COMP 1786 Logbook Exercise 1

Basic Information

1.1 Student name	**Nguyen Duc Hieu**
1.2 Who did you work with? Note that for logbook exercises you are allowed to work with one other person as long as you give their name and login id and both contribute to the work.	**Name:** **Login id:**
1.3 Which Exercise is this? Tick as appropriate.	• Exercise 1 ☒ • Exercise 2 ☐ • Exercise 3 ☐
1.4 How well did you complete the exercise? Tick as appropriate.	• I tried but couldn't complete it ☐ • I did it but I feel I should have done better ☐ • I did everything that was asked ☐ • I did more than was asked for ☒
1.5 Briefly explain your answer to question 1.4. Without any explanation/justification, your scores will be deducted.	**A, Frontend** 1. **Animated background slideshow** (3 images crossfading every 5s with Handler-managed lifecycle-aware rotation) 2. **Real-time blur/frosted glass effect** (RenderEffect + alpha overlay for Android 12+, graceful degradation for older versions) 3. **Structured, responsive layout** using ConstraintLayout + percentage sizing (width_percent, height_percent) for adaptive scaling vs simple fixed LinearLayout 4. **Distinct styled circular buttons** with custom drawables, ripple effects, accent vs normal color schemes, and bold style for equals – improved visual affordance 5. **Glass morphism UI design** with semi-transparent panels, gradient overlays, elevation shadows, and visual divider – modern aesthetic beyond basic layouts 6. **Extra keys beyond spec:** Clear (C), Backspace, Percent (%), Sign toggle (±), Decimal dot (.) – only +−×÷ and two operands were required 7. **Dual-line display:** expression line separate from result line with ellipsize handling and text size hierarchy – foundational spec only needs result display 8. **Thousand grouping / number formatting** (comma separators e.g.,

1,234,567) and engineering/scientific notation fallback for very large numbers – formatting was not mandated

9. **Extracted dimens/colors/styles resources** for maintainability (text sizes, margins, paddings, semantic color naming, style inheritance) vs hardcoded values
10. **Six custom XML animations** with precise timing and interpolators:

Number entry (scale 0.7→1.0 + fade, 200ms decelerate)

a. Result reveal (translate Y + fade + scale, 300ms decelerate_cubic)
b. Error shake (oscillation with cycle, 400ms × 3 repeats)
c. Clear button rotate (360° + scale pulse, 300ms overshoot)
d. Button press/release (scale feedback, 100ms)

B, Interaction / UX

1. **Sign toggle logic** on current input or first operand – negative numbers support not explicitly specified Percent operation converting current input to its fractional value
2. **Percent operation** converting current input to fractional value (÷100 with scale and HALF_UP rounding, 10 decimal precision)
3. **Backspace with state awareness** (removes operator, clears after computation, or deletes last digit depending on context) – nuanced editing beyond simple clear
4. **Intermediate chained computation** (pressing operator after entering second operand performs pending operation automatically)
5. **Automatic input sanitization:** stripping trailing dots, collapsing leading zeros, handling sign edge cases – prevents invalid states
6. **Ellipsizing on text overflow** to prevent hard UI breaks and maintain layout integrity even with long numbers
7. **Three-tier haptic feedback system:**

a. Light (10ms) for number buttons
b. Medium (20ms) for operators and copy actions
c. Heavy (50ms) for error conditions Provides tactile context beyond visual feedback

8. **Copy to clipboard functionality:**

a) Long press on result to copy (auto-strips "= " prefix)
b) Long press on expression to copy full calculation
c) Toast notifications + haptic confirmation
d) Empty state handling

9. **Operator highlight system:**

a) Active operator button dimmed to 60% opacity
b) Visual state persistence during calculation
c) Auto-reset on completion or clear Helps user track current operation

mode

10. Live preview calculation:

a. Real-time result display while typing second operand
b. Shown at 50% opacity with "= " prefix to distinguish from final result
c. Updates instantly per keystroke
d. Graceful fallback on calculation errors Reduces need to press "=" for simple checks

C, Backend / Architecture

1. **Separation of concerns:** Calculator pure engine class extracted from MainActivity – complete logic/UI decoupling for testability
2. **Use of BigDecimal** for high precision arithmetic and stripTrailingZeros – higher numerical fidelity than double or float
3. **Structured Result type** with explicit error codes (divide_by_zero, invalid_a, invalid_b, unsupported_op) – typed error modeling vs exception strings
4. **Percentage operation** implemented with divide, scale, and rounding mode (HALF_UP, 10 decimal places) – proper mathematical handling
5. **Input sanitization utility** (sanitizeNumber) centralizing edge case handling: trailing dots, leading zeros, sign normalization, empty inputs
6. **Formatting utility** (format / formatStringSafe) with thousand separators and large number fallback (engineering notation) – display-ready strings
7. **Graceful error toasts** for invalid input / division by zero with user-friendly localized messages vs bare validation
8. **Handler-managed lifecycle-aware** background rotation runnable with proper onStart/onStop/onDestroy cleanup – prevents memory leaks
9. **Operator state tracking** with View reference for UI highlight management – maintains visual consistency
10. **Multi-state display logic** managing idle, input, operator, preview, and result states – clear state machine pattern
11. **Animation resource management:** loaded once in onCreate, reusable Animation objects, memory-efficient pooling vs repeated inflation
12. **Clipboard integration** using ClipboardManager service, ClipData with labeled content, proper permission handling (VIBRATE only, no storage needed)
13. **Vibrator service management** with capability checking (hasVibrator), API level fallback (VibrationEffect for O+, direct vibrate for legacy), enum-based intensity control

