|  |  |  |  |
| --- | --- | --- | --- |
| TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP  THÀNH PHỐ HỒ CHÍ MINH |  | **ĐỀ THI KIỂM TRA TK LẦN 1** | |
| **KHOA CÔNG NGHỆ THÔNG TIN** |  | **Môn thi:** | **Mobile Programming** |
|  |  | **Lớp/Lớp học phần:** | **DHKTPM18A** |
|  |  | **Ngày thi:** | **10/09/2025 (Tiết 1-3)** |
|  |  | Thời gian làm bài: 30 phút | |
|  |  | *(Không kể thời gian phát đề)* | |

*Họ và tên thí sinh ………………….…………………; MSSV: ………………*

**ĐỀ**

***Câu 1: Thiết kế giao diện người dùng (UI)***

1. Sử dụng các component cơ bản:

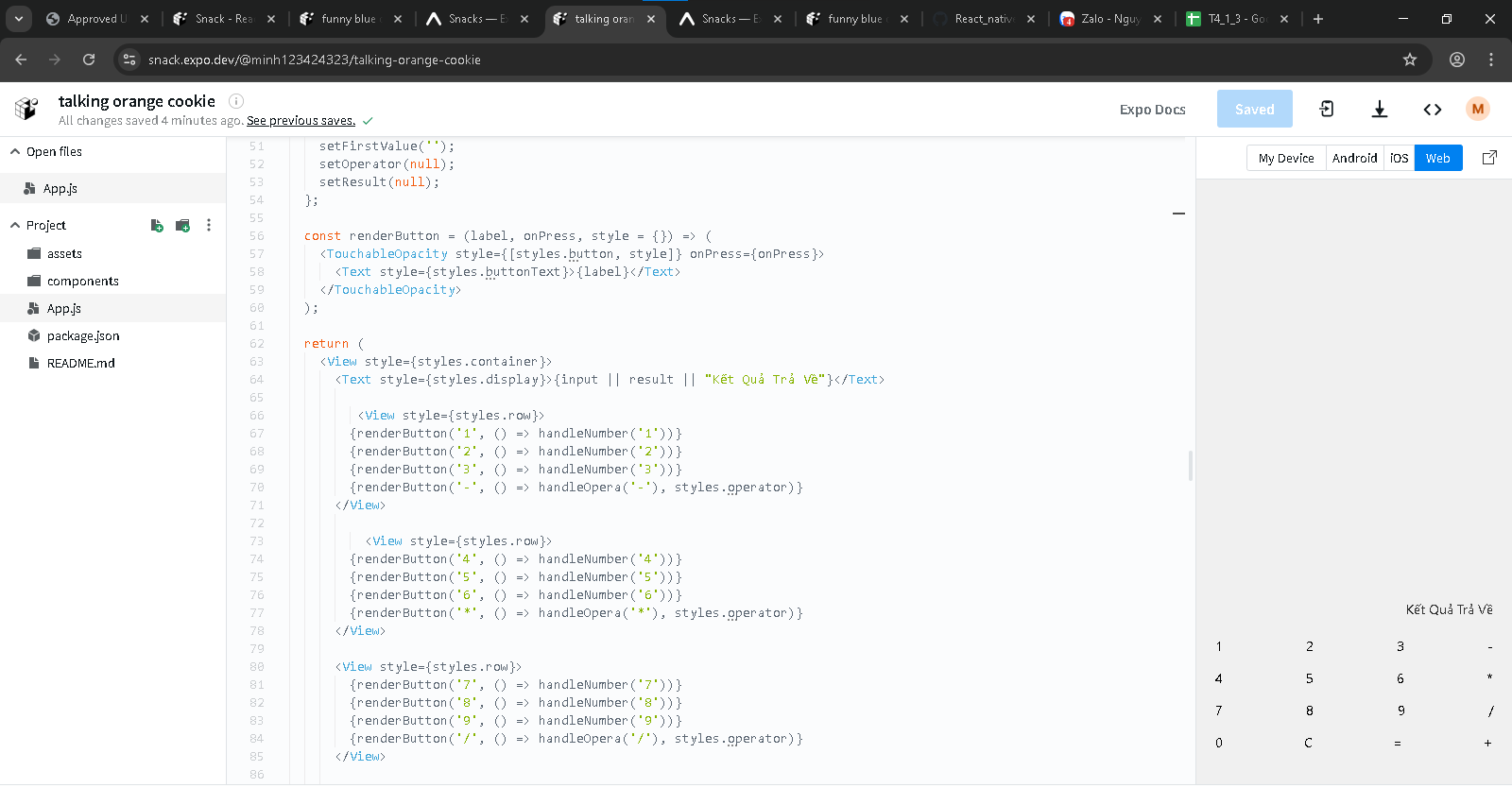
* View, Text, TextInput, Pressable hoặc TouchableOpacity

