
Project Report – Scientific Calculator

1.

Project Title: Scientific Calculator

Course/Subject:Introduction to problem solving

Submitted By: Shivam Gupta
Submitted To: D. Lakshmi

Institution: VIT BHOPAL

---

2. Introduction

The Scientific Calculator project is designed to perform both basic arithmetic and advanced scientific computations. While simple calculators provide only limited functions, this project offers a comprehensive tool capable of handling trigonometric, logarithmic, power, and other complex operations. The main objective is to create a user-friendly calculator that is efficient, accurate, and suitable for academic as well as general use.

---

3. Problem Statement

Manual mathematical calculations, especially complex ones, can lead to errors and take significant time. Basic calculators do not provide scientific features, and physical scientific calculators can be costly or unavailable. This project aims to create an accessible, accurate, and easy-to-use scientific calculator that performs a wide range of mathematical functions quickly.

---

4. Functional Requirements

Perform basic arithmetic operations: addition, subtraction, multiplication, and division.

Provide scientific functions:

Trigonometric functions (sin, cos, tan)

Logarithmic functions

Square and cube roots

Factorial and exponential functions

Power operations

Clear/reset functionality.

Display results instantly.

Handle invalid expressions gracefully.

---

5. Non-Functional Requirements

Usability: Simple and intuitive interface suitable for all users.

Performance: Calculations should be fast with minimal response time.

Accuracy: Must provide correct mathematical results with precision.

Reliability: Should function consistently without crashing.

Portability: Should run on multiple platforms (Windows, Linux, Mac).

Maintainability: Easy to update and add new functions.

---

6. System Architecture

The system follows a simple client-based architecture:

User Interface Layer: Handles user inputs and displays results.

Processing Layer: Executes calculations using mathematical logic and Python’s math library.

Output Layer: Shows results to the user in the UI panel.

Architecture Type: Layered / Modular Architecture

---

7. Design Diagrams

7.1 Use Case Diagram

Actors: User
Use Cases:

Enter input

Choose mathematical operation

Get result

Clear result

7.2 Workflow Diagram

1. User enters input

2. User selects an operation

3. System processes data

4. Result displayed

5. Option to clear/reset

7.3 Sequence Diagram

User → UI → Calculation Engine → UI → User

1. User selects function

2. UI sends input to backend

3. Backend performs calculation

4. UI displays result

7.4 Class/Component Diagram

Classes/Components:

CalculatorUI

CalculatorEngine

MathFunctions

Relationships:

UI interacts with Engine

Engine uses MathFunctions

7.5 ER Diagram (if storage used)

Not applicable (No database or storage used).

---

8. Design Decisions & Rationale

Python chosen due to rich math libraries and easy GUI development using Tkinter.

Modular design helps maintain code and add new functions.

Tkinter GUI selected for simplicity and cross-platform compatibility.

Math module ensures high precision and stability.

---

9. Implementation Details

Developed in Python 3.x.

GUI created using Tkinter.

Mathematical calculations handled by the math module.

Operations mapped to buttons for user-friendly navigation.

Input validation implemented to avoid runtime errors.

---

10. Screenshots / Results

Examples:

Home screen

Sample calculations (sin, cos, log, factorial)

---

11. Testing Approach

Unit Testing:

Tested each math function individually.

Functional Testing:

Checked accuracy of arithmetic and scientific operations.

Error Testing:

Input errors such as divide-by-zero and wrong expressions tested.

UI Testing:

Verified button functionality and display updates.

---

12. Challenges Faced

Managing complex scientific operations in the UI.

Handling invalid inputs and preventing crashes.

Ensuring the calculator layout remains clean and accessible.

Maintaining precision for trigonometric and logarithmic functions.

---

13. Learnings & Key Takeaways

Improved understanding of GUI development.

Enhanced knowledge of mathematical computing in Python.

Learned how to structure a software project from design to testing.

Gained experience in handling user input and exceptions effectively.

---

14. Future Enhancements

Add history/log feature to track previous calculations.

Include themes (dark mode, colorful themes).

Add graph plotting features for functions.

Voice input for calculations.

Support for complex numbers.

---

15. References

Python Official Documentation

Tkinter GUI Documentation

Math Module Docs

Online tutorials and resources for calculator logic
