```
import * as ScreenOrientation from "expo-screen-orientation";
import { Text, StyleSheet, View, SafeAreaView } from "react-
native";
import { useEffect } from "react";

export default function App() {

ScreenOrientation.lockAsync(ScreenOrientation.OrientationLock
.LANDSCAPE_LEFT);

  // START
  const name = "Apple 🍎 (She/Her)";
  // END
  return (
    <View style={styles.container}>
      <SafeAreaView style={styles.safeArea}>
        {/* START */}
        <Text style={styles.welcomeText}>Kumusta</Text>
        <Text style={styles.subtitleText}>ako ay si</Text>
        {/* END */}
        <View style={styles.nameBox}>
          <Text style={styles.nameText}>{name}</Text>
        </View>
      </SafeAreaView>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "red",
    alignItems: "center",
    justifyContent: "center",
  },
  safeArea: {
    flex: 1,
    width: "100%",
    height: "100%",
  },
  welcomeText: {
    fontSize: 90,
    textTransform: "uppercase",
    fontWeight: "bold",
    color: "white",
    textAlign: "center",
  },
  subtitleText: {
    fontSize: 30,
    textTransform: "uppercase",
    fontWeight: "bold",
    color: "white",
    marginBottom: 20,
    textAlign: "center",
  },
  nameBox: {
    width: "100%",
    height: "55%",
    backgroundColor: "white",
    borderRadius: 5,
    justifyContent: "center",
  },
  nameText: {
    fontSize: 60,
    textAlign: "center",
    fontWeight: "bold",
  },
});

// START

import {
  View,
  Text,
  StyleSheet,
  TouchableOpacity,
  SafeAreaView,
  useWindowDimensions,
} from "react-native";
import { useState, useEffect } from "react";

// Board's state according to the sample provided
const board = [
  ["O", "O", "X"],
  ["X", "O", "O"],
  ["X", "X", "O"],
];

// Different monochromatic color schemes for the additional
feature
```

```
const colorSchemes = [
  {
    name: "Green",
    background: "#f0f8f0",
    oColor: "#228B22",
    xColor: "#32CD32",
    gridColor: "#0d4d0d",
  },
  {
    name: "Blue",
    background: "#f0f8ff",
    oColor: "#0066cc",
    xColor: "#3399ff",
    gridColor: "#003366",
  },
  {
    name: "Purple",
    background: "#f8f0ff",
    oColor: "#663399",
    xColor: "#9966cc",
    gridColor: "#330066",
  },
  {
    name: "Orange",
    background: "#fff8f0",
    oColor: "#cc6600",
    xColor: "#ff9933",
    gridColor: "#663300",
  },
  {
    name: "Pink",
    background: "#fff0f8",
    oColor: "#cc3366",
    xColor: "#ff6699",
    gridColor: "#660033",
  },
];

export default function App() {
  const [currentScheme, setCurrentScheme] = useState(0);
  const { width, height } = useWindowDimensions();

  // Responsive measurements
  const grid_size = Math.min(width * 0.8, height * 0.6);
  const cell_size = grid_size / 3;
  const border_width = Math.max(2, Math.min(width, height) *
0.01);
  const button_padding = Math.max(10, Math.min(width, height)
* 0.02);
  const button_font_size = Math.max(14, Math.min(width,
height) * 0.04);
  const cell_font_size = cell_size * 0.6;

  const cycleColors = () => {
    setCurrentScheme((prev) => (prev + 1) %
colorSchemes.length);
  };

  const currentColors = colorSchemes[currentScheme];

  // Function to get dynamic styles based on current color
scheme
  const getDynamicStyles = () => {
    return StyleSheet.create({
      containerWithBackground: {
        backgroundColor: currentColors.background,
      },
      gridWithBackground: {
        backgroundColor: currentColors.background,
      },
      cellBorderBottom: {
        borderBottomWidth: border_width,
        borderBottomColor: currentColors.gridColor,
      },
      cellBorderRight: {
        borderRightWidth: border_width,
        borderRightColor: currentColors.gridColor,
      },
      oText: {
        color: currentColors.oColor,
      },
      xText: {
        color: currentColors.xColor,
      },
    });
  };

  const dynamicStyles = getDynamicStyles();

  // Create responsive styles
  // Inside App() since it needs access to dynamic values
  const responsiveStyles = StyleSheet.create({
    container: {
      flex: 1,
      justifyContent: "center",
      alignItems: "center",
    },
    grid: {
      width: grid_size,
      height: grid_size,
      marginHorizontal: width * 0.1,
    },
    row: {
      flex: 1,
      flexDirection: "row",
    },
    cell: {
      flex: 1,
      justifyContent: "center",
      alignItems: "center",
    },
    cell_text: {
      fontSize: cell_font_size,
      fontWeight: "bold",
