Project Report

Simple Python Calculator

Name: Mayank Sharma
Reg. No.: 25BCE10272

Contents

1   Introduction

The "Simple Python Calculator" is a desktop-based application developed to perform fundamental arithmetic operations. Built using the Python programming language and the Tkinter GUI toolkit, this project aims to provide a lightweight, efficient, and user-friendly alternative to complex calculation software. The application mimics the layout of standard physical calculators, ensuring immediate familiarity for users.

2   Problem Statement

In modern desktop environments, users often require a quick method to perform basic calculations without navigating through complex menus or waiting for resource-heavy applications to load. Web-based calculators require an internet connection, and scientific calculators often clutter the interface with unused functions. There is a need for a minimalist, instant-start application dedicated solely to basic arithmetic (Addition, Subtraction, Multiplication, Division).

3   Functional Requirements

The system must fulfill the following functional requirements:

- **Input Processing:** Accept numeric input (0-9) via on-screen buttons.
- **Arithmetic Operations:** Support addition (+), subtraction (-), multiplication (*), and division (/).
- **Evaluation:** Compute the result of the mathematical expression when the "=" button is pressed.
- **Clear Function:** Provide a "C" button to clear the current display and reset the internal state.
- **Error Handling:** Detect division by zero or invalid syntax and display a user-friendly "Error" message.

4   Non-functional Requirements

- **Usability:** The interface must be intuitive, using a standard grid layout.
- **Performance:** The application must launch instantly (under 1 second) and calculate results in real-time.
- **Reliability:** The application should not crash upon receiving invalid user input.
- **Portability:** The code must be runnable on any system with a standard Python 3 installation.