2. Exercise answer

2.1 Screen shots demonstrating what you achieved.

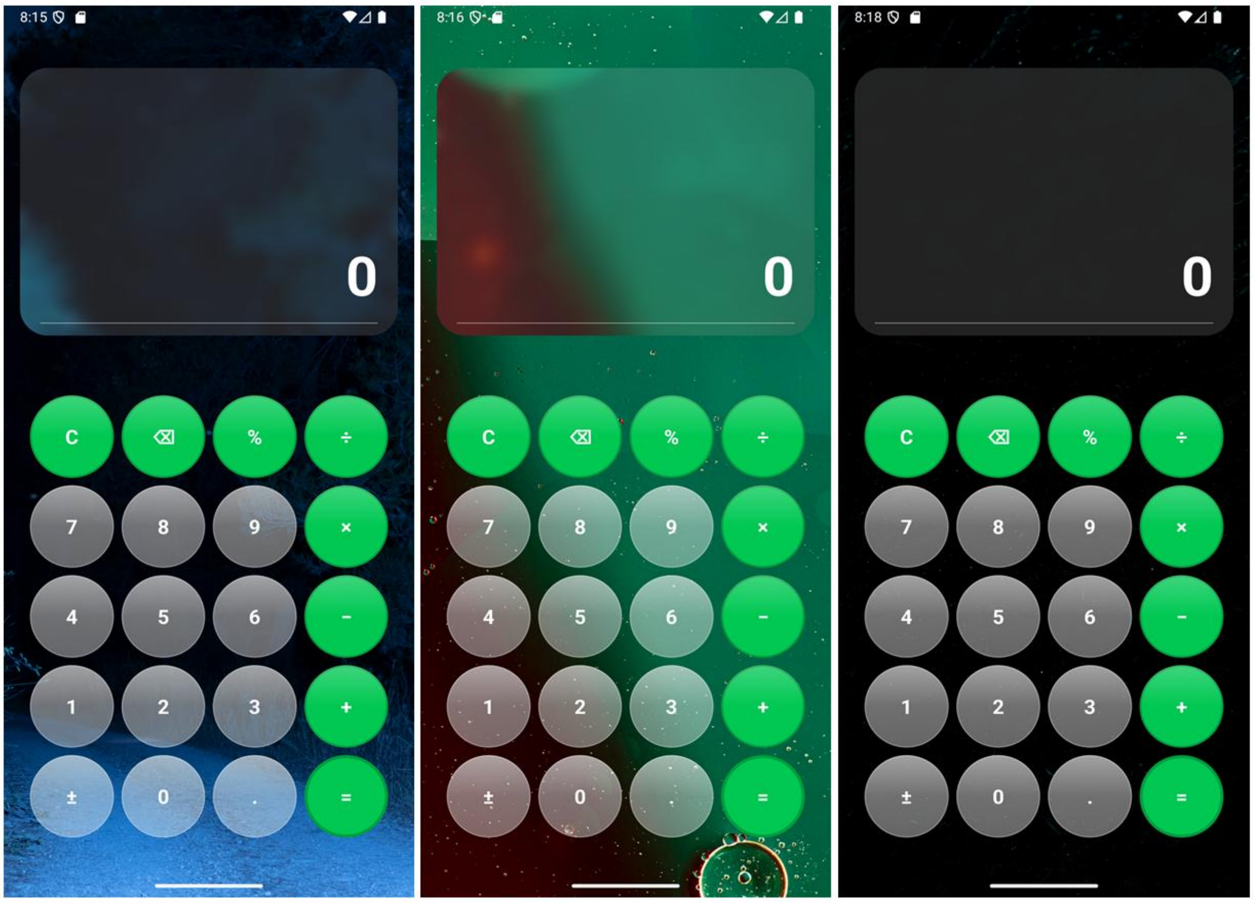


Figure 1 - Animated background slideshow

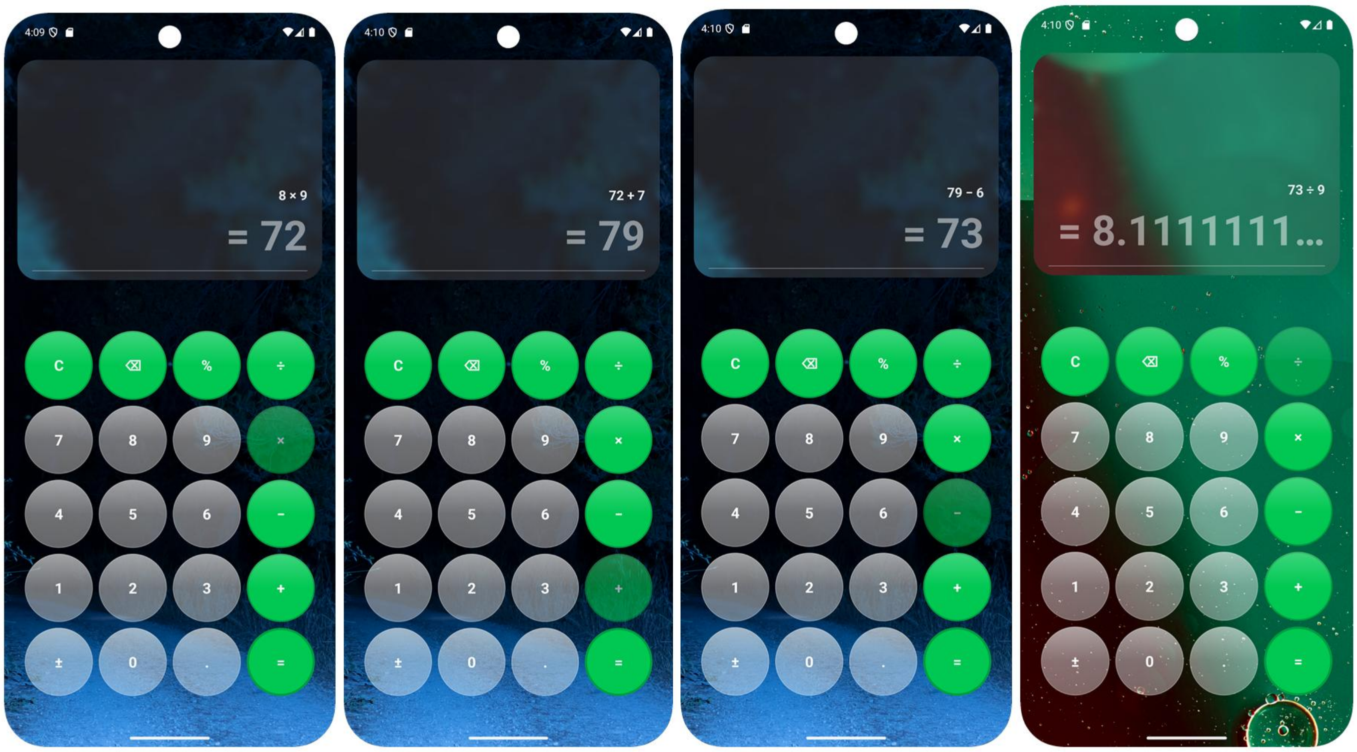


Figure 2 - Main operator

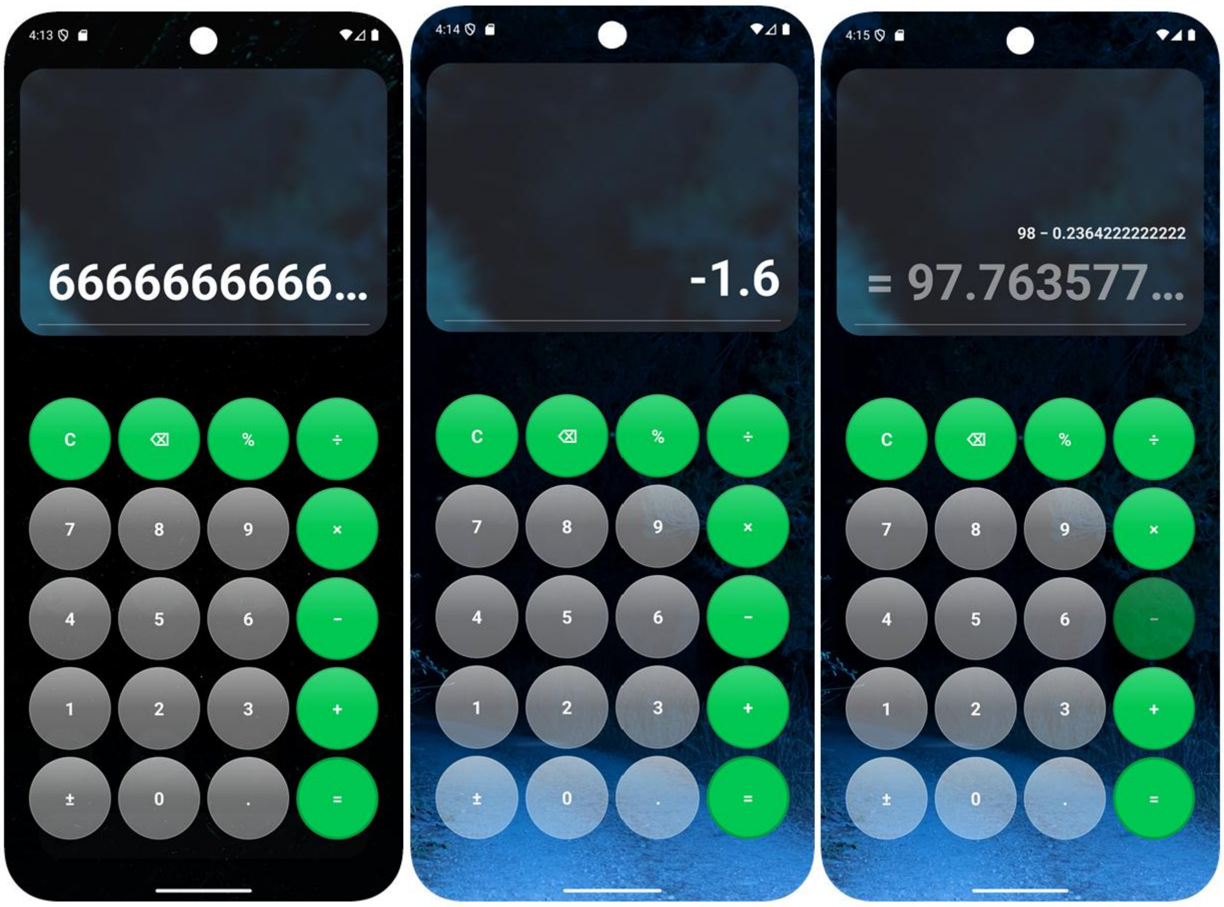
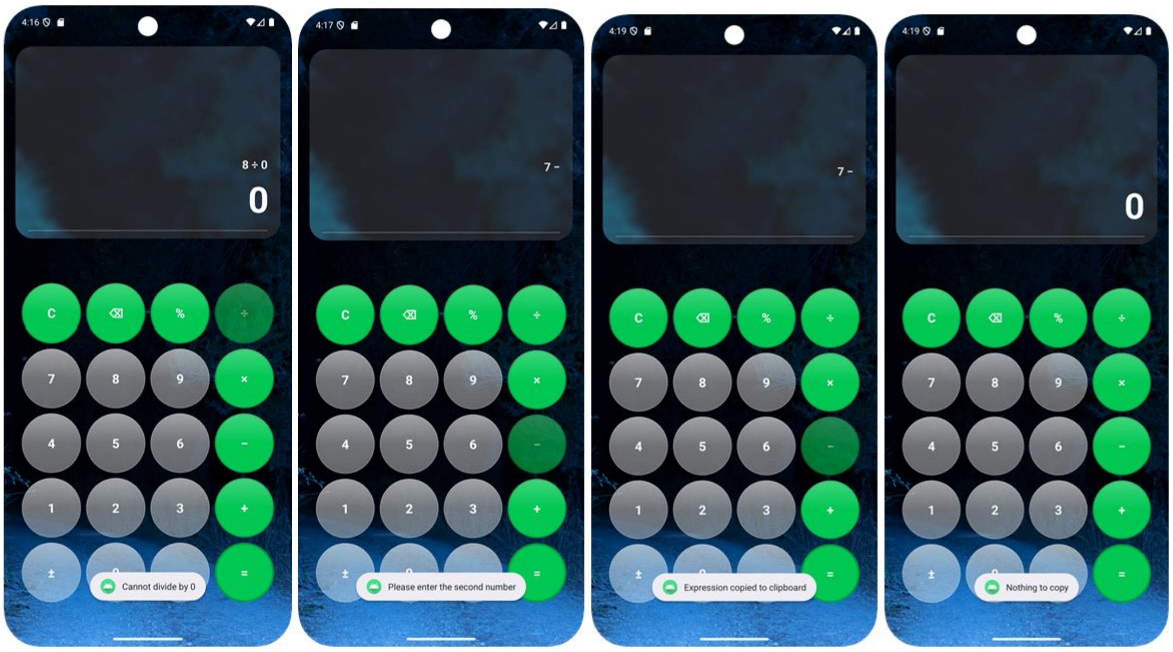


Figure 3 - Some special display format



Cannot divide by 0

Please enter the second number

Expression copied to clipboard

Nothing to copy

Figure 4 - Some input validation

2.2 Code that you wrote

Calculator.java

```java
package com.example.comp1786_logbook_ex1_calculatorapplication;

import java.math.BigDecimal;
import java.math.RoundingMode;

/**
 * Pure Java calculator engine: parse/sanitize inputs, compute basic binary ops, and format for
display.
 */
public final class Calculator {

    private Calculator() {}

    public static final class Result {
        public final boolean ok;
        public final BigDecimal value;
        public final String error;

        private Result(boolean ok, BigDecimal value, String error) {
            this.ok = ok; this.value = value; this.error = error;
        }
        public static Result ok(BigDecimal v) { return new Result(true, v, null); }
        public static Result error(String msg) { return new Result(false, null, msg); }
    }

    public static BigDecimal parse(String s) throws NumberFormatException {
        if (s == null) throw new NumberFormatException("null");
        String normalized = sanitizeNumber(s);
        return new BigDecimal(normalized);
    }

    /** Sanitize user input: trim spaces, handle leading/trailing dots, collapse leading zeros.
*/
    public static String sanitizeNumber(String s) {
        if (s == null) return "0";
        String t = s.trim();
        if (t.isEmpty()) return "0";
        if (t.endsWith(".")) t = t.substring(0, t.length() - 1);
        if (t.startsWith(".")) t = "0" + t;
        boolean neg = t.startsWith("-");
        String core = neg ? t.substring(1) : t;
        if (core.contains(".")) {
            int idx = core.indexOf('.');
            String intPart = core.substring(0, idx);
            String fracPart = core.substring(idx + 1);
            String intNorm = intPart.replaceFirst("^0+(?!$)", "");
            if (intNorm.isEmpty()) intNorm = "0";
            core = intNorm + "." + fracPart;
        } else {
            core = core.replaceFirst("^0+(?!$)", "");
            if (core.isEmpty()) core = "0";
        }
        return neg ? ("-" + core) : core;
    }

    public static Result compute(String aStr, String bStr, String op) {
        BigDecimal a, b;
        try {