      color: "#000",
    },
    button: {
      position: "absolute",
      bottom: Math.max(50, height * 0.05),
      alignSelf: "center",
      padding: button_padding,
      backgroundColor: "#000",
      borderRadius: Math.max(6, Math.min(width, height) *
0.015),
    },
    buttonText: {
      fontSize: button_font_size,
      fontWeight: "bold",
      color: "#fff",
    },
  });

  return (
    <SafeAreaView
      style={[
        responsiveStyles.container,
        dynamicStyles.containerWithBackground,
      ]}
    >
      {/* Color scheme button */}
      <TouchableOpacity style={responsiveStyles.button}
onPress={cycleColors}>
        <Text style={responsiveStyles.buttonText}>
          Change to{" "}
          {colorSchemes[(currentScheme + 1) %
colorSchemes.length].name}
        </Text>
      </TouchableOpacity>

      <View style={[responsiveStyles.grid,
dynamicStyles.gridWithBackground]}>
        {/* Map through each row of the board */}
        {board.map((row, rowIndex) => (
          <View key={rowIndex} style={responsiveStyles.row}>
            {/* Map through each cell in the current row */}
            {row.map((cell, colIndex) => (
              <View
                key={colIndex}
                style={[
                  responsiveStyles.cell,
                  // Add bottom border to all cells except
the last row
                  rowIndex < 2 &&
dynamicStyles.cellBorderBottom,
                  // Add right border to all cells except the
last column
                  colIndex < 2 &&
dynamicStyles.cellBorderRight,
                ]}
              >
                <Text
                  style={[
                    responsiveStyles.cell_text,
                    // Green color for O's
                    cell === "O" && dynamicStyles.oText,
                    // Red color for X's
                    cell === "X" && dynamicStyles.xText,
                  ]}
                >
                  {cell}
                </Text>
              </View>
            ))}
          </View>
        ))}
      </View>
    </SafeAreaView>
  );
}

// END
// START

import React, { useState } from "react";
import {
  Dimensions,
  SafeAreaView,
  StatusBar,
  View,
  Text,
  TouchableOpacity,
  StyleSheet,
} from "react-native";

// Button dimensions based on screen width; Responsiveness
const buttonWidth = Math.floor(Dimensions.get("window").width
/ 4) - 10;
const longButtonWidth = buttonWidth * 2 + 10;

// Calculator button layout and styling
const buttonRows = [
  [
    { label: "AC", style: "buttonLightGrey" },
    { label: "+/-", style: "buttonLightGrey" },
    { label: "%", style: "buttonLightGrey" },
    { label: "/", style: "buttonBlue" },
  ],
  [
    { label: "7", style: "buttonDark" },
    { label: "8", style: "buttonDark" },
    { label: "9", style: "buttonDark" },
    { label: "x", style: "buttonBlue" },
  ],
  [
    { label: "4", style: "buttonDark" },
    { label: "5", style: "buttonDark" },
    { label: "6", style: "buttonDark" },
    { label: "-", style: "buttonBlue" },
  ],
  [
    { label: "1", style: "buttonDark" },
    { label: "2", style: "buttonDark" },
    { label: "3", style: "buttonDark" },
    { label: "+", style: "buttonBlue" },
  ],
  [
    { label: "0", style: "buttonDark", long: true },
    { label: ".", style: "buttonDark" },
    { label: "=", style: "buttonBlue" },
  ],
];

export default function App() {
  // State management for calculator functionality
  const [answerValue, setAnswerValue] = useState("0"); //
Current display value
  const [readyToReplace, setReadyToReplace] = useState(true);
// Flag to replace or append numbers
  const [memoryValue, setMemoryValue] = useState("0"); //
Stored value for operations
  const [operatorValue, setOperatorValue] = useState("0"); //
Current operator (+, -, x, /)
  const [isAC, setIsAC] = useState(true); // Toggle between
AC and CE button
  const [equationDisplay, setEquationDisplay] = useState("");
// Scientific calculator equation display

  // Utility functions to check input types
  const isNumber = (val) => /[0-9]/.test(val);
  const isOperator = (val) => ["+", "-", "x",
"/"].includes(val);

  /**
   * Handles number input logic
   * @param {string} num - The number to be added
   * @returns {string} - The new number value
   */
  const handleNumber = (num) => {
    if (readyToReplace) {
      setReadyToReplace(false);
      return num;
    } else {
      return answerValue === "0" ? num : answerValue + num;
    }
  };

