MINI PROJECT REPORT

ON

<SCIENTIFIC CALCULATOR>

Submitted by

<Name: K.Manikanta Naidu>

<Roll No:NC.SC.U4CSE24042>

For

23CSE101-Computational Problem Solving

I Semester

B.Tech. CSE-A

School of Computing

Amrita Vishwa Vidyapeetham, Nagercoil

Index

Introduction:

I am THIRU MURUGAN from CSE-A

Here is my topic :SCIENTIFIC CALCULATOR

A scientific calculator in Python is a program that performs various mathematical and scientific operations beyond basic arithmetic. These operations typically include trigonometric functions, logarithms, exponentials, and more

Problem Statement:

In the modern world, performing complex mathematical calculations is essential for various fields like engineering, finance, data analysis, and education. Traditional calculators often lack the flexibility to handle advanced computations or custom user-defined functions, while existing software solutions can be overly complex, expensive, or not easily customizable.

This project aims to address the gap by developing a user-friendly, Python-based Scientific Calculator that supports a wide range of mathematical operations, including basic arithmetic, trigonometry, logarithmic calculations, exponentiation, and more. The tool will focus on simplicity, accuracy, and extensibility, allowing users to perform both standard and advanced mathematical computations efficiently in a single application.

Objectives:

- Implement core functions: basic arithmetic, trigonometry, logarithms, exponents, and more.
- Design a user-friendly, menu-driven interface for easy navigation.
- Ensure accurate and reliable calculations using Python libraries.
- Add error-handling for invalid inputs, division by zero, etc.
- Build a scalable structure to support future features like graphing or equation solving.
- Ensure cross-platform compatibility for Windows, Mac, and Linux.
- Optimize performance for efficient execution of complex operations.
- Provide clear documentation and help features for user guidance.
- Open-source the project to encourage community collaboration.
- Test rigorously to validate accuracy and robustness.

Python Libraries used in the project

NUMPY

A numpy stands for numerical python. It uses multi dimensional array object and has function and tools

PANDAS

PANDAS is a python library for a data analysis and manipulation.
PANDAS is high level tool for analysing.

MODULES OF THE PROJECT

- **Operations**: Functions (add, subtract, multiply, divide) perform math tasks. The divide function handles division by zero errors.
- **Input**: The get_input() function gets two numbers from the user and validates them.
- **Menu**: display_menu() prints the options for the user to choose an operation.
- **Main Logic**: The main() function runs a loop where the user selects an operation, inputs numbers, and gets a result. The operations are mapped in a dictionary for quick lookup.
- **Error Handling**: It checks for invalid inputs and division by zero using exceptions.
- **Exit**: After each calculation, the user can choose whether to continue.

CODE

```
import json
import matplotlib.pyplot as plt
from datetime import datetime

# Function to load expenses from file
def load_expenses(filename='expenses.json'):
    try:
        with open(filename, 'r') as file:
            return json.load(file)
    except (FileNotFoundError,
json.JSONDecodeError):
        return []

# Function to save expenses to file
def save_expenses(expenses,
filename='expenses.json'):
    with open(filename, 'w') as file:
        json.dump(expenses, file, indent=4)

# Function to add an expense
def add_expense():
    try:
        amount = float(input("Enter expense
amount: $"))
        category = input("Enter category (e.g.,
Food, Entertainment): ").strip()
        date_str = input("Enter date (YYYY-MM-
```

```
DD): ").strip()
        date = datetime.strptime(date_str, "%Y-%m-
%d") if date_str else datetime.today()

        expense = {
            "amount": amount,
            "category": category,
            "date": date.strftime("%Y-%m-%d")
        }
        return expense
    except ValueError:
        print("Invalid input. Please enter valid
data.")
        return None

# Function to display all expenses
def display_all_expenses(expenses):
    if not expenses:
        print("No expenses to display.")
        return
    print("\nAll Expenses:")
    for expense in expenses:
        print(f"Amount: ${expense['amount']:.2f},
Category: {expense['category']}, Date:
{expense['date']}")

# Function to display expenses by category
def display_expenses_by_category(expenses):
    category_totals = {}
```

```
for expense in expenses:

 category_totals[expense['category']] =
category_totals.get(expense['category'], 0) +
expense['amount']
    print("\nExpenses by Category:")
    for category, total in category_totals.items():
        print(f"{category}: ${total:.2f}")

# Function to generate a summary report
def generate_report(expenses):
    total_spent = sum(expense['amount'] for
expense in expenses)
    print(f"\nTotal Expenses: ${total_spent:.2f}")
    display_expenses_by_category(expenses)

# Function to visualize expenses by category
def visualize_expenses(expenses):
    category_totals = {}
    for expense in expenses:
        category_totals[expense['category']] =
category_totals.get(expense['category'], 0) +
expense['amount']

    categories = list(category_totals.keys())
    totals = list(category_totals.values())

    plt.figure(figsize=(10, 6))
    plt.bar(categories, totals, color='skyblue')
```

```
 plt.title('Expenses by Category')
    plt.xlabel('Category')
    plt.ylabel('Total Amount ($)')

 plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

# Function to display the main menu
def show_menu():
    print("\nExpense Tracker Menu:")
    print("1. Add an Expense")
    print("2. View All Expenses")
    print("3. View Expenses by Category")
    print("4. Generate Summary Report")
    print("5. Visualize Expenses by Category")
    print("6. Exit")

# Function to process user input and handle their
choice
def process_choice(choice, expenses):
    if choice == '1':
        expense = add_expense()
        if expense:
            expenses.append(expense)
            print(f"Expense of
${expense['amount']:.2f} added.")
    elif choice == '2':
        display_all_expenses(expenses)
```

```
elif choice == '3':
        display_expenses_by_category(expenses)
    elif choice == '4':
        generate_report(expenses)

 elif choice == '5':
        visualize_expenses(expenses)
    elif choice == '6':
        save_expenses(expenses)
        print("Goodbye!")
        return False
    else:
        print("Invalid choice. Please try again.")
    return True

# Main function that controls the flow of the
program
def main():
    expenses = load_expenses()

    while True:
        show_menu()
        choice = input("Enter your choice (1-6):
").strip()

        # Process user's choice
        if not process_choice(choice, expenses):
            break
```