            a = (aStr == null) ? BigDecimal.ZERO : parse(aStr);
        } catch (NumberFormatException ex) {
            return Result.error("invalid_a");
        }
        try {
            b = (bStr == null) ? null : parse(bStr);
        } catch (NumberFormatException ex) {
            return Result.error("invalid_b");
        }
        if (b == null) return Result.error("missing_b");

        BigDecimal res;
        switch (op) {
            case "+": res = a.add(b); break;
            case "−": case "-": res = a.subtract(b); break;
            case "×": case "x": case "X": res = a.multiply(b); break;
            case "÷": case "/":
                if (b.compareTo(BigDecimal.ZERO) == 0) return Result.error("divide_by_zero");
                res = a.divide(b, 10, RoundingMode.HALF_UP);
                break;
            default: return Result.error("unsupported_op");
        }
        res = res.stripTrailingZeros();
        return Result.ok(res);
    }

    /** Format BigDecimal: add grouping separators for moderate sizes; otherwise engineering
string. */
    public static String format(BigDecimal bd) {
        if (bd == null) return "0";
        BigDecimal v = bd.stripTrailingZeros();
        String plain = v.toPlainString();
        if (plain.contains("E") || plain.contains("e")) {
            return v.toEngineeringString();
        }
        boolean neg = plain.startsWith("-");
        String core = neg ? plain.substring(1) : plain;
        String intPart = core;
        String fracPart = null;
        if (core.contains(".")) {
            int idx = core.indexOf('.');
            intPart = core.substring(0, idx);
            fracPart = core.substring(idx + 1);
        }
        if (intPart.length() > 15) {
            return v.toEngineeringString();
        }
        StringBuilder grouped = new StringBuilder();
        int len = intPart.length();
        for (int i = 0; i < len; i++) {
            grouped.append(intPart.charAt(i));
            int posFromEnd = len - i - 1;
            if (posFromEnd > 0 && posFromEnd % 3 == 0) grouped.append(',');
        }
        StringBuilder out = new StringBuilder();
        if (neg) out.append('-');
        out.append(grouped);
        if (fracPart != null && !fracPart.isEmpty()) {
            out.append('.').append(fracPart);
        }
        return out.toString();
    }

    /** Format a numeric string; if parsing fails, return original string. */
    public static String formatStringSafe(String s) {
        try {
            BigDecimal bd = parse(s);
            return format(bd);
        } catch (NumberFormatException ex) {
            return s == null ? "" : s;
        }
    }
}
```