1. Cấu trúc UI:

Sắp xếp các component này thành một layout hợp lý, bao gồm màn hình hiển thị kết quả và các nút cho số từ 0-9, các phép toán cơ bản (+, -, \*, /), và nút "Xóa" (C).

***Yêu cầu:***

* Chụp hình minh chứng layout



***Câu 2: Viết logic xử lý***

1. Quản lý trạng thái:

Sử dụng hook useState của React để quản lý trạng thái của máy tính, bao gồm số hiện tại, giá trị đầu vào và phép toán đang thực hiện.

1. Xử lý đầu vào:

Viết hàm để thêm số vào màn hình khi người dùng nhấn nút số và xử lý logic khi nhấn các nút phép toán.

1. Thực hiện phép tính:

Viết hàm thực hiện các phép toán dựa trên toán tử đã chọn và hiển thị kết quả lên màn hình.

***Yêu cầu:***

* Chụp hình minh chứng layout

***Câu 3: Tách Component***

Tách thành các component riêng biệt và import vào Index

***Yêu cầu:***

* Chụp hình minh chứng code tách components

Code :   
import React, { useState } from 'react';

import { View, Text, TouchableOpacity, StyleSheet } from 'react-native';

const Calculator = () => {

const [input, setInput] = useState('');

const [firstValue, setFirstValue] = useState('');

const [operator, setOperator] = useState(null);

const [result, setResult] = useState(null);

const handleNumber = (num) => {

setInput((prev) => prev + num);

};

const handleOpera = (op) => {

if (input === '') return;

setFirstValue(input);

setOperator(op);

setInput('');

};

const calculateResult = () => {

const a = parseFloat(firstValue);

const b = parseFloat(input);

let res = 0;

switch (operator) {

case '+':

res = a + b;

break;

case '-':

res = a - b;

break;

case '\*':

res = a \* b;

break;

case '/':

res = b !== 0 ? a / b : 'Error';

break;

default:

res = 'Invalid';

}

setResult(res);

setInput(res.toString());

setOperator(null);

setFirstValue('');

};

const clearAll = () => {

setInput('');

setFirstValue('');

setOperator(null);

setResult(null);

};

const renderButton = (label, onPress, style = {}) => (

<TouchableOpacity style={[styles.button, style]} onPress={onPress}>

<Text style={styles.buttonText}>{label}</Text>

</TouchableOpacity>

);

return (

<View style={styles.container}>

<Text style={styles.display}>{input || result || "Kết Quả Trả Về"}</Text>

<View style={styles.row}>

{renderButton('1', () => handleNumber('1'))}

{renderButton('2', () => handleNumber('2'))}

{renderButton('3', () => handleNumber('3'))}

{renderButton('-', () => handleOpera('-'), styles.operator)}

</View>

<View style={styles.row}>

{renderButton('4', () => handleNumber('4'))}

{renderButton('5', () => handleNumber('5'))}

{renderButton('6', () => handleNumber('6'))}

{renderButton('\*', () => handleOpera('\*'), styles.operator)}

</View>

<View style={styles.row}>

{renderButton('7', () => handleNumber('7'))}

{renderButton('8', () => handleNumber('8'))}

{renderButton('9', () => handleNumber('9'))}

{renderButton('/', () => handleOpera('/'), styles.operator)}

</View>

<View style={styles.row}>

{renderButton('0', () => handleNumber('0'))}

{renderButton('C', clearAll, styles.clear)}

{renderButton('=', calculateResult, styles.equals)}

{renderButton('+', () => handleOpera('+'), styles.operator)}

</View>

</View>

);

};

const styles = StyleSheet.create({

container: {

flex: 1,

justifyContent: 'flex-end',

padding: 20,

backgroundColor: '#f0f0f0',

},

display: {

textAlign: 'right',

marginBottom: 20,

color: '#333',

},

row: {

flexDirection: 'row',

justifyContent: 'space-between',

marginBottom: 15,

},

});

export default Calculator;

------------------------- Hết -----------------------------