To create an automated paint application with a dialogue box, you can use Python's tkinter library. This will include essential features like a canvas for drawing, color selection, and options to save your work.

In [1]:
```
import tkinter as tk
from tkinter import colorchooser, filedialog, messagebox

class PaintApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Automated Paint Box")

        # Set up default values
        self.brush_color = "black"
        self.brush_size = 5

        # Create a canvas
        self.canvas = tk.Canvas(self.root, bg="white", width=800, height=600)
        self.canvas.pack(fill=tk.BOTH, expand=True)

        # Bind mouse events to canvas
        self.canvas.bind("<B1-Motion>", self.paint)

        # Create toolbar
        toolbar = tk.Frame(self.root, bg="lightgray", height=50)
        toolbar.pack(side=tk.TOP, fill=tk.X)

        # Add color picker button
        color_btn = tk.Button(toolbar, text="Choose Color", command=self.choose_color)
        color_btn.pack(side=tk.LEFT, padx=5, pady=5)

        # Add brush size selector
        size_label = tk.Label(toolbar, text="Brush Size:")
        size_label.pack(side=tk.LEFT, padx=5)
        self.size_slider = tk.Scale(toolbar, from_=1, to=20, orient=tk.HORIZONTAL)
        self.size_slider.set(self.brush_size)
        self.size_slider.pack(side=tk.LEFT, padx=5)
        # Add save button
        save_btn = tk.Button(toolbar, text="Save", command=self.save_drawing)
        save_btn.pack(side=tk.LEFT, padx=5, pady=5)

        # Add clear button
        clear_btn = tk.Button(toolbar, text="Clear", command=self.clear_canvas)
        clear_btn.pack(side=tk.LEFT, padx=5, pady=5)

        # Add exit button
        exit_btn = tk.Button(toolbar, text="Exit", command=self.exit_app)
        exit_btn.pack(side=tk.RIGHT, padx=5, pady=5)

    def paint(self, event):
        x1, y1 = (event.x - self.brush_size), (event.y - self.brush_size)
        x2, y2 = (event.x + self.brush_size), (event.y + self.brush_size)
        self.canvas.create_oval(x1, y1, x2, y2, fill=self.brush_color, outline=self.brush_color)

    def choose_color(self):
        color = colorchooser.askcolor()[1]
        if color:
            self.brush_color = color

    def save_drawing(self):
        file_path = filedialog.asksaveasfilename(defaultextension=".png", filetypes=[("PNG files", "*.png"), ("All files", "*.*")])
        if file_path:
            try:
                # Save the canvas as an image
                self.canvas.postscript(file="temp.ps", colormode='color')
                from PIL import Image
                img = Image.open("temp.ps")
                img.save(file_path, "png")
                messagebox.showinfo("Success", "Your drawing has been saved successfully!")
            except Exception as e:
                messagebox.showerror("Error", f"An error occurred while saving: {str(e)}")

    def clear_canvas(self):
        self.canvas.delete("all")

    def exit_app(self):
        if messagebox.askyesno("Exit", "Are you sure you want to exit?"):
            self.root.destroy()

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

How It Works: Canvas: The canvas allows you to draw using the mouse. Dragging the left mouse button () paints on the canvas. Color Picker: Click "Choose Color" to select a brush color using a color dialog. Brush Size: Use the slider to adjust the brush size. Save Functionality: Save your drawing as a .png image using the "Save" button. Clear Canvas: The "Clear" button erases everything on the canvas. Exit: Exit the application with confirmation.

```
In [ ]:
```