MainActivity.java

```
package com.example.comp1786_logbook_ex1_calculatorapplication;

import android.content.ClipData;
import android.content.ClipboardManager;
import android.content.Context;
import android.graphics.RenderEffect;
import android.graphics.Shader;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.os.VibrationEffect;
import android.os.Vibrator;
import android.text.TextUtils;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.GridLayout;
import android.widget.ImageView;
```

```java
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.text.DecimalFormat;

/** Main activity controlling UI and delegating math to Calculator. */
public class MainActivity extends AppCompatActivity {

    private TextView txtExpression, txtResult;
    private GridLayout grid;

    // Animations
    private Animation numberEntryAnim;
    private Animation resultRevealAnim;
    private Animation errorShakeAnim;
    private Animation clearRotateAnim;
    private Animation buttonPressAnim;
    private Animation buttonReleaseAnim;

    // Haptic feedback
    private Vibrator vibrator;

    // Background slideshow
    private static final long INTERVAL_MS = 5_000;
    private static final long FADE_MS = 600;
    private final int[] BACKGROUNDS = new int[] {
            R.drawable.bg_leaves,
            R.drawable.bgimage,
            R.drawable.bgimage2
    };
    private ImageView bg1, bg2, ivBlur;
    private boolean showingFirst = true;
    private int index = 0;

    private final Handler handler = new Handler(Looper.getMainLooper());
    private final Runnable switchTask = new Runnable() {
        @Override public void run() {
            if (!isFinishing()) {
                crossfadeToNext();
                handler.postDelayed(this, INTERVAL_MS);
            }
        }
    };

    // Calculator state
    private String firstOperand = null;
    private String currentOp = null; // "+", "-", "×", "÷"
    private final StringBuilder input = new StringBuilder();
    private boolean computed = false;

    // UI state
    private View lastOperatorButton = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        txtExpression = findViewById(R.id.txtExpression);
        txtResult = findViewById(R.id.txtResult);
        grid = findViewById(R.id.grid);

        // Load animations
        numberEntryAnim = AnimationUtils.loadAnimation(this, R.anim.number_entry);
        resultRevealAnim = AnimationUtils.loadAnimation(this, R.anim.result_reveal);
        errorShakeAnim = AnimationUtils.loadAnimation(this, R.anim.error_shake);
        clearRotateAnim = AnimationUtils.loadAnimation(this, R.anim.clear_button_rotate);
        buttonPressAnim = AnimationUtils.loadAnimation(this, R.anim.button_press);
        buttonReleaseAnim = AnimationUtils.loadAnimation(this, R.anim.button_release);

        // Initialize vibrator for haptic feedback
        vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);

        setupBackgroundRotation();
        wireKeysFromGrid();
        setupCopyToClipboard();
        updateDisplay();
    }

    @Override
    protected void onStart() {
        super.onStart();
        if (BACKGROUNDS.length > 1) {
            handler.removeCallbacks(switchTask);
            handler.postDelayed(switchTask, INTERVAL_MS);
        }
    }

    @Override
    protected void onStop() {
        super.onStop();
        handler.removeCallbacks(switchTask);
    }

    private void setupBackgroundRotation() {
        bg1 = findViewById(R.id.bgImage);
        bg2 = findViewById(R.id.bgImage2);
        ivBlur = findViewById(R.id.ivBlur);

        if (BACKGROUNDS.length > 0) {
            index = 0;
            if (bg1 != null) bg1.setImageResource(BACKGROUNDS[index]);
            if (ivBlur != null) {
                ivBlur.setImageResource(BACKGROUNDS[index]);
                applyBlurIfSupported(ivBlur);
            }
        }
    }

    private void crossfadeToNext() {
        if (BACKGROUNDS.length <= 1) return;

        index = (index + 1) % BACKGROUNDS.length;
        final ImageView fadeOut = showingFirst ? bg1 : bg2;
        final ImageView fadeIn = showingFirst ? bg2 : bg1;
        if (fadeIn == null || fadeOut == null) return;

        fadeIn.setImageResource(BACKGROUNDS[index]);
        fadeIn.setAlpha(0f);
        fadeIn.animate().alpha(1f).setDuration(FADE_MS).start();
        fadeOut.animate().alpha(0f).setDuration(FADE_MS).start();

        if (ivBlur != null) {
            ivBlur.setImageResource(BACKGROUNDS[index]);
            applyBlurIfSupported(ivBlur);
        }
        showingFirst = !showingFirst;
    }

    private void applyBlurIfSupported(ImageView target) {
        if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.S) {
            target.setRenderEffect(RenderEffect.createBlurEffect(28f, 28f,
Shader.TileMode.CLAMP));
        }
    }

    private void setupCopyToClipboard() {
        // Long press on result to copy
        txtResult.setOnLongClickListener(v -> {
            String text = txtResult.getText().toString();
            if (text.isEmpty() || text.equals("0")) {
                toast(getString(R.string.msg_nothing_to_copy));
                return true;
            }

            // Remove "= " prefix if present
            if (text.startsWith("= ")) {
                text = text.substring(2);
            }

            ClipboardManager clipboard = (ClipboardManager)
getSystemService(Context.CLIPBOARD_SERVICE);
            ClipData clip = ClipData.newPlainText("calculator_result", text);
            clipboard.setPrimaryClip(clip);

            toast(getString(R.string.msg_result_copied));
            performHapticFeedback(HapticFeedbackType.MEDIUM);
            return true;
        });

        // Long press on expression to copy
        txtExpression.setOnLongClickListener(v -> {
            String text = txtExpression.getText().toString();
            if (text.isEmpty()) {
                toast(getString(R.string.msg_nothing_to_copy));
                return true;
            }

            ClipboardManager clipboard = (ClipboardManager)
getSystemService(Context.CLIPBOARD_SERVICE);
            ClipData clip = ClipData.newPlainText("calculator_expression", text);
            clipboard.setPrimaryClip(clip);

            toast(getString(R.string.msg_expression_copied));
            performHapticFeedback(HapticFeedbackType.MEDIUM);
            return true;
        });
    }

    private void wireKeysFromGrid() {
        int n = grid.getChildCount();
        for (int i = 0; i < n; i++) {
            View v = grid.getChildAt(i);
            if (v instanceof TextView) {
                final TextView btn = (TextView) v;
                btn.setClickable(true);
                btn.setOnClickListener(view -> {
                    // Animate button press
                    animateButtonPress(btn);
                    onKey(btn.getText().toString());
                });
            }
        }
    }

    private void animateButtonPress(View button) {
        button.startAnimation(buttonPressAnim);
        button.postDelayed(() -> button.startAnimation(buttonReleaseAnim), 100);
        // Light haptic feedback for button press
        performHapticFeedback(HapticFeedbackType.LIGHT);
    }

    // Haptic feedback helper methods
    private enum HapticFeedbackType {
        LIGHT,    // For normal button presses
        MEDIUM,   // For operators
        HEAVY     // For errors
    }

    private void performHapticFeedback(HapticFeedbackType type) {
        if (vibrator == null || !vibrator.hasVibrator()) return;

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            VibrationEffect effect;
            switch (type) {
                case LIGHT:
                    effect = VibrationEffect.createOneShot(10,
VibrationEffect.DEFAULT_AMPLITUDE);
                    break;
                case MEDIUM:
                    effect = VibrationEffect.createOneShot(20,
VibrationEffect.DEFAULT_AMPLITUDE);
                    break;
                case HEAVY:
                    effect = VibrationEffect.createOneShot(50,
VibrationEffect.DEFAULT_AMPLITUDE);
                    break;
                default:
                    effect = VibrationEffect.createOneShot(10,
VibrationEffect.DEFAULT_AMPLITUDE);
            }
            vibrator.vibrate(effect);
        } else {
            // Fallback for older devices
            switch (type) {
                case LIGHT:
                    vibrator.vibrate(10);