  /**
   * Performs mathematical calculations based on stored
operator and values
   * @returns {string} - The calculated result
   */
  const calculateEquals = () => {
    const previous = parseFloat(memoryValue);
    const current = parseFloat(answerValue);
    let result = 0;

    switch (operatorValue) {
      case "+":
        result = previous + current;
        break;
      case "-":
        result = previous - current;
        break;
      case "x":
        result = previous * current;
        break;
      case "/":
        result = current !== 0 ? previous / current :
"Error";
        break;
      default:
        result = current;
    }
    return result.toString();
  };

  /**
   * Main button press handler - processes all calculator
inputs
   * @param {string} value - The button value pressed
   */
  const buttonPressed = (value) => {
    if (isNumber(value)) {
      const newValue = handleNumber(value);
      setAnswerValue(newValue);
      setIsAC(false); // Switch from AC to CE mode

      // Update equation display for scientific calculator
      if (operatorValue !== "0") {
        setEquationDisplay(`${memoryValue} ${operatorValue} $
{newValue}`);
      } else {
        setEquationDisplay(newValue);
      }
      return;
    }

    // Handle clear functions (AC/CE)
    if (value === "AC" || value === "CE") {
      if (value === "AC") {
        // AC: Reset entire calculator state to initial
values
        setAnswerValue("0");
        setMemoryValue("0");
        setOperatorValue("0");
        setEquationDisplay("");
        setIsAC(true);
      } else {
        // CE: Clear only current input, preserve operation
context
        if (readyToReplace) {
          // If ready to replace, just reset to 0
          setAnswerValue("0");
          setIsAC(true);
          setEquationDisplay("");
        } else {
          if (answerValue.length > 1) {
            // Remove last digit from current number
            const newValue = answerValue.slice(0, -1);
            setAnswerValue(newValue);
            // Update equation display to reflect the change
            if (operatorValue !== "0") {
              setEquationDisplay(`${memoryValue} $
{operatorValue} ${newValue}`);
            } else {
              setEquationDisplay(newValue);
            }
          } else {
            // If only one digit remains, reset to 0
            setAnswerValue("0");
            setIsAC(true);
            setReadyToReplace(false);
            setEquationDisplay("");
          }
        }
      }
      // Ensure readyToReplace is set for AC operations
      if (value === "AC") {
        setReadyToReplace(true);
      }
      return;
    }

    // Handle mathematical operators (+, -, x, /)
    if (isOperator(value)) {
      if (operatorValue !== "0") {
        // Chain operations: calculate previous operation
first
        const chained = calculateEquals();
        setMemoryValue(chained);
        setAnswerValue(chained);
        setEquationDisplay(`${chained} ${value}`);
      } else {
        // First operation: store current value and operator
        setMemoryValue(answerValue);
        setEquationDisplay(`${answerValue} ${value}`);
      }
      setReadyToReplace(true);
      setOperatorValue(value);
      return;
    }

    // Handle equals button (=)
    if (value === "=") {
      const result = calculateEquals();
      const finalEquation = `${memoryValue} ${operatorValue}
${answerValue} = ${result}`;
      setAnswerValue(result);
      setMemoryValue("0");
      setReadyToReplace(true);
      setOperatorValue("0");
      setEquationDisplay(finalEquation);
      return;
    }
    // Handle sign change (+/-)
    if (value === "+/-") {
      if (answerValue !== "0") {
        const newValue = (parseFloat(answerValue) *
-1).toString();
        setAnswerValue(newValue);
        // Update equation display to show sign change
        if (operatorValue !== "0") {
          setEquationDisplay(`${memoryValue} ${operatorValue}
${newValue}`);
        } else {
          setEquationDisplay(newValue);
        }
      }
      return;
    }

    // Handle percentage (%)
    if (value === "%") {
      const newValue = (parseFloat(answerValue) *
0.01).toString();
      setAnswerValue(newValue);
      if (operatorValue !== "0") {
        setEquationDisplay(`${memoryValue} ${operatorValue} $
{newValue}`);
      } else {
        setEquationDisplay(newValue);
      }
      return;
    }

    // Handle decimal point (.)
