INFORMATION TECHNOLOGY UNIVERSITY

Software Engineering Department - ITU

MD442: Programming Fundamentals Lab

Course Instructor: Usama Bin Shakeel	**Dated: 28/01/2025**
Teaching Assistant: Hammad Kamran	**Semester: Fall 2025**
Lab Engineer: Hateem Hassan	**Batch: BSCE2021**

Lab 1. Simple Calculator App in React Native

Name	Roll number	Report (out of 35)
Noor Fatima	BSCE21038	

Checked on: _____

Signature: _____

1.1 Objective

The objective of this lab assignment is to develop a simple portrait-only calculator app using **React Native** and the **mathjs** library. This project will help students:

- Learn to handle user input in a mobile app.
- Perform basic mathematical operations.
- Design a clean, responsive user interface.

1.2 Equipment and Component

Component Description	Value	Quantity
Computer	Available in lab	1

1.3 Conduct of Lab

1. Students are required to perform this experiment individually.
2. In case the lab experiment is not understood, the students are advised to seek help from the course instructor, lab engineers, assigned teaching assistants (TA) and lab attendants.

Components Required

React Native Components

The following core components will be used to build the user interface:

- **View:** Acts as a container for structuring the layout.

```
<View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
  <Text>Welcome to the Calculator App</Text>
</View>
```

- **Text:** Displays input, results, and button labels.

```
<Text style={{ fontSize: 24, fontWeight: 'bold', color: 'blue' }}>
  Welcome to the Calculator App
</Text>
```

- **Pressable:** Provides touch interactions for buttons.

```
<Pressable
  onPress={() => alert('Button Pressed!')}
  style={{ backgroundColor: '#007AFF', padding: 10, borderRadius: 5 }}>
  <Text style={{ color: '#fff' }}>Press Me</Text>
</Pressable>
```

- **StyleSheet:** Defines styling rules for the components.

```
import { StyleSheet, View, Text } from 'react-native';
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
  },
  text: {
    fontSize: 20,
    color: 'green',
  },
});
<View style={styles.container}>
  <Text style={styles.text}>Styled Component</Text>
</View>
```

- **SafeAreaView:** Ensures proper content positioning on iOS devices, avoiding screen notches and status bars.

```
<SafeAreaView style={{ flex: 1, backgroundColor: '#f5f5f5' }}>
  <Text>Content inside SafeAreaView</Text>
</SafeAreaView>
```

Mathjs Library

- **Purpose:** To handle mathematical operations and evaluate expressions accurately.
- **Installation Command:**

```
npm install mathjs
```

- **Usage:**

```
import { evaluate } from 'mathjs';

const result = evaluate("2 + 3 * 4");
```

Lock the orientation

- **Open `android/app/src/main/AndroidManifest.xml` and find the `<activity>` tag inside `<application>`.**

```
<activity
   android:name=".MainActivity"
   android:label="@string/app_name"
   android:screenOrientation="portrait">
</activity>
```

Project Structure:

To ensure clean and organized code, the project will follow this folder structure:

```
/calculator-app
├── App.js              # Main entry point
├── components/
│   ├── Button.js       # Button component
│   ├── Display.js      # Display component
│   └── styles.js       # Styles for components
└── package.json        # Project dependencies
```

Lab Tasks

Task 1

User Interface

- **Layout:**
  - The app should be designed for **portrait orientation only**.
  - Include a display section to show input and results.
  - Add interactive buttons for numbers and operations.
- **Buttons:**
  - Numbers: 0-9
  - Operators: +, -, *, /, %, +/-
  - Additional functionalities:
    - **Clear (C)** – Resets the calculator.
    - **Equals (=)** – Computes the entered expression.
- **Design Guidelines:**
  - The interface should be **user-friendly**, visually appealing, and intuitive to use.

Basic Functionalities

The calculator should support the following functionalities:

- **Number Input:** Users should be able to tap buttons to enter numbers (0–9).
- **Operations:** The app should support the following operations:
  - Addition (+)
  - Subtraction (-)
  - Multiplication (*)
  - Division (/)
  - Modulus (%)
- **Clear Functionality:** A button to reset the current input and clear results.
- **Evaluation:** An equals button to compute and display the result.