                    break;
                case MEDIUM:
                    vibrator.vibrate(20);
                    break;
                case HEAVY:
                    vibrator.vibrate(50);
                    break;
            }
        }
    }

    private void onKey(String key) {
        String clear = getString(R.string.key_clear);
        String back = getString(R.string.key_backspace);
        String pct = getString(R.string.key_percent);
        String eq = getString(R.string.key_equals);
        String plusMinus = getString(R.string.key_plusminus);

        if (key.equals(clear)) {
            // Find and animate the clear button
            animateClearButton();
            performHapticFeedback(HapticFeedbackType.MEDIUM);
            clearAll();
            return;
        }
        if (key.equals(back)) { backspace(); return; }
        if (key.equals(pct)) { percent(); return; }
        if (key.equals(eq)) { equalsCompute(); return; }
        if (key.equals(plusMinus)) { toggleSign(); return; }

        if (key.equals(getString(R.string.key_add)) || key.equals(getString(R.string.key_sub))
                || key.equals(getString(R.string.key_mul)) ||
key.equals(getString(R.string.key_divide))) {
            setOperator(key); return;
        }
        if (key.equals(getString(R.string.key_dot))) { appendDot(); return; }

        if (key.length() == 1 && Character.isDigit(key.charAt(0))) {
            appendDigit(key.charAt(0)); return;
        }
        toast(String.format(getString(R.string.msg_key_not_supported), key));
    }

    private void animateClearButton() {
        // Find the clear button in grid and animate it
        String clearKey = getString(R.string.key_clear);
        for (int i = 0; i < grid.getChildCount(); i++) {
            View v = grid.getChildAt(i);
            if (v instanceof TextView) {
                TextView btn = (TextView) v;
                if (clearKey.equals(btn.getText().toString())) {
                    btn.startAnimation(clearRotateAnim);
                    break;
                }
            }
        }
    }

    private void toggleSign() {
        if (input.length() > 0) {
            if (input.charAt(0) == '-') input.deleteCharAt(0);
            else input.insert(0, '-');
        } else if (firstOperand != null && currentOp == null) {
            if (firstOperand.startsWith("-")) firstOperand = firstOperand.substring(1);
            else firstOperand = "-" + firstOperand;
        }
        updateDisplay();
    }

    private void appendDigit(char d) {
        if (computed) clearAll();
        if (input.length() == 1 && input.charAt(0) == '0') {
            if (d != '0') input.setCharAt(0, d);
        } else {
            input.append(d);
        }
        updateDisplay();
        // Animate number entry
        txtResult.startAnimation(numberEntryAnim);
    }

    private void appendDot() {
        if (computed) clearAll();
        if (TextUtils.indexOf(input, ".") >= 0) return;
        if (input.length() == 0) input.append("0.");
        else input.append('.');
        updateDisplay();
        // Animate number entry
        txtResult.startAnimation(numberEntryAnim);
    }

    private void setOperator(String op) {
        if (input.length() == 0 && firstOperand == null) {
            firstOperand = "0";
        } else if (firstOperand == null) {
            firstOperand = Calculator.sanitizeNumber(input.toString());
            input.setLength(0);
        } else if (currentOp != null && input.length() > 0) {
            equalsCompute(false);
        }
        currentOp = op;
        computed = false;

        // Highlight the operator button
        highlightOperatorButton(op);

        updateDisplay();
    }

    private void highlightOperatorButton(String op) {
        // Reset previous operator button
        if (lastOperatorButton != null) {
            lastOperatorButton.setAlpha(1.0f);
            lastOperatorButton = null;
        }

        // Find and highlight current operator button
        for (int i = 0; i < grid.getChildCount(); i++) {
            View v = grid.getChildAt(i);
            if (v instanceof TextView) {
                TextView btn = (TextView) v;
                if (op.equals(btn.getText().toString())) {
                    btn.setAlpha(0.6f);
                    lastOperatorButton = btn;
                    break;
                }
            }
        }
    }

    private void backspace() {
        if (input.length() > 0)      input.deleteCharAt(input.length() - 1);
        else if (currentOp != null)  currentOp = null;
        else if (computed)           clearAll();
        updateDisplay();
    }

    private void percent() {
        if (input.length() == 0) return;
        try {
            BigDecimal x = Calculator.parse(input.toString());
            x = x.divide(BigDecimal.valueOf(100), 10,
RoundingMode.HALF_UP).stripTrailingZeros();
            input.setLength(0);
            input.append(x.toPlainString());
            updateDisplay();
        } catch (NumberFormatException ex) {
            toast(getString(R.string.msg_invalid_number));
        }
    }

    private void equalsCompute() { equalsCompute(true); }

    private void equalsCompute(boolean showEquals) {
        if (currentOp == null) {
            if (input.length() == 0 && firstOperand != null) computed = true;
            updateDisplay(); return;
        }
        String aStr = (firstOperand != null) ? firstOperand : "0";
        String bStr = (input.length() > 0) ? input.toString() : null;
        if (bStr == null) {
            toast(getString(R.string.msg_enter_second_number));
            // Shake on error
            txtResult.startAnimation(errorShakeAnim);
            performHapticFeedback(HapticFeedbackType.HEAVY);
            return;
        }

        Calculator.Result r = Calculator.compute(aStr, bStr, currentOp);
        if (!r.ok) {
            // Shake on error
            txtResult.startAnimation(errorShakeAnim);
            performHapticFeedback(HapticFeedbackType.HEAVY);
            switch (r.error) {
                case "divide_by_zero": toast(getString(R.string.msg_divide_by_zero)); return;
                case "invalid_a": case "invalid_b":
toast(getString(R.string.msg_invalid_number)); return;
                default: toast(getString(R.string.msg_invalid_number)); return;
            }
        }

        firstOperand = r.value.toPlainString();
        currentOp = null;
        input.setLength(0);
        computed = showEquals;

        // Reset operator highlight after computation
        if (lastOperatorButton != null) {
            lastOperatorButton.setAlpha(1.0f);
            lastOperatorButton = null;
        }

        updateDisplay();

        // Animate result reveal
        if (showEquals) {
            txtResult.startAnimation(resultRevealAnim);
            performHapticFeedback(HapticFeedbackType.MEDIUM);
        }
    }

    private void clearAll() {
        input.setLength(0);
        firstOperand = null;
        currentOp = null;
        computed = false;

        // Reset operator highlight
        if (lastOperatorButton != null) {
            lastOperatorButton.setAlpha(1.0f);
            lastOperatorButton = null;
        }

        updateDisplay();
    }

    private void updateDisplay() {
        if (firstOperand == null && currentOp == null) {
            txtExpression.setText("");
        } else if (currentOp == null) {
            txtExpression.setText(Calculator.formatStringSafe(firstOperand));
        } else {
            String b = (input.length() > 0) ? Calculator.sanitizeNumber(input.toString()) : "";
            String expr = (firstOperand == null ? "" :
Calculator.formatStringSafe(firstOperand))
                    + (currentOp == null ? "" : " " + currentOp)
                    + (b.isEmpty() ? "" : " " + Calculator.formatStringSafe(b));
            txtExpression.setText(expr);
        }

        if (computed) {
            txtResult.setText(String.format(getString(R.string.result_equals),
Calculator.formatStringSafe(firstOperand)));
            txtResult.setAlpha(1.0f);
        } else if (currentOp == null) {
            String disp = (input.length() > 0) ? Calculator.formatStringSafe(input.toString())
                    : (firstOperand != null ? Calculator.formatStringSafe(firstOperand) : "0");
            txtResult.setText(disp);
            txtResult.setAlpha(1.0f);