    if (value === ".") {
      if (!answerValue.includes(".")) {
        const newValue = answerValue + ".";
        setAnswerValue(newValue);
        setReadyToReplace(false); // Allow further decimal
input
        // Update equation display to show decimal addition
        if (operatorValue !== "0") {
          setEquationDisplay(`${memoryValue} ${operatorValue}
${newValue}`);
        } else {
          setEquationDisplay(newValue);
        }
      }
      return;
    }
  };

  /**
   * Determines button styling based on current state
   * @param {Object} btn - Button configuration object
   * @returns {Array} - Array of style objects
   */
  const getButtonStyle = (btn) => {
    // Highlight active operator button
    if (isOperator(btn.label) && operatorValue === btn.label)
{
      return [styles.button, styles[btn.style],
styles.activeOperator];
    }
    return [styles.button, styles[btn.style]];
  };

  return (
    <SafeAreaView style={styles.container}>
      <StatusBar barStyle="light-content" />

      {/* Scientific calculator equation display */}
      <View style={styles.equationContainer}>
        <Text style={styles.equationText}>{equationDisplay}</
Text>
      </View>

      {/* Main calculator result display */}
      <View style={styles.resultContainer}>
        <Text style={styles.resultText}>{answerValue}</Text>
      </View>

      {/* Calculator button grid */}
      {buttonRows.map((row, rowIndex) => (
        <View style={styles.row} key={rowIndex}>
          {row.map((btn, btnIndex) => (
            <TouchableOpacity
              key={btn.label}
              style={[
                ...getButtonStyle(btn),
                // Handle long button (zero button) styling
                btn.long
```

```
                  ? {
                      width: longButtonWidth,
                      height: buttonWidth,
                      borderRadius: buttonWidth / 2,
                      alignItems: "flex-start",
                      paddingLeft: 35,
                    }
                  : {
                      width: buttonWidth,
                      height: buttonWidth,
                      borderRadius: buttonWidth / 2,
                    },
              ]}
              activeOpacity={0.7}
              onPress={() =>
                buttonPressed(
                  btn.label === "AC" ? (isAC ? "AC" : "CE") :
btn.label
                )
              }
            >
              <Text style={styles.buttonText}>
                {btn.label === "AC" ? (isAC ? "AC" : "CE") :
btn.label}
              </Text>
            </TouchableOpacity>
          ))}
        </View>
      ))}
    </SafeAreaView>
  );
}

// Component styles
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "black",
    justifyContent: "flex-end",
  },
  equationContainer: {
    marginBottom: 10,
    alignItems: "flex-end",
    paddingRight: 20,
    minHeight: 40,
  },
  equationText: {
    color: "#888888",
    fontSize: 24,
    textAlign: "right",
  },
  resultContainer: {
    marginBottom: 20,
    alignItems: "flex-end",
    paddingRight: 20,
  },
  resultText: {
    color: "white",
    fontSize: 70,
    textAlign: "right",
  },
  row: {
    flexDirection: "row",
    justifyContent: "space-between",
    alignItems: "center",
    marginBottom: 15,
    paddingHorizontal: 5,
  },
  button: {
    justifyContent: "center",
    alignItems: "center",
  },
  buttonLightGrey: {
    backgroundColor: "#d4d4d2",
  },
  buttonDark: {
    backgroundColor: "#333333",
  },
  buttonBlue: {
    backgroundColor: "#2196f3",
  },
  activeOperator: {
    backgroundColor: "#ff9500",
  },
  buttonText: {
    color: "white",
    fontSize: 32,
    fontWeight: "bold",
  },
});
```

```
// END
import { NavigationContainer } from "@react-navigation/
native";
import { createStackNavigator } from "@react-navigation/
stack";
import { Cell, Section, TableView } from "react-native-
tableview-simple";
import React, { useState } from "react";
import {
  View,
  ScrollView,
  Text,
  StyleSheet,
  Image,
  TouchableOpacity,
  Alert,
  Dimensions,
  StatusBar,
} from "react-native";

// START
// Set up navigation stack
const Stack = createStackNavigator();
const { width } = Dimensions.get("window");

// Resaurant and menu data that will be mapped through
const restaurants = [
  // END
  {
    id: 1,
    title: "Joe's Gelato",
    tagline: "Dessert, Ice cream, £££",
    eta: "10-30",
    rating: 4.8,
    imgUri: require("./assets/ice-cream-header.jpg"),
    menu: [
      {
        title: "Gelato",
        contents: [
          {
            title: "Vanilla Bean",
            price: "£4.50",
            image: require("./assets/ice-cream-header.jpg"),
            inStock: true,
            description: "Classic vanilla with real vanilla
beans",
          },