5   Design Diagrams

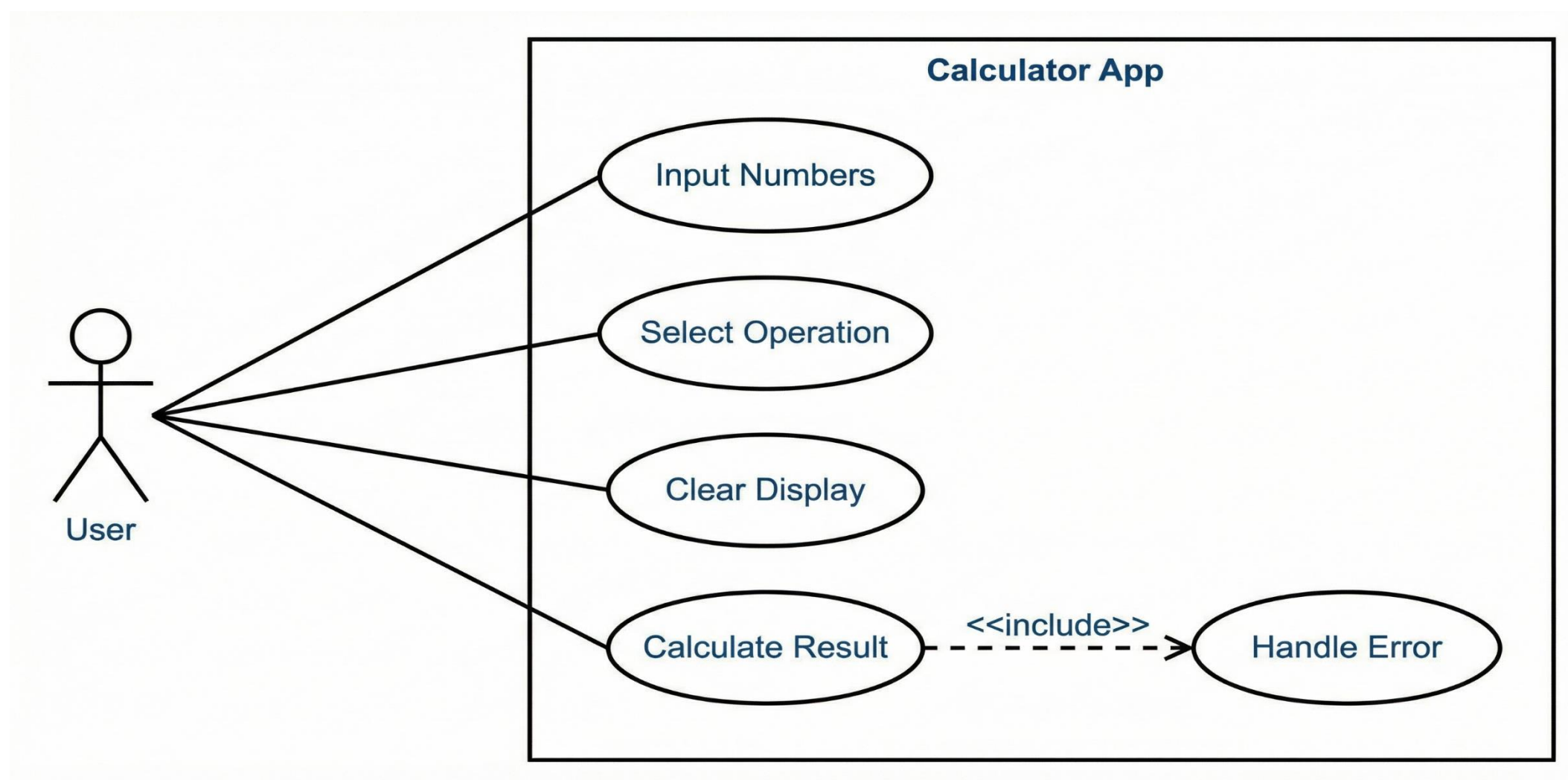## 5.1   Use Case Diagram

3

Figure 1: Use Case Diagram for Calculator Interaction
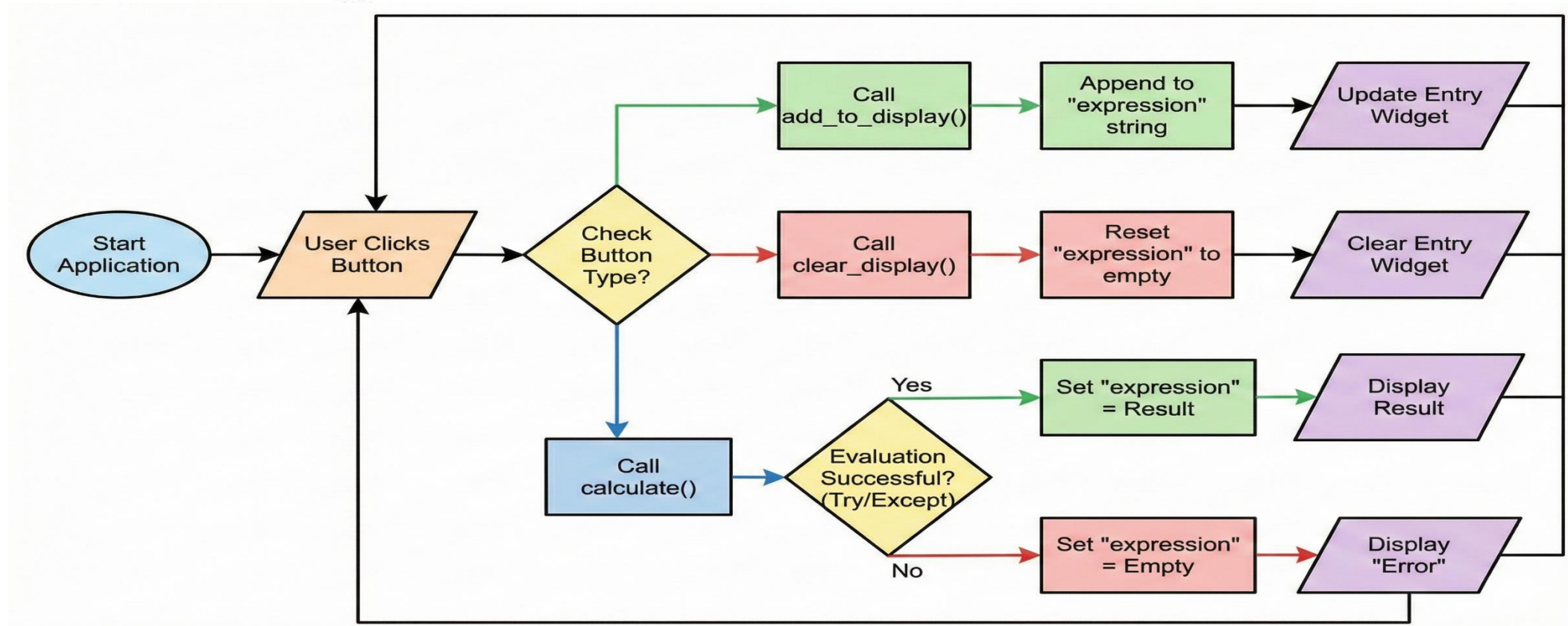
5.2 Workflow Diagram

Figure 2: Application Workflow

6 Implementation Details

The core logic revolves around a global string variable, expression, which accumulates key presses.

- **Global State:** A variable expression holds the current math string (e.g., "5+5").

- **Input Handling:** The add_to_display(value) function appends characters to the global string.