        } else {
            // Show live preview when operator is set and user is typing second operand
            if (input.length() > 0) {
                try {
                    String aStr = (firstOperand != null) ? firstOperand : "0";
                    Calculator.Result preview = Calculator.compute(aStr, input.toString(),
currentOp);
                    if (preview.ok) {
                        // Show preview with "= " prefix and dimmed
                        txtResult.setText("= " + Calculator.format(preview.value));
                        txtResult.setAlpha(0.5f);
                    } else {
                        // Show current input if preview fails
                        txtResult.setText(Calculator.formatStringSafe(input.toString()));
                        txtResult.setAlpha(1.0f);
                    }
                } catch (Exception e) {
                    // Fallback to showing current input
                    txtResult.setText(Calculator.formatStringSafe(input.toString()));
                    txtResult.setAlpha(1.0f);
                }
            } else {
                txtResult.setText("");
                txtResult.setAlpha(1.0f);
            }
        }
    }

    private void toast(String msg) { Toast.makeText(this, msg, Toast.LENGTH_SHORT).show(); }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        handler.removeCallbacks(switchTask);
    }
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/root"
    android:layout_width="match_parent"
```

```xml
    android:layout_height="match_parent"
    android:background="@android:color/transparent">

    <!-- Background image 1: bottom layer -->
    <ImageView
        android:id="@+id/bgImage"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:src="@drawable/bg_leaves"
        android:scaleType="centerCrop"
        android:contentDescription="@string/desc_bg_decorative"
        android:importantForAccessibility="no"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <!-- Background image 2: crossfade target (initially hidden) -->
    <ImageView
        android:id="@+id/bgImage2"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:alpha="0"
        android:scaleType="centerCrop"
        android:contentDescription="@string/desc_bg_decorative"
        android:importantForAccessibility="no"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <!-- Display panel: frosted glass card -->
    <FrameLayout
        android:id="@+id/displayCard"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginTop="@dimen/display_top_margin"
        android:background="@drawable/glass_panel_outline"
        android:clipToOutline="true"
        android:elevation="4dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHeight_percent="@dimen/display_height_percent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintWidth_percent="0.92">

        <!-- Blur source (decorative) -->
        <ImageView
            android:id="@+id/ivBlur"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:scaleType="centerCrop"
            android:contentDescription="@string/desc_bg_decorative"
            android:importantForAccessibility="no" />

        <!-- Glass tint overlay -->
        <View
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="@drawable/glass_tint_overlay" />

        <!-- Content stack -->
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:gravity="end|bottom"
            android:orientation="vertical"
            android:paddingLeft="@dimen/display_padding_horizontal"
            android:paddingTop="@dimen/display_padding_vertical"
            android:paddingRight="@dimen/display_padding_horizontal"
            android:paddingBottom="@dimen/display_padding_vertical">

            <TextView
                android:id="@+id/txtExpression"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginBottom="4dp"
                android:ellipsize="end"
                android:gravity="end"
                android:maxLines="1"
                android:textColor="@color/textPrimaryDim"
                android:textSize="@dimen/expression_text_size"
                android:textStyle="bold" />

            <TextView
                android:id="@+id/txtResult"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginBottom="12dp"
                android:gravity="end"
                android:text="@string/initial_zero"
                android:textColor="@color/textPrimary"
                android:textSize="@dimen/result_text_size"
                android:textStyle="bold"
                android:singleLine="true"
                android:ellipsize="end"
                android:maxLines="1"
                android:contentDescription="@string/desc_display" />

            <View
                android:layout_width="match_parent"
                android:layout_height="1dp"
                android:background="@color/dividerThin" />
        </LinearLayout>
    </FrameLayout>

    <!-- Key grid container -->
    <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/keysBox"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:elevation="4dp"
        app:layout_constraintTop_toBottomOf="@id/displayCard"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintWidth_percent="0.92"
        app:layout_constraintDimensionRatio="4:5">

        <GridLayout
            android:id="@+id/grid"
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:alignmentMode="alignMargins"
            android:columnCount="4"
            android:rowCount="5"
            android:useDefaultMargins="true"
            android:padding="@dimen/grid_padding"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent">

            <TextView style="@style/CalcKey.CircleAccent" android:text="@string/key_clear"
android:contentDescription="@string/key_clear" />
            <TextView style="@style/CalcKey.CircleAccent" android:text="@string/key_backspace"
android:contentDescription="@string/key_backspace" />
            <TextView style="@style/CalcKey.CircleAccent" android:text="@string/key_percent"
android:contentDescription="@string/key_percent" />
            <TextView style="@style/CalcKey.CircleAccent" android:text="@string/key_divide"
android:contentDescription="@string/key_divide" />

            <TextView style="@style/CalcKey.Circle" android:text="@string/key_7"
android:contentDescription="@string/key_7" />
            <TextView style="@style/CalcKey.Circle" android:text="@string/key_8"
android:contentDescription="@string/key_8" />
            <TextView style="@style/CalcKey.Circle" android:text="@string/key_9"
android:contentDescription="@string/key_9" />
            <TextView style="@style/CalcKey.CircleAccent" android:text="@string/key_mul"
android:contentDescription="@string/key_mul" />
```

```
            <TextView style="@style/CalcKey.Circle" android:text="@string/key_4"
android:contentDescription="@string/key_4" />
            <TextView style="@style/CalcKey.Circle" android:text="@string/key_5"
android:contentDescription="@string/key_5" />
            <TextView style="@style/CalcKey.Circle" android:text="@string/key_6"
android:contentDescription="@string/key_6" />
            <TextView style="@style/CalcKey.CircleAccent" android:text="@string/key_sub"
android:contentDescription="@string/key_sub" />

            <TextView style="@style/CalcKey.Circle" android:text="@string/key_1"
android:contentDescription="@string/key_1" />
            <TextView style="@style/CalcKey.Circle" android:text="@string/key_2"
android:contentDescription="@string/key_2" />
            <TextView style="@style/CalcKey.Circle" android:text="@string/key_3"
android:contentDescription="@string/key_3" />
            <TextView style="@style/CalcKey.CircleAccent" android:text="@string/key_add"
android:contentDescription="@string/key_add" />

            <TextView style="@style/CalcKey.Circle" android:text="@string/key_plusminus"
android:contentDescription="@string/key_plusminus" />
            <TextView style="@style/CalcKey.Circle" android:text="@string/key_0"
android:contentDescription="@string/key_0" />
            <TextView style="@style/CalcKey.Circle" android:text="@string/key_dot"
android:contentDescription="@string/key_dot" />
            <TextView style="@style/CalcKey.CircleAccentBold" android:text="@string/key_equals"
android:contentDescription="@string/key_equals" />
        </GridLayout>
    </androidx.constraintlayout.widget.ConstraintLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

strings.xml