          {
            title: "Chocolate Fudge",
            price: "£4.50",
            image: require("./assets/ice-cream-header.jpg"),
            inStock: true,
            description: "Rich chocolate with fudge pieces",
          },
          {
            title: "Strawberry",
            price: "£4.50",
            image: require("./assets/ice-cream-header.jpg"),
            inStock: false,
            description: "Fresh strawberry gelato",
          },
        ],
      },
      {
        title: "Drinks",
        contents: [
          {
            title: "Espresso",
            price: "£2.50",
            image: require("./assets/ice-cream-header.jpg"),
            inStock: true,
            description: "Single shot of premium coffee",
          },
          {
            title: "Sparkling Water",
            price: "£1.50",
            image: require("./assets/ice-cream-header.jpg"),
            inStock: true,
            description: "Refreshing sparkling water",
          },
        ],
      },
    ],
  },
  {
    id: 2,
    title: "Joe's Diner",
    tagline: "American, Burgers, ££",
    eta: "25-45",
    rating: 4.6,
    imgUri: require("./assets/burger-header.jpg"),
    menu: [
      {
        title: "Burgers",
        contents: [
          {
            title: "Classic Burger",
            price: "£12.99",
            image: require("./assets/burger-header.jpg"),
            inStock: true,
            description: "Beef patty with lettuce, tomato,
and special sauce",
          },
          {
            title: "Cheese Burger",
            price: "£14.99",
            image: require("./assets/burger-header.jpg"),
            inStock: true,
            description: "Classic burger with melted cheddar
cheese",
          },
          {
            title: "Veggie Burger",
            price: "£13.99",
            image: require("./assets/burger-header.jpg"),
            inStock: false,
            description: "Plant-based patty with fresh
vegetables",
          },
        ],
      },
      {
        title: "Sides",
        contents: [
          {
            title: "Crispy Fries",
            price: "£4.99",
            image: require("./assets/burger-header.jpg"),
            inStock: true,
            description: "Golden crispy french fries",
          },
          {
            title: "Onion Rings",
            price: "£5.99",
            image: require("./assets/burger-header.jpg"),
            inStock: true,
            description: "Beer-battered onion rings",
          },
        ],
      },
      {
        title: "Drinks",
        contents: [
          {
            title: "Classic Cola",
            price: "£2.99",
            image: require("./assets/burger-header.jpg"),
            inStock: true,
            description: "Refreshing cola drink",
          },
          {
            title: "Chocolate Milkshake",
            price: "£4.99",
            image: require("./assets/burger-header.jpg"),
            inStock: true,
            description: "Thick and creamy chocolate
milkshake",
          },
        ],
      },
    ],
  },
  // START
  {
    id: 3,
    title: "Tita's Kitchen",
    tagline: "Filipino, Traditional, ££",
    eta: "25-45",
    rating: 4.9,
    imgUri: require("./assets/lechon-header.jpg"),
    menu: [
      {
        title: "Main Dishes",
        contents: [
          {
            title: "Chicken Adobo",
            price: "£15.99",
            image: require("./assets/lechon-header.jpg"),
            inStock: true,
            description: "Tender chicken braised in soy sauce
and vinegar",
          },
          {
            title: "Sinigang na Baboy",
            price: "£16.99",
            image: require("./assets/lechon-header.jpg"),
            inStock: true,
            description: "Sour tamarind soup with pork and
vegetables",
          },
          {
            title: "Kare-kare",
            price: "£17.99",
            image: require("./assets/lechon-header.jpg"),
            inStock: false,
            description: "Peanut stew with oxtail and
vegetables",
          },
          {
            title: "Lechon Kawali",
            price: "£18.99",
            image: require("./assets/lechon-header.jpg"),
            inStock: true,
            description: "Crispy fried pork belly",
          },
        ],
      },
      {
        title: "Rice & Noodles",
        contents: [
          {
            title: "Garlic Rice",
            price: "£3.99",
            image: require("./assets/lechon-header.jpg"),
            inStock: true,
            description: "Fragrant rice cooked with garlic",
          },
          {
            title: "Pancit Canton",
            price: "£12.99",
            image: require("./assets/lechon-header.jpg"),
            inStock: true,
