AccFileOpenner.py

```
        end_index = self.editing_files_text.index(tk.END)
        tag_name = f"tag_{file_path}"  # Use a unique tag name for each file
path
        self.editing_files_text.tag_add(tag_name, start_index, end_index)
        self.editing_files_text.tag_bind(tag_name, "<Button-1>", lambda e,
path=file_path: self.open_file_folder(path))
        self.editing_files_text.tag_bind(tag_name, "<Enter>", lambda e,
tag=tag_name: self.on_enter(e, tag))
        self.editing_files_text.tag_bind(tag_name, "<Leave>", lambda e,
tag=tag_name: self.on_leave(e, tag))

    def on_enter(self, event, tag):
        event.widget.config(cursor="cross")
        event.widget.tag_configure(tag, background="yellow", foreground="black")

    def on_leave(self, event, tag):
        event.widget.config(cursor="")
        event.widget.tag_configure(tag, background="", foreground="")

    def open_file_folder(self, file_path):
        folder_path = os.path.dirname(file_path)
        if os.path.exists(folder_path):
            os.startfile(folder_path)
        else:
            messagebox.showerror("Error", "Folder not found.")

    def create_finished_button(self, file_type):
        if self.finished_button:
            self.finished_button.destroy()
        self.finished_button = tk.Button(self.root,
                                         text=f"I am finished with this
{file_type} file, click to remove the [editing] marker file.",
                                         command=self.copy_back_to_original)
        self.finished_button.grid(row=4, column=0, columnspan=3, pady=10)

    def remove_finished_button(self):
        if self.finished_button:
            self.finished_button.destroy()
            self.finished_button = None

    def reset_all(self):
        self.original_file = None
        self.selected_file = ""
        self.lock_file = None
        self.finished_button = None
        self.clear_file_path_label()

@_Exe_Util.try_catch_error
def main():
    root = TkinterDnD.Tk()
    app = FileProcessorApp(root)
    root.mainloop()

if __name__ == "__main__":
    main()
```

gui.py

```
        self.dashboard_label = tk.Label(self.canvas, text="Drag and Drop a file
here or Click to Select a file", bg='#3e3e3e', fg='white', font=('Helvetica',
14, 'bold'))
```

```
        self.dashboard_label.place(relx=0.5, rely=0.1, anchor='n')  # Positioned
the label at the top

        self.file_path_label = tk.Label(self.canvas, text="", bg='#3e3e3e',
fg='white', font=('Helvetica', 12), wraplength=int(self.windowX * 0.8))
        self.file_path_label.place(relx=0.5, rely=0.5, anchor='center')

        self.dashboard_label.bind("<Button-1>", self.open_file_dialog)
        self.dashboard_frame.bind("<Button-1>", self.open_file_dialog)
        self.root.drop_target_register(DND_FILES)
        self.root.dnd_bind('<<Drop>>', self.handle_file_drop)

        self.dashboard_label.bind("<Enter>", self.change_cursor_to_hand)
        self.dashboard_label.bind("<Leave>", self.change_cursor_to_arrow)
        self.dashboard_frame.bind("<Enter>", self.change_cursor_to_hand)
        self.dashboard_frame.bind("<Leave>", self.change_cursor_to_arrow)

    def change_cursor_to_hand(self, event):
        return
        event.widget.config(cursor="hand2")

    def change_cursor_to_arrow(self, event):
        return
        event.widget.config(cursor="")

    def draw_rounded_rect(self, x1, y1, x2, y2, radius, **kwargs):
        # Draw the lines
        self.canvas.create_line(x1 + radius, y1, x2 - radius, y1, **kwargs)
        self.canvas.create_line(x2, y1 + radius, x2, y2 - radius, **kwargs)
        self.canvas.create_line(x2 - radius, y2, x1 + radius, y2, **kwargs)
        self.canvas.create_line(x1, y2 - radius, x1, y1 + radius, **kwargs)

        # Draw the arcs
        self.canvas.create_arc(x1, y1, x1 + 2 * radius, y1 + 2 * radius,
start=90, extent=90, style='arc', **kwargs)
        self.canvas.create_arc(x2 - 2 * radius, y1, x2, y1 + 2 * radius,
start=0, extent=90, style='arc', **kwargs)
        self.canvas.create_arc(x2 - 2 * radius, y2 - 2 * radius, x2, y2,
start=270, extent=90, style='arc', **kwargs)
        self.canvas.create_arc(x1, y2 - 2 * radius, x1 + 2 * radius, y2,
start=180, extent=90, style='arc', **kwargs)

    def update_title_with_days_left(self):
        days_left = (EXPIRATION_DATE - datetime.date.today()).days
        if days_left <= 0:
            self.root.title("ACC File Opener - Tool Expired")
        elif days_left <= 30:
            self.root.title(f"ACC File Opener - {days_left} days left")
        else:
            self.root.title("ACC File Opener")

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