**Developed by:** Phil Nevins  
**Course:** ECE 558 – Embedded Systems Programming Kotlin  
**Institution:** Portland State University  
**Date:** February 12, 2025

# ECE 558 Scientific Calculator Project Report

## ****1. Introduction****

The **ECE 558 Scientific Calculator** is an Android application developed using **Jetpack Compose**. The application provides basic and advanced mathematical functions with an intuitive user interface. The project was designed as part of the **ECE 558** course at **Portland State University**, aiming to explore modern Android development practices and state management in **Kotlin**.

## ****2. Features****

The calculator includes the following functionalities:

* **Basic Arithmetic Operations**: Addition (+), Subtraction (-), Multiplication (\*), Division (/)
* **Advanced Mathematical Functions**:
  + Square (x²), Square Root (√)
  + Logarithm (Log, Ln)
  + Trigonometric Functions (Sin, Cos, Tan)
  + Percentage Calculation (%)
  + +/- Toggle is fully implemented
  + 2nd button to bring up additional functions is not fully implemented
* **User Interface**:
  + **History Feature**: Displays the last five calculations
  + **Two-Screen Navigation**: A secondary screen for extended functionality (Under Construction)
  + **TI-89 Calculator Layout**

## ****3. Technologies Used****

* **Programming Language**: Kotlin
* **Framework**: Jetpack Compose
* **Navigation**: Jetpack Navigation
* **Material Design 3**

## ****4. Application Architecture****

The application is structured into multiple files:

### ****4.1 MainActivity.kt****

* **Handles the main application entry point.**
* **Manages navigation between screens.**
* **Implements Jetpack Compose UI components**.

### ****4.2 UI Component Functions****

* **CalculatorScreen()**: Implements the primary calculator UI and state management.
* **SecondScreen()**: Provides an alternative screen for additional functions.
* **HistoryDisplay()**: Displays the calculation history.
* **Display()**: Manages the expression and result display.
* **CalculatorButton()**: Defines button UI and interactions.

### ****4.3 Theme and Styling****

* **Color.kt**: Defines custom color schemes.
* **Theme.kt**: Configures Material Design 3 themes.
* **Type.kt**: Defines typography styles.

## ****5. Implementation Details****.

### ****5.1 State Management****

* The app uses **remember { mutableStateOf() }** to manage user input, history, and calculations.
* **Shared State Management** ensures history and user input persist across screens. (Under Construction)

### ****5.2 Navigation System**** (Under Construction)

* Implemented using **Jetpack Navigation Compose**.
* Two screens: **CalculatorScreen** (Main UI) and **SecondScreen** (Extended Functions).
* **NavigationController** ensures smooth transitions between screens.

### ****5.3 UI Enhancements****

* **Dynamic Expression Formatting**: Displays expressions like Sin(input) instead of just numbers.
* **Adaptive Layout**: Uses **LazyVerticalGrid** for button layout.

## ****6. Challenges and Solutions****

### ****6.1 Persistent State Across Screens****

* **Issue**: Navigating to the second screen reset the calculator history and input.
* **Solution**: Moved state management to **CalculatorApp()** to persist values across navigation. This solution was not fully implemented. It was a stretch goal

### ****6.2 Handling Trigonometric Inputs****

* **Issue:** Display was showing raw input instead of formatted expressions (e.g., displaying 7 instead of Sin(7)). This was the main issue.
* **Solution:** Updated state management by explicitly storing the operator and first operand when a function like Sin, Cos, or Tan is pressed. This ensures the display properly reflects the function call instead of just the input number.

#### ****Key Fix:****

Previously, when a user pressed Sin, Cos, or Tan, only the input was being stored, so the display would not update correctly. The fix was to **explicitly assign the operator and first operand in the state**, like this:

operator = button

firstOperand = input

This change ensures that the UI recognizes both the function and the input together, properly formatting expressions like Sin(7), Cos(45), and Tan(30) instead of just showing the number. This fix applies to all **single-number operator functions**, including Log, Ln, and √, but these only required:

operator = button

## ****7. Future Improvements****

* **Landscape Mode Support**: Optimize layout for tablet and landscape use.
* **Memory Storage**: Allow saving and recalling previous calculations.
* **More Functions:** Include additional scientific operations (e.g., factorial, exponentiation).
* **More Units and Display Options:** Include options to change to RAD/DEC and display units in scientific notation.

## ****8. Conclusion****

This project successfully demonstrated **Android development with Jetpack Compose**. By implementing a scientific calculator with **advanced mathematical operations, history tracking, and a modern UI**, the application provides a **solid foundation** for future enhancements. The use of **state management and navigation** improved performance and user experience.