            description: "Stir-fried noodles with vegetables
and meat",
          },
          {
            title: "Chicken Arroz Caldo",
            price: "£11.99",
            image: require("./assets/lechon-header.jpg"),
            inStock: true,
            description: "Chicken rice porridge with ginger",
          },
        ],
      },
      {
        title: "Desserts",
        contents: [
          {
            title: "Halo-halo",
            price: "£6.99",
            image: require("./assets/lechon-header.jpg"),
            inStock: true,
            description: "Mixed dessert with shaved ice and
sweet beans",
          },
          {
            title: "Leche Flan",
            price: "£5.99",
            image: require("./assets/lechon-header.jpg"),
            inStock: true,
            description: "Creamy caramel custard",
          },
          {
            title: "Bibingka",
            price: "£4.99",
            image: require("./assets/lechon-header.jpg"),
            inStock: false,
            description: "Traditional rice cake with
coconut",
          },
        ],
      },
      {
        title: "Drinks",
        contents: [
          {
            title: "Calamansi Juice",
            price: "£3.99",
            image: require("./assets/lechon-header.jpg"),
            inStock: true,
            description: "Refreshing citrus juice",
          },
          {
            title: "Buko Juice",
            price: "£4.99",
            image: require("./assets/lechon-header.jpg"),
            inStock: true,
            description: "Fresh young coconut juice",
          },
          {
            title: "Sago't Gulaman",
            price: "£3.99",
            image: require("./assets/lechon-header.jpg"),
            inStock: true,
            description: "Sweet drink with tapioca pearls and
jelly",
          },
        ],
      },
    ],
  },
  {
    id: 4,
    title: "Fuji Sushi",
    tagline: "Japanese, Sushi, £££",
    eta: "20-35",
    rating: 4.7,
    imgUri: require("./assets/sushi-header.jpg"),
    menu: [
      {
        title: "Sushi Rolls",
        contents: [
          {
            title: "California Roll",
            price: "£8.99",
            image: require("./assets/sushi-header.jpg"),
            inStock: true,
            description: "Crab, avocado, and cucumber roll",
          },
          {
            title: "Spicy Tuna Roll",
            price: "£9.99",
            image: require("./assets/sushi-header.jpg"),
            inStock: true,
            description: "Spicy tuna with cucumber",
          },
          {
            title: "Dragon Roll",
            price: "£12.99",
            image: require("./assets/sushi-header.jpg"),
            inStock: false,
            description: "Eel and avocado topped with tempura
shrimp",
          },
        ],
      },
      {
        title: "Nigiri",
        contents: [
          {
            title: "Salmon Nigiri",
            price: "£3.50",
            image: require("./assets/sushi-header.jpg"),
            inStock: true,
            description: "Fresh salmon over seasoned rice",
          },
          {
            title: "Tuna Nigiri",
            price: "£3.50",
            image: require("./assets/sushi-header.jpg"),
            inStock: true,
            description: "Premium tuna over seasoned rice",
          },
        ],
      },
    ],
  },
  {
    id: 5,
    title: "Pizza Palace",
    tagline: "Italian, Pizza, ££",
    eta: "30-50",
    rating: 4.5,
    imgUri: require("./assets/pizza-header.jpg"),
    menu: [
      {
        title: "Pizzas",
        contents: [
          {
            title: "Margherita",
            price: "£14.99",
            image: require("./assets/pizza-header.jpg"),
            inStock: true,
            description: "Classic tomato sauce, mozzarella,
and basil",
          },
          {
            title: "Pepperoni",
            price: "£16.99",
            image: require("./assets/pizza-header.jpg"),
            inStock: true,
            description: "Spicy pepperoni with melted
cheese",
          },
          {
            title: "Quattro Formaggi",
            price: "£18.99",
            image: require("./assets/pizza-header.jpg"),
            inStock: false,
            description: "Four cheese blend pizza",
          },
        ],
      },
      {
        title: "Pasta",
        contents: [
          {
            title: "Spaghetti Carbonara",
            price: "£12.99",
            image: require("./assets/pizza-header.jpg"),
            inStock: true,
            description: "Creamy pasta with pancetta and
parmesan",
          },
          {
            title: "Penne Arrabbiata",
            price: "£11.99",
            image: require("./assets/pizza-header.jpg"),