```
<resources>
    <string name="app_name">COMP1786 Logbook Ex1 - Calculator</string>

    <!-- Key labels -->
    <string name="key_clear">C</string>
    <string name="key_backspace">□</string>
    <string name="key_percent">%</string>
    <string name="key_divide">÷</string>
    <string name="key_mul">×</string>
    <string name="key_sub">−</string>
    <string name="key_add">+</string>
    <string name="key_plusminus">±</string>
    <string name="key_dot">.</string>
    <string name="key_equals">=</string>

    <string name="key_0">0</string>
    <string name="key_1">1</string>
    <string name="key_2">2</string>
    <string name="key_3">3</string>
    <string name="key_4">4</string>
    <string name="key_5">5</string>
    <string name="key_6">6</string>
    <string name="key_7">7</string>
    <string name="key_8">8</string>
    <string name="key_9">9</string>

    <!-- User messages -->
    <string name="msg_enter_second_number">Please enter the second number</string>
    <string name="msg_divide_by_zero">Cannot divide by 0</string>
    <string name="msg_invalid_number">Invalid number</string>
    <string name="msg_key_not_supported">Key not supported: %1$s</string>
    <string name="msg_result_copied">Result copied to clipboard</string>
    <string name="msg_expression_copied">Expression copied to clipboard</string>
    <string name="msg_nothing_to_copy">Nothing to copy</string>

    <!-- Accessibility -->
    <string name="desc_bg_decorative"/> <!-- decorative background -->
    <string name="desc_display">Result display area</string>

    <!-- Result format -->
    <string name="result_equals">=%1$s</string>
    <string name="initial_zero">0</string>
</resources>
```

styles.xml

```
<resources>

    <!-- Base: square cell, centered text -->
    <style name="CalcKeyBase">
        <item name="android:layout_width">0dp</item>
        <item name="android:layout_height">0dp</item>
        <item name="android:layout_rowWeight">1</item>
        <item name="android:layout_columnWeight">1</item>
        <item name="android:gravity">center</item>
        <item name="android:textSize">@dimen/key_text_size</item>
        <item name="android:textStyle">bold</item>
        <item name="android:minHeight">0dp</item>
        <item name="android:minWidth">0dp</item>
        <!-- Accessibility: ensure focusable and min touch target handled by parent or keys -->
    </style>

    <!-- Circular glass button -->
    <style name="CalcKey.Circle" parent="CalcKeyBase">
        <item name="android:textColor">@color/keyTextDim</item>
        <item name="android:background">@drawable/circle_glass</item>
        <item name="android:foreground">@drawable/ripple_glass</item>
        <item name="android:focusable">true</item>
    </style>

    <!-- Circular accent button (green) -->
    <style name="CalcKey.CircleAccent" parent="CalcKeyBase">
        <item name="android:textColor">@android:color/white</item>
        <item name="android:background">@drawable/circle_accent</item>
        <item name="android:foreground">@drawable/ripple_glass_accent</item>
        <item name="android:focusable">true</item>
    </style>

    <!-- Emphasized equals button -->
    <style name="CalcKey.CircleAccentBold" parent="CalcKeyBase">
        <item name="android:textColor">@android:color/white</item>
        <item name="android:background">@drawable/circle_accent_bold</item>
        <item name="android:foreground">@drawable/ripple_glass_accent</item>
        <item name="android:focusable">true</item>
    </style>

