Tkinter Widget Showcase Application Assignment

This assignment challenges you to create a Python application using Tkinter that demonstrates the functionality of a wide range of Tkinter widgets. The application should be interactive and showcase the unique features of each widget.

Objective:

To gain practical experience with Tkinter widget creation, layout management, event handling, and application development.

Requirements:

1. **Application Theme:** Choose a theme for your application (e.g., a simple calculator, a to-do list, a unit converter, a simple text editor, a generative AI based chatbot etc.). The theme should be appropriate for showcasing a variety of widgets.
2. **Widget Integration:** Your application must incorporate the following Tkinter widgets (at least one of each):
    - **Label:** Display static text or images.
    - **Entry:** Allow single-line text input.
    - **Text:** Allow multi-line text input and display.
    - **Button:** Trigger actions when clicked.
    - **Checkbutton:** Allow users to select one or more options.
    - **Radiobutton:** Allow users to select one option from a group.
    - **Combobox (ttk):** Provide a dropdown list of options.
    - **Listbox:** Display a list of selectable items.
    - **Scale:** Allow users to select a value within a range.
    - **Spinbox:** Allow users to select a value from a limited set.
    - **Progressbar (ttk):** Display the progress of an operation.
    - **Menu:** Create a menu bar with dropdown menus.
    - **Canvas:** Allow drawing shapes, images, and text.
    - **Frame:** Organize and group widgets.

    - **LabelFrame:** A frame with a label.
    - **Toplevel:** Create a new, independent window.
    - **Message:** Display a multi-line message (similar to a label, but handles wrapping).

3. **Functionality:**

The widgets should not just be present but also functional. For example:

- Buttons should perform actions.
- Entry and Text widgets should allow input and potentially display it elsewhere.
- Checkbuttons and Radiobuttons should store and reflect the user's choices.
- The Scale and Spinbox should affect some other element in the application.
- The Combobox and Listbox should allow item selection.
- The Menu should have working menu items.
- The Canvas can be used for simple drawing or displaying images related to the theme.
- Toplevel windows can be used for dialogs or secondary information displays.

4. **Layout:**

Use appropriate layout managers (e.g., pack,) to arrange the widgets in a clear and organized manner. Consider the user experience and make the application visually appealing.

5. **Event Handling:**

Implement event handlers to respond to user interactions (e.g., button clicks, changes in entry fields, selections in listboxes).

6. **Code Quality:**

Write clean, well-documented, and modular code. Use meaningful variable names and comments to explain the purpose of different code sections.

Deliverables:

- Python source code (.py file).

- A brief README file explaining the application's functionality, how to run it, and any special instructions. Include a screenshot of your application.

Grading Criteria:

- Completeness: All required widgets are used and functional.
- Functionality: The application performs the intended tasks effectively.
- Layout and User Interface: The application is well-organized and user-friendly.
- Code Quality: The code is clean, well-documented, and efficient.
- **The code must be yours and you should be able to explain the code and modify the code when asked.**