencv-python
```

Install with:

```
pip install -r requirements.txt
```

🐍 `main.py` (Full Source Code)

```
import os
import tkinter as tk
from tkinter import filedialog, messagebox, ttk
from PIL import Image
import cv2

SUPPORTED_FORMATS = ['.jpg', '.jpeg', '.png', '.bmp', '.gif']

def is_supported_file(file_path):
    return any(file_path.lower().endswith(ext) for ext in SUPPORTED_FORMATS)

class ImageConverterApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Image Converter")
        self.root.geometry("500x500")
        self.image_paths = []
```

```
        # UI elements
        self.create_widgets()

    def create_widgets(self):
        ttk.Label(self.root, text="Image Converter", font=("Arial",
18)).pack(pady=10)

        # Add buttons
        ttk.Button(self.root, text="Select Images",
command=self.select_images).pack(pady=5)
        ttk.Button(self.root, text="Select Output Folder",
command=self.select_output_folder).pack(pady=5)

        # Output format
        ttk.Label(self.root, text="Select Output Format:").pack(pady=5)
        self.output_format = tk.StringVar()
        ttk.Combobox(self.root, textvariable=self.output_format,
values=['.jpg', '.png', '.bmp', '.gif'], state="readonly").pack()

        # Resize options
        ttk.Label(self.root, text="Resize (optional):").pack(pady=5)
        resize_frame = ttk.Frame(self.root)
        resize_frame.pack(pady=5)
        self.width_entry = ttk.Entry(resize_frame, width=10)
        self.width_entry.pack(side="left", padx=5)
        ttk.Label(resize_frame, text="x").pack(side="left")
        self.height_entry = ttk.Entry(resize_frame, width=10)
        self.height_entry.pack(side="left", padx=5)

        # Compression quality
        ttk.Label(self.root, text="Compression (1-100, JPEG
only):").pack(pady=5)
        self.quality_entry = ttk.Entry(self.root)
        self.quality_entry.pack()

        ttk.Button(self.root, text="Convert",
command=self.convert_images).pack(pady=20)

        self.status_text = tk.StringVar()
        ttk.Label(self.root, textvariable=self.status_text).pack(pady=5)

    def select_images(self):
        files = filedialog.askopenfilenames(title="Select Images",
filetypes=[("Image files", "*.jpg *.jpeg *.png *.bmp *.gif")])
        self.image_paths = list(files)
        self.status_text.set(f"{len(self.image_paths)} image(s) selected.")

    def select_output_folder(self):
        self.output_folder = filedialog.askdirectory(title="Select Output
Folder")
        self.status_text.set("Output folder selected.")

    def convert_images(self):
        if not self.image_paths:
            messagebox.showerror("Error", "No images selected!")
            return
        if not hasattr(self, 'output_folder') or not self.output_folder:
```

```
            messagebox.showerror("Error", "Please select output folder.")
            return
        if not self.output_format.get():
            messagebox.showerror("Error", "Please select output format.")
            return

        output_format = self.output_format.get()
        width = self.width_entry.get()
        height = self.height_entry.get()
        quality = self.quality_entry.get()

        try:
            width = int(width) if width else None
            height = int(height) if height else None
            quality = int(quality) if quality else 95
        except ValueError:
            messagebox.showerror("Error", "Invalid resize or quality
values.")
            return

        for img_path in self.image_paths:
            try:
                img = Image.open(img_path)

                # Resize
                if width and height:
                    img = img.resize((width, height), Image.ANTIALIAS)

                base_name = os.path.splitext(os.path.basename(img_path))[0]
                output_path = os.path.join(self.output_folder, base_name +
output_format)

                if output_format in ['.jpg', '.jpeg']:
                    img = img.convert("RGB")
                    img.save(output_path, quality=quality)
                else:
                    img.save(output_path)

            except Exception as e:
                messagebox.showerror("Error", f"Failed to convert
{img_path}\n{e}")
                continue

        self.status_text.set("Conversion completed successfully.")
        messagebox.showinfo("Success", "All images converted successfully!")

if __name__ == "__main__":
    root = tk.Tk()
    app = ImageConverterApp(root)
    root.mainloop()
```

🚀 How to Run the Project

1. Save the code in `main.py`

2. Create and activate a virtual environment (optional but recommended):
```
python -m venv venv
source venv/bin/activate  # or venv\Scripts\activate on Windows
```
5. Install required packages:
```
pip install -r requirements.txt
```
7. Run the app:
```
python main.py
```