            inStock: true,
            description: "Spicy tomato sauce with garlic and
chili",
          },
        ],
      },
    ],
  },
];

// Card component for each restaurant on the home screen
const HomeScreenCell = ({
  title,
  tagline,
  eta,
  rating,
  imgUri,
  action,
  height = 320,
  backgroundColor = "transparent",
  highlightColor = "#ccc",
  ...props
}) => {
  return (
    <TouchableOpacity
      onPress={action}
      style={[styles.cell, { height, backgroundColor }]}
      activeOpacity={0.7}
      underlayColor={highlightColor}
      {...props}
    >
      <View style={styles.cellContentView}>
        {/* Restaurant header image */}
        <Image source={imgUri} style={styles.headerImage}
resizeMode="cover" />
        {/* Delivery time badge */}
        <View style={styles.etaBadge}>
          <Text style={styles.etaText}>{eta} mins</Text>
        </View>
        {/* Rating badge */}
        <View style={styles.ratingBadge}>
          <Text style={styles.ratingText}>★ {rating}</Text>
        </View>
        {/* Restaurant name and tagline */}
        <View style={styles.cardInfo}>
          <Text style={styles.title}>{title}</Text>
          <Text style={styles.tagline}>{tagline}</Text>
        </View>
      </View>
    </TouchableOpacity>
  );
```

```
};

// Menu item component
const MenuItem = ({ item, onPress }) => {
  // Handle press: show alert if out of stock, otherwise call
onPress
  const handlePress = () => {
    if (!item.inStock) {
      Alert.alert("Out of Stock", "This item is currently
unavailable.");
      return;
    }
    onPress(item);
  };

  return (
    <TouchableOpacity
      style={[styles.menuItem, !item.inStock &&
styles.menuItemDisabled]}
      onPress={handlePress}
      disabled={!item.inStock}
    >
      {/* Menu item image */}
      <Image source={item.image} style={styles.menuItemImage}
/>
      <View style={styles.menuItemContent}>
        <View style={styles.menuItemHeader}>
          {/* Menu item name */}
          <Text
            style={[
              styles.menuItemTitle,
              !item.inStock && styles.menuItemTitleDisabled,
            ]}
          >
            {item.title}
          </Text>
          {/* Menu item price */}
          <Text
            style={[
              styles.menuItemPrice,
              !item.inStock && styles.menuItemPriceDisabled,
            ]}
          >
            {item.price}
          </Text>
        </View>
        {/* Menu item description */}
        <Text
          style={[
            styles.menuItemDescription,
            !item.inStock &&
styles.menuItemDescriptionDisabled,
          ]}
        >
          {item.description}
        </Text>
        {/* Out of stock badge */}
        {!item.inStock && (
          <View style={styles.outOfStockBadge}>
            <Text style={styles.outOfStockText}>Out of
Stock</Text>
          </View>
        )}
      </View>
    </TouchableOpacity>
  );
};

const styles = StyleSheet.create({
  cell: {
    height: 320,
    marginBottom: 20,
  },
  cellContentView: {
    flex: 1,
    backgroundColor: "white",
    borderRadius: 16,
    shadowColor: "#000",
    shadowOffset: { width: 0, height: 4 },
    shadowOpacity: 0.15,
    shadowRadius: 8,
    elevation: 8,
    overflow: "hidden",
  },
  headerImage: {
    width: "100%",
    height: 200,
  },
  etaBadge: {
    position: "absolute",
    right: 16,
    top: 160,
    backgroundColor: "#4CAF50",
    borderRadius: 20,
    paddingHorizontal: 12,
    paddingVertical: 6,
    elevation: 4,
    shadowColor: "#000",
    shadowOffset: { width: 0, height: 2 },
    shadowOpacity: 0.2,
    shadowRadius: 4,
  },
  etaText: {
    fontWeight: "bold",
    fontSize: 14,
    color: "white",
  },
  ratingBadge: {
    position: "absolute",
    left: 16,
    top: 160,
    backgroundColor: "#FFD700",
    borderRadius: 20,
    paddingHorizontal: 12,
    paddingVertical: 6,
    elevation: 4,
    shadowColor: "#000",
    shadowOffset: { width: 0, height: 2 },
    shadowOpacity: 0.2,
    shadowRadius: 4,
  },
  ratingText: {
    fontWeight: "bold",
    fontSize: 14,
    color: "#333",
  },
  cardInfo: {
    padding: 16,
  },
  title: {
    fontWeight: "bold",
    fontSize: 24,
    marginBottom: 4,
    color: "#333",
  },
  tagline: {
    color: "#666",
    fontSize: 16,
  },
  menuItem: {
    flexDirection: "row",
    backgroundColor: "white",
    marginHorizontal: 16,
    marginVertical: 8,
    borderRadius: 12,
    padding: 12,
    shadowColor: "#000",
    shadowOffset: { width: 0, height: 2 },
    shadowOpacity: 0.1,
    shadowRadius: 4,
    elevation: 3,
  },
  menuItemDisabled: {
    opacity: 0.6,
  },
  menuItemImage: {
    width: 80,
    height: 80,
    borderRadius: 8,
    marginRight: 12,
  },
  menuItemContent: {
    flex: 1,
    justifyContent: "center",
