**Thông tin sinh viên:**

* Họ tên: Nguyễn Quốc Bảo
* MSSV: 20110179

**Bài 01:** Cài đặt môi trường lập trình

* App.js

import React from "react";

import { NavigationContainer } from "@react-navigation/native";

import { createNativeStackNavigator } from "@react-navigation/native-stack";

import Introduce from "./pages/Introduce";

import HomePage from "./pages/HomePage";

const Stack = createNativeStackNavigator();

export default function App() {

  return (

    <NavigationContainer>

      <Stack.Navigator

        initialRouteName="Introduce"

        screenOptions={{

          headerShown: false,

        }}

      >

        <Stack.Screen name="Introduce" component={Introduce} />

        <Stack.Screen name="HomePage" component={HomePage} />

      </Stack.Navigator>

    </NavigationContainer>

  );

}

**Mô tả:**

* + Đầu tiên tạo Stack Navigator bằng createNativeStackNavigator() để tạo một navigator kiểu stack dùng để điều hướng qua lại giữa các trang.
  + Sử dụng component <NavigationContainer> để bao bọc lại navigator.
  + Sử dụng component <Stack.Naviagtor> để đại diện cho navigator dạng stack đã tạo. Trong component này sử dụng các thuộc tính như:
    - “initialRouteName” để khởi tạo màn hình đầu tiên được hiển thị khi khởi động ứng dụng. Ở đây đặt là “Introduce”, tức là khi khởi tạo ứng dụng sẽ hiển thị trang “Introduce” đầu tiên.
    - “screenOptions” để tùy chỉnh giao diện và hành vi cho màn hình. Ở đây, ta tùy chỉnh “headerShown: false” để ẩn header của màn hình.
  + Sử dụng component <Stack.Screen> để xác định từng màn hình trong Stack. Ở đây ta đưa hai Screen vào Navigator dạng Stack là “Introduce” và “HomePage”. Ở mỗi Sreen sẽ có các thuộc tính như:
    - “name” là để xác định tên.
    - “component” để chỉ định component nào sẽ được dùng để hiển thị lên Screen khi được kích hoạt.
* Introduce.js

import React, { useEffect } from "react";

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

import { StatusBar } from "expo-status-bar";

const Introduce = ({ navigation }) => {

  useEffect(() => {

    const timeoutId = setTimeout(() => {

      navigation.navigate("HomePage");

    }, 10000);

    return () => clearTimeout(timeoutId);

  }, []);

  return (

    <View style={styles.container}>

      <Text style={[styles.text, { fontSize: 24 }]}>

        Họ tên: Nguyễn Quốc Bảo

      </Text>

      <Text style={[styles.text, { fontSize: 20 }]}>MSSV: 20110179</Text>

      <StatusBar style="auto" />

    </View>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: "#fff",

    alignItems: "center",

    justifyContent: "center",

  },

  text: {

    fontWeight: 700,

    marginBottom: 10,

  },

});

export default Introduce;

**Mô tả:**

* + “Introduce” nhận “navigation” được truyền vào và được dùng để điều hướng sang trang khác.
  + “styles” là một hằng được tạo bằng “StyleSheet.create” dùng để định nghĩa các style được sử dụng.
  + Sử dụng component <View> để làm container chứa các component khác. Ở component này sử dụng thuộc tính “style” có giá trị được lấy từ “styles.container” để thiết lập giao diện cho <View> với các style đã được định nghĩa trong “styles.container”.
  + Sử dụng component <Text> để chứa dữ liệu dạng text. Ở hai component <Text> được sử dụng có thuộc tính “style” đều dùng để thiết lập giao diện cho <Text>.
  + Sử dụng component <StatusBar> để quản lý thanh trạng thái của thiết bị. Ở đây <StatusBar> có “style” với giá trị “auto” tức là sẽ tự động thiết lập màu cho thanh trạng thái tùy theo thiết lập màu của ứng dụng.
  + Sử dụng hàm “useEffect”, trong đó tạo timeout với giá trị là 10 giây để thực hiện đếm ngược 10 giây, khi đếm ngược hoàn tất sẽ gọi đến “navigation.navigate("HomePage")” để điều hướng sang trang “HomePage” với navigation được nhận được.
  + Hàm “useEffect” chỉ gọi một lần sau khi component được mount ([] là dependencies array rỗng), và hàm “clearTimeout” sẽ được gọi khi component bị unmount để tránh rò rỉ bộ nhớ.
* HomePage.js

import React, { useEffect } from "react";

import { StyleSheet, View, Text, BackHandler } from "react-native";

import { StatusBar } from "expo-status-bar";

const HomePage = () => {

    useEffect(() => {

        const backHandler = BackHandler.addEventListener(

          "hardwareBackPress",

          () => {

            return true;

          }

        );

        return () => backHandler.remove();

      }, []);

  return (

    <View style={styles.container}>

      <Text style={[styles.text]}>Chào mừng đến với trang chủ!</Text>

      <StatusBar style="auto" />

    </View>

  );

};

const styles = StyleSheet.create({

    container: {

      flex: 1,

      backgroundColor: "#fff",

      alignItems: "center",

      justifyContent: "center",

    },

    text: {

        fontWeight: 700,

        fontSize: 20,

    }

  });

export default HomePage;

**Mô tả:**

* + “styles” là một hằng được tạo bằng “StyleSheet.create” dùng để định nghĩa các style được sử dụng.
  + Sử dụng component <View> để làm container chứa các component khác. Ở component này sử dụng thuộc tính “style” có giá trị được lấy từ “styles.container” để thiết lập giao diện cho <View> với các style đã được định nghĩa trong “styles.container”.
  + Sử dụng component <Text> để chứa dữ liệu dạng text. Ở component <Text> được sử dụng có thuộc tính “style” được dùng để thiết lập giao diện cho <Text>.
  + Sử dụng component <StatusBar> để quản lý thanh trạng thái của thiết bị. Ở đây <StatusBar> có “style” với giá trị “auto” tức là sẽ tự động thiết lập màu cho thanh trạng thái tùy theo thiết lập màu của ứng dụng.
  + Sử dụng hàm “useEffect”, trong đó tạo “BackHandler” với sự kiện là “hardwareBackPress” để bắt được sự kiện khi người dùng nhấn nút quay lại của thiết bị, phương thức khi bắt được sự kiện sẽ trả về giá trị “true”, tức là khi người dùng nhấn nút quay lại của thiết bị thì sẽ bị vô hiệu hóa khả năng quay lại của ứng dụng, cách này dùng để ngăn chặn người dùng quay trở lại màn hình “Introduce”.
  + Hàm “useEffect” chỉ gọi một lần sau khi component được mount ([] là dependencies array rỗng), và gọi phương thức “backHandler.remove()” để xóa “BackHandler” được tạo khi component bị unmount để tránh rò rỉ bộ nhớ.
* Giao diện của ứng dụng
  + Màn hình “Introduce” (Sau 10 giây sẽ chuyển sang màn hình “HomePage”



* + Màn hình “HomePage”