</resources>
```

Justification

1. Overall architecture

The app consists of a single Activity (MainActivity), a pure Java engine class (Calculator), plus Android layouts and XML resources.

The computation logic is fully separated from the UI:

- Calculator is pure Java, not dependent on Android → easy to test and reuse.
- MainActivity only holds UI state, wires events from the keypad, and updates the display.

2. Calculator.java – Core engine

Goal: a pure function calculator, independent of Android.

Main methods

Method	Description
sanitizeNumber(String s)	Normalizes user input: trims spaces, handles . and -., removes trailing ., adds leading 0 if starting with ., strips leading zeros, splits integer/fraction parts, handles negative sign; returns a “clean” string for safe BigDecimal parsing
parse(String s)	Calls sanitizeNumber and then new BigDecimal(...); may throw NumberFormatException if the input is invalid
compute(String aStr, String bStr, String op)	Parses a and b using parse, sets error codes on failure (invalid_a, invalid_b, missing_b); switches on operator +, −, ×, ÷ (supports multiple symbol variants); division uses scale = 10 and RoundingMode.HALF_UP to avoid non-terminating division errors; returns Result.ok(value) or Result.error(code)
format(BigDecimal)	Calls stripTrailingZeros; if number is large or in E notation, uses toEngineeringString; otherwise manually inserts , as thousands separators for the integer part (up to 15 digits, beyond that falls back to engineering notation)
formatStringSafe(String s)	Attempts to parse and format a numeric string; on parse failure, returns the original string (or empty if null)

Key points:

- Logic is purely numeric, no UI concerns mixed in.
- Use of BigDecimal plus careful sanitization significantly reduces format and precision edge cases.

3. MainActivity.java – UI controller + state machine

MainActivity acts as the UI controller and manages the calculator state.

3.1 Views, animations, and haptics

Core views:

View	Role
txtExpression	Displays the expression (A op B)
txtResult	Displays the result or live preview
grid	GridLayout containing the keypad
bg1, bg2	Two ImageViews used for the animated background
ivBlur	ImageView used as the blur source for glass effect

In onCreate, the activity loads animations: numberEntryAnim, resultRevealAnim, errorShakeAnim, clearRotateAnim, buttonPressAnim, buttonReleaseAnim. It also initializes a Vibrator instance for haptic feedback.

3.2 Background slideshow + blur

Background slideshow is driven by BACKGROUNDS[], a Handler, and a Runnable called switchTask.

- Every 5 seconds, crossfadeToNext() is called.
- Two ImageViews (bg1 and bg2) are crossfaded by adjusting their alpha values.
- ivBlur is set to the same drawable; on API 31+ it applies RenderEffect.createBlurEffect to create a blur/glass feel.

Lifecycle handling:

- onStart() posts switchTask via postDelayed.
- onStop() and onDestroy() both removeCallbacks(switchTask).

This is lifecycle-aware and prevents background handlers from running while the activity is not visible, avoiding leaks.

3.3 Haptic feedback

Haptics are abstracted via an enum:

Enum value	**Purpose**	**Default duration**
LIGHT	Number buttons and casual interactions	10 ms
MEDIUM	Operators and copy actions	20 ms
HEAVY	Errors and exceptional conditions	50 ms

The code checks hasVibrator(). On Android O+ it uses VibrationEffect.createOneShot; on legacy versions it falls back to vibrator.vibrate(ms).

3.4 Wiring the keypad

wireKeysFromGrid() iterates over all children of the GridLayout.

For each TextView child:

- Assigns an OnClickListener.
- Plays the press/release animations.
- Calls onKey(btn.getText().toString()).

Because it doesn’t hard-code button IDs, changing the keypad layout is easier to maintain.

3.5 Calculator state logic

Control flow:

onKey(String key) maps the label (from strings.xml: C, “delete one, %, =, ±, +, −, ×, ÷, ., digits) to corresponding behavior and dispatches to:

- clearAll
- backspace
- percent
- equalsCompute
- toggleSign
- setOperator
- appendDot
- appendDigit

Some key methods:

Method	**Main behavior**
appendDigit	If computed is true, calls clearAll() first; handles leading 0; appends digit to input, calls updateDisplay(), runs numberEntryAnim
setOperator	If firstOperand is null but input has content, moves input into firstOperand and clears input; if firstOperand, currentOp, and input are all set, calls equalsCompute(false) for chained operations; then sets currentOp, marks computed = false, and calls highlightOperatorButton(op)

backspace	If input has content, removes the last character; if input is empty but there is a currentOp, clears the operator; if computed is true, calls clearAll(); effectively a context-aware backspace
percent	If input is empty, does nothing; otherwise parses input as BigDecimal, divides by 100 with scale 10 and RoundingMode.HALF_UP, writes result back into input (no double shortcut)
equalsCompute	If currentOp is null, only sets computed = true and updateDisplay(); if operand B is missing, shows a toast, triggers errorShakeAnim and HEAVY haptic; otherwise calls Calculator.compute(aStr, bStr, currentOp), handles error codes via toasts; on success, stores result in firstOperand, clears currentOp and input, sets computed, resets operator highlight, calls updateDisplay(), and if showEquals = true, plays the result animation and MEDIUM haptic
clearAll	Resets all state variables and clears any operator highlight

3.6 updateDisplay() – state machine for expression/result

Expression (txtExpression):

State	**Display content**
firstOperand == null and currentOp == null	Empty string
Only firstOperand is set	formatStringSafe(firstOperand)
Both firstOperand and currentOp are set	"A op B" where A and B are formatted (B may be empty)

Result (txtResult):

State	**Display content**
computed == true	"= " + formatted(firstOperand) using the result_equals template
currentOp == null	If input has content, show formatted input; otherwise show firstOperand if non-null, else "0"
Entering second operand (currentOp != null and input.length > 0)	Try Calculator.compute(a, input, currentOp); if OK, show "= " + formatted(preview) with alpha 0.5 as a live preview; if not OK, show input as a normal formatted value
Has currentOp but input is empty	txtResult is set to empty

This gives you a real-time live preview of the result with multiple display states.

3.7 Copy to clipboard

Long press on txtResult:

- If text is empty or "0", shows "Nothing to copy".
- If text starts with "= ", strips the prefix before copying.
- Copies the cleaned text into ClipboardManager, shows a toast, and runs MEDIUM haptic.

Long press on txtExpression:

- If not empty, copies the full expression, plus toast and MEDIUM haptic feedback.

4. activity_main.xml – Layout + glassmorphism

The root is a ConstraintLayout.

Layering:

Layer	Content / purpose
Background	bgImage full-screen and bgImage2 full-screen (initially alpha=0) used for crossfade
Display panel	FrameLayout containing ivBlur (blur source), an overlay View with glass_tint_overlay, and an inner LinearLayout
Display content	Inside LinearLayout: txtExpression (top line, smaller, bold, ellipsized), txtResult (bottom line, larger, bold, ellipsized), and a thin View as divider
Keypad	A nested ConstraintLayout containing a 4x5 GridLayout; each key is a TextView styled via CalcKey.Circle*

The keypad includes keys: C, “delete one”, %, ÷, 7–9, ×, 4–6, −, 1–3, +, ±, 0, ., =.

Notable technical choices:

- layout_constraintWidth_percent and layout_constraintHeight_percent are used for percentage-based sizing, making the layout more responsive.
- Keys are circular TextViews with a glass-style background and ripple foreground.
- Both the display panel and keypad have elevation, giving them a card-like floating appearance over the animated background.

5. strings.xml

This file cleanly separates different string groups:

Group	Content
Key labels	Labels for keys: C, “delete one”, %, ÷, ×, −, +, ±, ., 0–9
User messages	Messages like invalid input, divide by zero, nothing to copy, etc.
Accessibility	Text for contentDescription attributes
Result template	"=%1$s" used to format the result with a leading =

This makes text handling standardized and ready for i18n and UI content control.

6. styles.xml

This file defines the keypad styling system.

Style	Role

CalcKeyBase	Sets size and weight for GridLayout cells (rowWeight, columnWeight = 1), centers text with gravity=center, defines textSize, bold, and removes hard-coded minWidth / minHeight
CalcKey.Circle	Uses background = circle_glass and foreground = ripple_glass, with a dim text color; used for numeric keys
CalcKey.CircleAccent	Uses background = circle_accent, textColor = white; used for operator / function keys (C, “delete one”, %, ÷, ×, −, +)
CalcKey.CircleAccentBold	Uses background = circle_accent_bold, textColor = white; reserved for the = key to make it visually dominant