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

OUTPUT

```
Expense Tracker Menu:
1. Add an Expense
2. View All Expenses
3. View Expenses by Category
4. Generate Summary Report
5. Visualize Expenses by Category
6. Exit
Enter your choice (1-6):  1
Enter expense amount: $ 50
Enter category (e.g., Food, Entertainment):  food
Enter date (YYYY-MM-DD):  2006-05-06
Expense of $50.00 added.

Expense Tracker Menu:
1. Add an Expense
2. View All Expenses
3. View Expenses by Category
4. Generate Summary Report
5. Visualize Expenses by Category
6. Exit
Enter your choice (1-6):  2

All Expenses:
Amount: $50.00, Category: food, Date: 2006-05-06

Expense Tracker Menu:
1. Add an Expense
2. View All Expenses
3. View Expenses by Category
4. Generate Summary Report
5. Visualize Expenses by Category
6. Exit
Enter your choice (1-6):  3
```

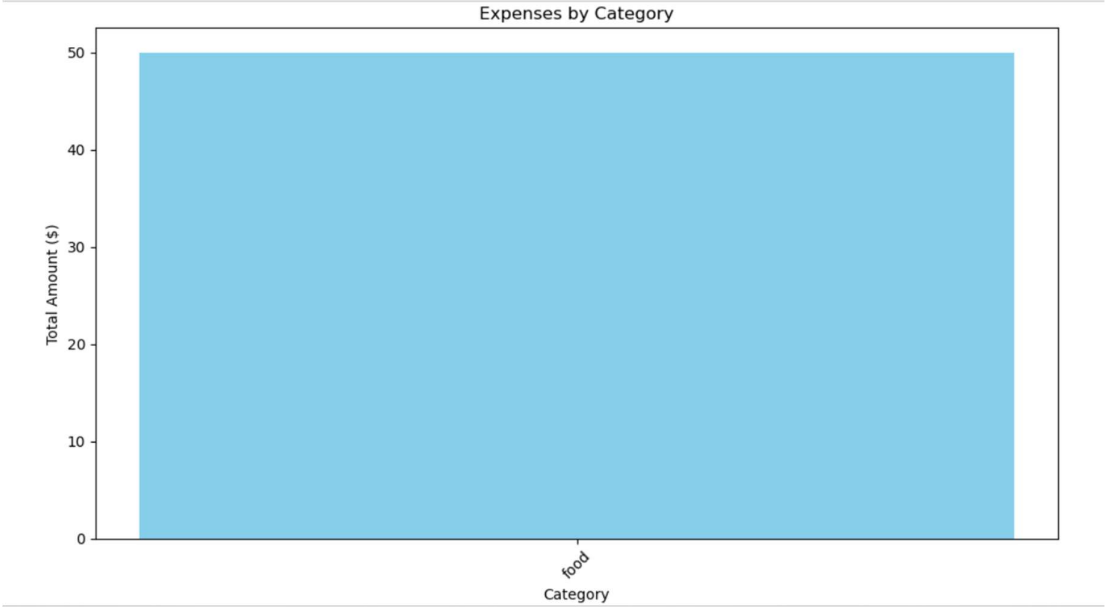
```
Expenses by Category:
food: $50.00

Expense Tracker Menu:
1. Add an Expense
2. View All Expenses
3. View Expenses by Category
4. Generate Summary Report
5. Visualize Expenses by Category
6. Exit
Enter your choice (1-6):  4

Total Expenses: $50.00

Expenses by Category:
food: $50.00

Expense Tracker Menu:
1. Add an Expense
2. View All Expenses
3. View Expenses by Category
4. Generate Summary Report
5. Visualize Expenses by Category
6. Exit
Enter your choice (1-6):  5
```



```
Expense Tracker Menu:
1. Add an Expense
2. View All Expenses
3. View Expenses by Category
4. Generate Summary Report
5. Visualize Expenses by Category
6. Exit
Enter your choice (1-6):  6
Goodbye!
```

APPLICATION OF THE PROJECT

The **Simple Calculator** has several practical applications:

1. **Daily Use**: For basic arithmetic like budgeting, shopping, or splitting bills.
2. **Business & Finance**: Quick calculations of costs, profits, or taxes for small businesses.
3. **Education**: Helps students and teachers with basic math operations and learning.
4. **Data Analysis**: Assists researchers and technicians with simple calculations.
5. **App Development**: Can be expanded for more complex tools in app or software development.
6. **Automation**: Used in scripts for basic calculations in data processing or logging.

This calculator serves as a foundation for more advanced applications in various fields.

LIMITATION OF THE PROJECTS

The **Simple Calculator** has the following limitations:

1. Only supports basic arithmetic operations (no advanced functions).
2. Text-based interface, lacking a graphical user interface (GUI).
3. No memory to store results or perform continuous calculations.
4. Limited error handling (only handles division by zero and invalid input).
5. Lacks scientific functions (e.g., trigonometry,
6. Basic input validation, not handling edge cases well.
7. Can only perform one operation at a time (no multi-operation support).

These limitations make it suitable for simple tasks but not complex calculations.

GITHUB LINK OF THE PROJECT

https://github.com/Manikanta991/Mani_143.git