- **Calculation:** The calculate() function uses a try-except block to catch errors (like ZeroDivisionErro and updates the view.
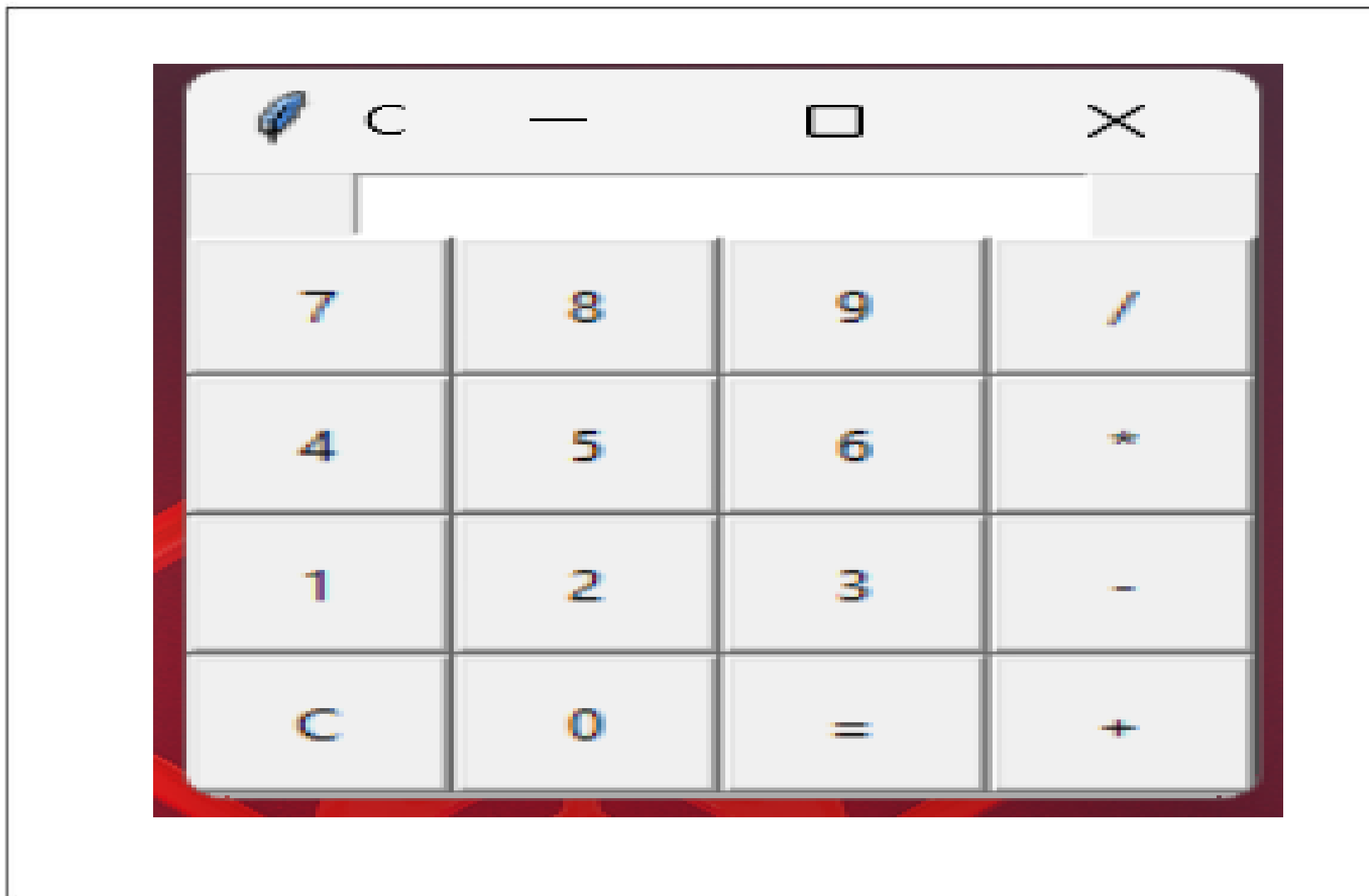
4