App.tsx

```
import React, {useState} from 'react';
import {View, Pressable, Text, StyleSheet, Dimensions} from 'react-native';
import styles from './styles';

interface DisplayProps {
  input: string;
  result: string;
}

const Display = ({ input, result }: DisplayProps) => {
  return (
    <View style={styles.displayContainer}>
      <Text style={styles.inputText}>{input || '0'}</Text>
      <Text style={styles.resultText}>{result}</Text>
```

```
    </View>
  );
};

export default Display;
```

button.tsx

```
import React, {useState} from 'react';
import {View, Pressable, Text, StyleSheet, Dimensions} from 'react-native';
import styles from './styles';

interface ButtonProps {
  label: string;
  onPress: (value: string) => void;
}

const Button = ({ label, onPress }: ButtonProps) => {
  return (
    <Pressable style={styles.button} onPress={() => onPress(label)}>
      <Text style={styles.buttonText}>{label}</Text>
    </Pressable>
  );
};

export default Button;
```

display.tsx

```
import React, {useState} from 'react';
import {View, Pressable, Text, StyleSheet, Dimensions} from 'react-native';
import styles from './styles';

interface DisplayProps {
  input: string;
  result: string;
}

const Display = ({ input, result }: DisplayProps) => {
  return (
    <View style={styles.displayContainer}>
      <Text style={styles.inputText}>{input || '0'}</Text>
      <Text style={styles.resultText}>{result}</Text>
    </View>
  );
};
```

```
export default Display;
```

styles.ts

```
import {View, Pressable, Text, StyleSheet, Dimensions} from 'react-native';

const { width: screenWidth } = Dimensions.get('window');
const buttonSize = screenWidth / 4 - 10;

const styles = StyleSheet.create({
    container: {
        flex: 1,
        backgroundColor: '#FFB6C1',
        justifyContent: 'flex-end',
        alignItems: 'center',
    },
    displayContainer: {
        width: '90%',
        padding: 20,
        backgroundColor: '#fff',
        borderRadius: 10,
        marginBottom: 20,
        elevation: 3,
    },
    inputText: {
        fontSize: 40,
        color: '#DC143C',
        textAlign: 'right',
        marginBottom: 10,
    },
    resultText: {
        fontSize: 40,
        fontWeight: 'bold',
        color: '#770737',
        textAlign: 'right',
    },
    button: {
        backgroundColor: '#DC143C',
        margin: 5,
        borderRadius: buttonSize / 2,
        width: buttonSize,
        height: buttonSize,
        justifyContent: 'center',
        alignItems: 'center',
    },
    buttonContainer: {
```

```
        flexDirection: 'row',
        flexWrap: 'wrap',
        justifyContent: 'center',
        alignItems: 'center',
        marginTop: 20,
    },
    buttonText: {
        fontSize: 20,
        color: '#fff',
    },
});

export default styles;
```

INFORMATION TECHNOLOGY UNIVERSITY

Assessment Rubric for Lab

Performance metric	CLO	Able to complete the task over 80% (4-5)	Able to complete the task 50-80% (2-3)	Able to complete the task below 50% (0-1)	Marks
1. Realization of experiment	1	Executes without errors excellent user prompts, good use of symbols, spacing in output. The testing has been completed.	Executes without errors, user prompts are understandable,minimum use of symbols or spacing in output. Some testing has been completed.	Does not execute due to syntax errors, runtime errors, user prompts are misleading or non- existent. No testing has been completed.	
2. Conducting experiment	1	Able to make changes and answer all questions.	Partially able to make changes and few incorrect answers.	Unable to make changes and answer all questions.	
3. Computer use	2	Document submission timely.	Document submission late.	Document submission not done.	
4. Teamwork	3	Actively engages and cooperates with other group member(s) in an effective manner.	Cooperates with other group member(s) in a reasonable manner but conduct can be improved.	Distracts or discourages other group members from conducting the experiment	
5. Laboratory safety and disciplinary rules	3	Code comments are added and do help the reader to understand the code.	Code comments are added and do not help the reader to understand the code.	Code comments are not added.	
6. Data collection	3	Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap.	Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables.	Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy.	
7. Data analysis	4	Solution is efficient, easy to understand, and maintain.	A logical solution that is easy to follow but it is not the most efficient.	A difficult and inefficient solution.	
Total (out of 35):					