  },
  menuItemHeader: {
    flexDirection: "row",
    justifyContent: "space-between",
    alignItems: "center",
    marginBottom: 4,
  },
  menuItemTitle: {
    fontWeight: "bold",
    fontSize: 18,
    color: "#333",
    flex: 1,
  },
  menuItemTitleDisabled: {
    color: "#999",
  },
  menuItemPrice: {
    fontWeight: "bold",
    fontSize: 16,
    color: "#4CAF50",
  },
  menuItemPriceDisabled: {
    color: "#999",
  },
  menuItemDescription: {
    fontSize: 14,
    color: "#666",
    lineHeight: 20,
  },
  menuItemDescriptionDisabled: {
    color: "#999",
  },
  outOfStockBadge: {
    backgroundColor: "#FF5252",
    borderRadius: 12,
    paddingHorizontal: 8,
    paddingVertical: 4,
    alignSelf: "flex-start",
    marginTop: 4,
  },
  outOfStockText: {
    color: "white",
    fontSize: 12,
    fontWeight: "bold",
  },
  sectionHeader: {
    backgroundColor: "#f8f9fa",
    paddingVertical: 12,
    paddingHorizontal: 16,
    borderBottomWidth: 1,
    borderBottomColor: "#e9ecef",
  },
  sectionHeaderText: {
    fontSize: 20,
    fontWeight: "bold",
    color: "#333",
  },
});

// Shows restaurants available
function HomeScreen({ navigation }) {
  return (
    <View style={{ flex: 1, backgroundColor: "#f5f5f5" }}>
      {/* Status bar styling */}
      <StatusBar barStyle="dark-content"
backgroundColor="#f5f5f5" />
      <ScrollView style={{ paddingHorizontal: 16 }}>
        <View style={{ paddingTop: 20, paddingBottom: 20 }}>
          {/* App title and subtitle */}
          <Text
            style={{
              fontSize: 28,
              fontWeight: "bold",
              color: "#333",
              marginBottom: 8,
            }}
          >
            Food Delivery
          </Text>
          <Text style={{ fontSize: 16, color: "#666",
marginBottom: 20 }}>
            Discover amazing restaurants near you
          </Text>
          {/* Render all restaurants */}
          {restaurants.map((restaurant) => (
            <HomeScreenCell
              key={restaurant.id}
              title={restaurant.title}
              tagline={restaurant.tagline}
              eta={restaurant.eta}
              rating={restaurant.rating}
              imgUri={restaurant.imgUri}
              action={() =>
                navigation.navigate("Menu", {
                  items: restaurant.menu,
                  restaurantName: restaurant.title,
                })
              }
            />
          ))}
        </View>
      </ScrollView>
    </View>
  );
}
```

```
// Shows the menu for a selected restaurant
function MenuScreen({ route, navigation }) {
  // Get menu items and restaurant name from navigation
params
  const { items = [], restaurantName = "Restaurant" } =
route.params || {};

  // Handle menu item press: show add-to-cart dialog
  const handleMenuItemPress = (item) => {
    Alert.alert(
      "Add to Cart",
      `Would you like to add "${item.title}" to your cart for
${item.price}?`,
      [
        {
          text: "Cancel",
          style: "cancel",
        },
        {
          text: "Add to Cart",
          onPress: () => {
            Alert.alert(
              "Success",
              `${item.title} has been added to your cart!`
            );
          },
        },
      ]
    );
  };

  return (
    <View style={{ flex: 1, backgroundColor: "#f5f5f5" }}>
      {/* Status bar styling */}
      <StatusBar barStyle="dark-content"
backgroundColor="#f5f5f5" />
      <ScrollView>
        {/* Render each menu section and its items */}
        {items.map((section, idx) => (
          <View key={idx}>
            <View style={styles.sectionHeader}>
              <Text style={styles.sectionHeaderText}
>{section.title}</Text>
            </View>
            {section.contents.map((item, i) => (
              <MenuItem key={i} item={item}
onPress={handleMenuItemPress} />
            ))}
          </View>
        ))}
      </ScrollView>
    </View>
  );
}

// Main app component with navigation setup
export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator
        screenOptions={{
          headerStyle: {
            backgroundColor: "#4CAF50",
          },
          headerTintColor: "#fff",
          headerTitleStyle: {
            fontWeight: "bold",
          },
        }}
      >
        {/* Home screen: list of restaurants */}
        <Stack.Screen
          name="Restaurants"
          component={HomeScreen}
          options={{ title: "Food Delivery" }}
        />
        {/* Menu screen: menu for selected restaurant */}
        <Stack.Screen
          name="Menu"
          component={MenuScreen}
          options={({ route }) => ({
            title: route.params?.restaurantName || "Menu",
          })}
        />
      </Stack.Navigator>
    </NavigationContainer>
  );
}

// END
```