CustomTkinter Calculator Project

1. Introduction

This document provides an overview of the CustomTkinter-based Calculator application. It explains the user interface, features, underlying logic, and necessary steps to run the project.

2. Features and Functionality

- Modern Dark/Light themes powered by CustomTkinter.
- Dynamic display textbox that adapts font size to expression length.
- History panel that shows past calculations and can be cleared.
- Percentage (%) button with custom logic (e.g., 20 - 60% yields 8).
- Square root (√) function with error handling for negative inputs.
- ANS button to reuse the last calculated result.
- Keyboard shortcuts for faster input (Enter, Backspace, r, t, etc.).
- Automatic prevention of invalid operator sequences (e.g., "++").

3. User Interface Overview

The main window is 350×650 px by default (minimum: 350×650, maximum: 400×750). It contains a read-only display textbox, an optional history panel, three control buttons at the top (History, Clear, Theme), and a 4×6 grid of calculator buttons for numbers and operators.

4. Keyboard Shortcuts

Key	Function
0-9, +, -, *, /, (,)	Enter numbers and operators
Enter	Evaluate the current expression
Backspace	Delete the last character
Escape	Clear the entire expression
r	Calculate square root (√)
t	Toggle Dark/Light theme
%	Calculate percentage
^	Power operation

5. Calculation Logic

Expressions are sanitized and transformed before evaluation. Custom symbols are mapped to Python equivalents (e.g., '÷' → '/', '×' → '*', '^' → '**'). The eval() function computes the result, which is rounded to two decimal places where appropriate.

Percentage calculations apply contextual rules: for example, in '20 - 60%', 60 % of 20 is subtracted, producing 8.

6. Technical Stack

- Python 3.x
- CustomTkinter (modern wrapper around Tkinter)
- Tkinter for underlying GUI components
- Object-Oriented Programming (single CalculatorApp class)
- Exception handling for division by zero and invalid input

7. Error Handling

The application gracefully handles common errors such as division by zero, invalid percentage syntax, and attempts to take the square root of negative numbers. User-friendly messages are displayed instead of program crashes.

8. How to Run the Application

1. Install the CustomTkinter library:
   pip install customtkinter

2. Ensure the icon file path (calcu.ico) is valid, or adjust the path in the code.
3. Run the script using Python:
   python calculator.py

9. Conclusion

This project demonstrates how CustomTkinter can be employed to build a modern, user-friendly calculator with advanced features such as theme switching, history tracking, and extended mathematical functions. It serves as an effective showcase of GUI development and event handling in Python.