7 Screenshots / Results
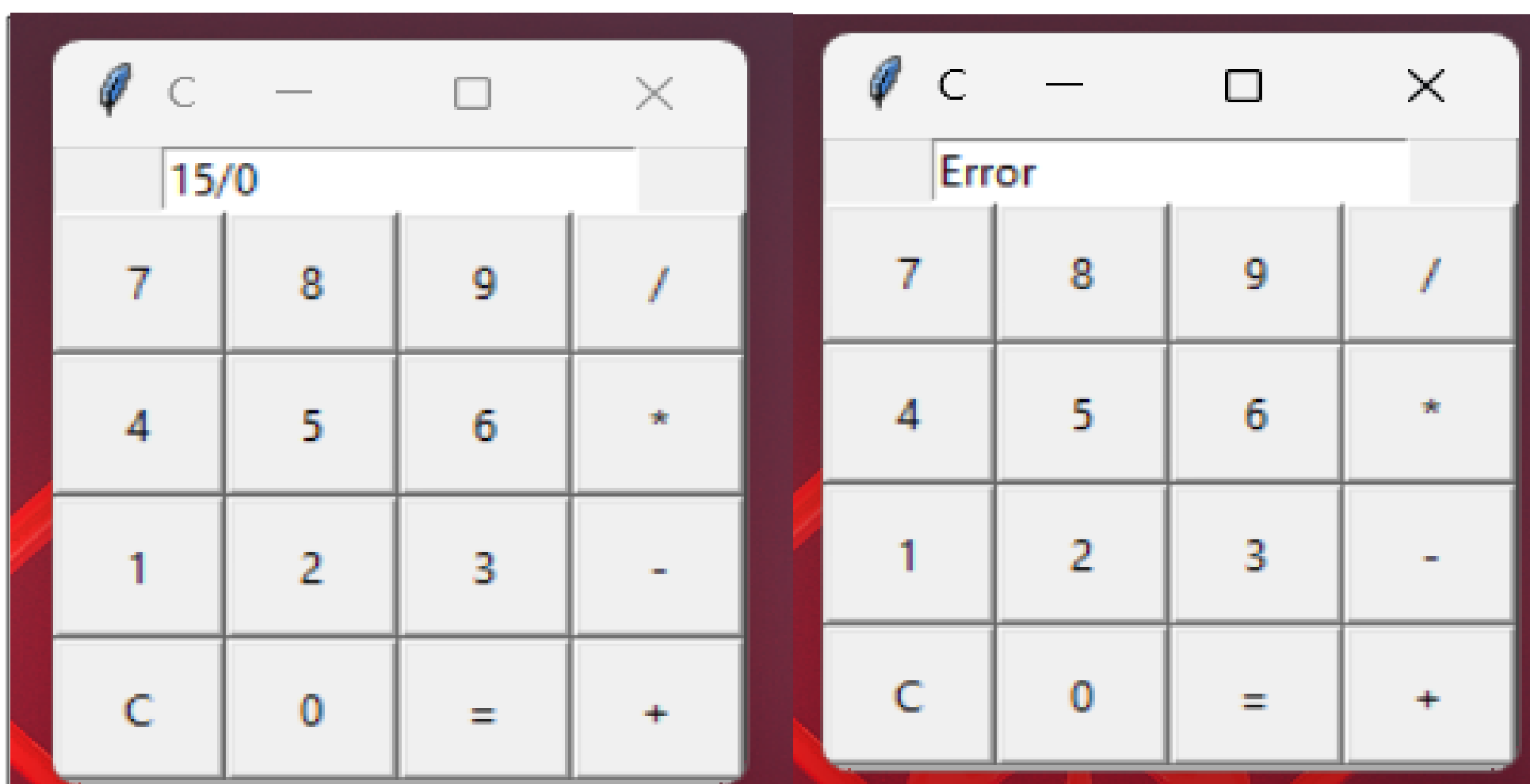

Figure 5: The Calculator User Interface

Figure 6: Result of dividing by zero

8 Testing Approach

Manual Black-box testing was performed to verify functionality:

1. **Unit Testing:** Verified individual functions (add_to_display, clear).

2. **Integration Testing:** Verified that clicking buttons correctly updates the display string.

3. **Edge Case Testing:**

   - Division by Zero (Input: 9/0=) → Result: Error.

6

- Multiple Operators (Input: 5++5=) → Result: Error (syntax check).
- Floating Point logic (Input: 5/2=) → Result: 2.5.

9   Challenges Faced

- **Layout Management:** Aligning buttons perfectly in a grid required careful adjustment of row and column spans, particularly for the display widget which needs to span all 4 columns.
- **State Management:** Ensuring the global expression variable was correctly updated and cleared to prevent old numbers from interfering with new calculations.
- **Error Trapping:** Preventing the application from crashing completely when the user enters mathematically impossible equations.

10   Learnings & Key Takeaways

- Gained proficiency in Python's Tkinter library, specifically the Entry and Button widgets.
- Understood the importance of separating UI definition (Grid layout) from application logic (Command callbacks).
- Learned how to use Lambda functions in Python to pass arguments to button callbacks (e.g., command=lambda: add_to_display('1')).

11   Future Enhancements

- **Keyboard Support:** Bind keyboard keys (Numpad) to the GUI buttons for faster input.
- **Scientific Mode:** Add buttons for Square Root, Exponent, and Modulo.
- **History Log:** Add a side panel to show previous calculations.
- **Styling:** Improve the visual design using custom fonts and color schemes (Dark Mode).

12   References

1. Python Software Foundation. "Tkinter — Python interface to Tcl/Tk." Python 3.10 Documentation.
2. Lutz, M. "Programming Python," O'Reilly Media.