# Appendix

## Button Functionality

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Button** | **Operand 1** | **Operand 2** | **Operation** |
| + (add) | valid decimal | valid decimal | op1 + op2 |
| - (subtract) | valid decimal | valid decimal | op1 - op2 |
| X (multiply) | valid decimal | valid decimal | op1 \* op2 |
| / (divide) | valid decimal | valid decimal | op1 / op2 (Error if op2 = 0) |
| % (percent) | valid decimal | no entry | op1 / 100.0 |
| √ (square root) | valid decimal | no entry | Math.sqrt(op1) (Error if op1 < 0) |
| x² (square) | valid decimal | no entry | op1 \* op1 |
| Log (logarithm base 10) | valid decimal | no entry | log10(op1) (Error if op1 ≤ 0) |
| Ln (natural log) | valid decimal | no entry | ln(op1) (Error if op1 ≤ 0) |
| Sin (sine) | valid decimal | no entry | sin(op1) |
| Cos (cosine) | valid decimal | no entry | cos(op1) |
| Tan (tangent) | valid decimal | no entry | tan(op1) |
| + (add) | no entry | no entry | ERROR |
| - (subtract) | no entry | valid decimal | ERROR |
| X (multiply) | no entry | valid decimal | ERROR |
| / (divide) | valid decimal | 0 | ERROR (division by zero) |
| % (percent) | no entry | valid decimal | ERROR |
| √ (square root) | valid decimal (negative) | no entry | ERROR (negative input) |
| x² (square) | no entry | valid decimal | ERROR |
| Log (logarithm base 10) | valid decimal (≤ 0) | no entry | ERROR (input ≤ 0) |
| Ln (natural log) | valid decimal (≤ 0) | no entry | ERROR (input ≤ 0) |
| Sin (sine) | no entry | valid decimal | ERROR |
| Cos (cosine) | no entry | valid decimal | ERROR |
| Tan (tangent) | no entry | valid decimal | ERROR |
| 2nd |  |  | OPENS NEW SCREEN |

## Psuedocode

BEGIN Program

DEFINE MainActivity CLASS

FUNCTION onCreate(Bundle)

CALL setContent

APPLY CalculatorTheme

SET background color

CALL CalculatorApp()

DEFINE FUNCTION CalculatorApp()

CREATE navigation controller

INITIALIZE state variables (input, result, operator, firstOperand, history, etc.)

DEFINE Navigation Host:

"calculatorScreen" → CALL CalculatorScreen()

"secondScreen" → CALL SecondScreen()

DEFINE FUNCTION CalculatorScreen()

DECLARE input, result, operator, firstOperand, history as mutable states

DISPLAY:

Show Title: "ECE 558 Calculator"

Show Calculation History (last 5 calculations)

Show Expression & Result

CREATE Button Grid:

FOR each button in ["<-", "√", "x²", "CH", "C", "Sin", "Log", "Ln", "%", "\*", ..., "="]

CREATE Button

ON Button Click:

IF button is "2nd"

NAVIGATE to "SecondScreen"

ELSE IF button is "<-"

REMOVE last character from input

ELSE IF button is "C"

CLEAR input, result, operator, firstOperand

ELSE IF button is "CH"

IF first press → Clear current inputs

ELSE IF second press → Clear history

ELSE IF button is an Operator ("+", "-", "\*", "/")

ELSE IF button is Function ("Sin", "Cos", "Tan", "Log", "Ln", "√")

ELSE IF button is "%"

DIVIDE input by 100

ELSE IF button is "x²"

SQUARE input value

ELSE IF button is "+/-"

TOGGLE sign of input

ELSE IF button is "="

DEFINE FUNCTION SecondScreen()

SIMILAR to CalculatorScreen, but with alternative functions

INCLUDE "Back" button to return to CalculatorScreen

DEFINE FUNCTION HistoryDisplay(history)

DISPLAY last 5 calculations in history

DEFINE FUNCTION Display(input, result, operator, firstOperand)

FORMAT and DISPLAY current expression and result

DEFINE FUNCTION CalculatorButton(text, onClick)

SET Button color based on function type

EXECUTE onClick action when pressed

DEFINE FUNCTION calculate(firstOperand, secondOperand, operator)

CONVERT operands to Double

EXECUTE operation based on operator

RETURN result

DEFINE FUNCTION CalculatorAppPreview()

PREVIEW CalculatorApp() using Jetpack Compose tools

END Program