

MÔN HỌC: MOB306 – Lập trình Mobile đa nền tảng (REACT NATIVE)

GIÁO TRÌNH THỰC HÀNH

I. Giới thiệu

Cross platform (đa nền tảng) ngày càng phát triển mạnh, chiếm tỷ trọng dự án app mobile rất cao, do đó React Native cũng không ngừng phát triển với nhiều cải tiến mới về code cũng như thư viện phù hợp hơn nhằm đáp ứng nhu cầu thị trường app mobile.

Hơn thế nữa, sự ra đời React Hooks với những thay đổi lớn giúp code nhanh hơn, tối ưu hơn vì hướng function thay vì class như trước đây.

Chính vì lý do đó, giáo trình thực hành React Native sẽ giúp các bạn có cái nhìn mới hơn về React Native và góp phần giúp các bạn code tốt hơn cũng như theo kịp môn học đang HOT này.

II. Chuẩn đầu ra

- Khả năng code cơ bản React Native.
 - Cài đặt và kiến thức cơ bản (Lab1)
 - Thiết kế giao diện (Lab2)
 - Điều hướng Navigation, Firebase Authentication (Lab3)
- Làm việc với Realtime database với ứng dụng React Native (Lab4).
- Biết cách sử dụng authen với mạng xã hội Facebook và Google (Lab5).
- Kết thúc giáo trình bạn có thể viết được ứng dụng React Native với các chức năng quản lý cơ bản (Cụ thể là app quản lý sản phẩm).

III. Chuẩn bị:

- Chi tiết các thành phần cần chuẩn bị:
 - IDE: Visual Studio Code hoặc bất kì có hỗ trợ JS
 - Ngôn ngữ: JavaScript trên nền tảng Nodejs
 - Platform: expo-cli hoặc react-native-cli
- Hướng dẫn cài đặt môi trường develop: <https://reactnative.dev/docs/environment-setup>

Khuyến cáo nên dùng expo-cli : npm install -g expo-cli

LAB 1: CÁC KHÁI NIỆM VÀ COMPONENT CƠ BẢN

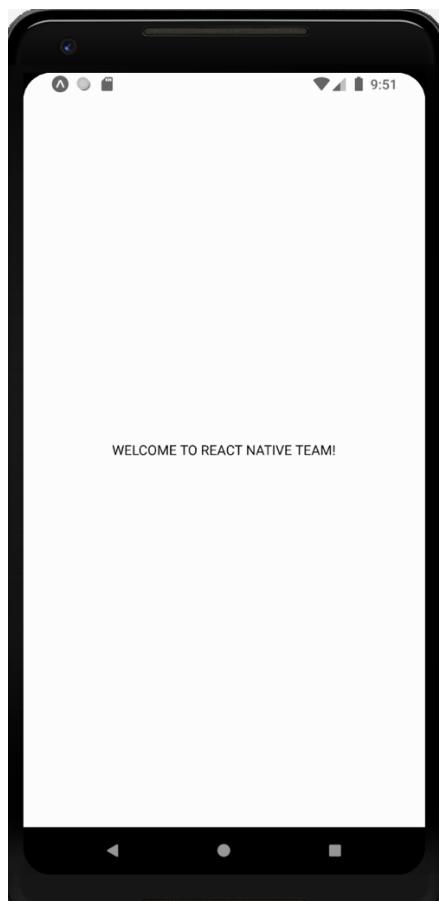
1. Chuẩn đầu ra

- Tạo project và tổ chức code.
- Hiểu được props, state, useEffect và tạo component, function
- Component cơ bản: View, Text, TextInput, Button, Alert

2. Nội dung các bài tập

Bài tập 1:

- *Mô tả bài tập:*
 - Tạo project react native đầu tiên
 - Tổ chức code phù hợp nhằm khắc phục tình trạng Internet chậm.
- *Yêu cầu thực hiện:*
 - Tạo project react native có tên MOB306-ReactNative trong thư mục làm việc.
 - Tạo các thư mục cho các LAB, bài tập phù hợp nhằm khắc phục tình trạng Internet chậm.
 - Tạo app đầu tiên với yêu cầu hiển thị thông báo như hình:



- Cấu hình chạy app đầu tiên.

- Hướng dẫn thực hiện:

1. Tạo project react native có tên MOB306-ReactNative:

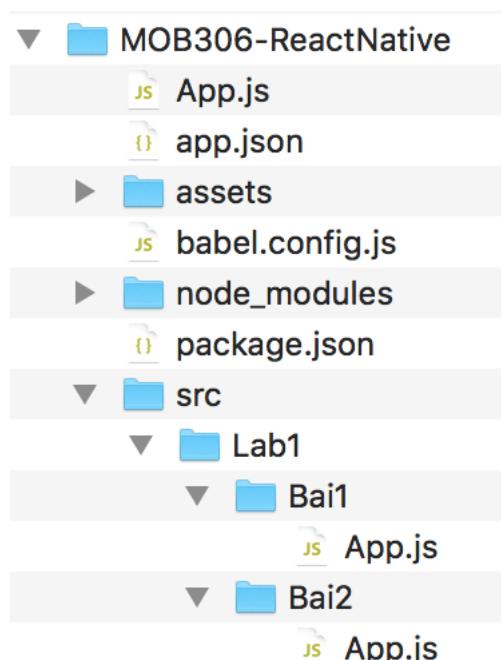
- Từ CMD đánh lệnh: **expo init MOB306-ReactNative**

```
lephamtuankiet — node /usr/local/bin/expo init MOB306-ReactNative — 80x24
[Les-MacBook-Pro:~ lephamtuankiet$ expo init MOB306-ReactNative
? Choose a template: (Use arrow keys)
    ---- Managed workflow ----
  > blank           a minimal app as clean as an empty canvas
  blank (TypeScript) same as blank but with TypeScript configuration
  tabs              several example screens and tabs using react-navigation
    ---- Bare workflow ----
  minimal          bare and minimal, just the essentials to get you started
  minimal (TypeScript) same as minimal but with TypeScript configuration
```

- Chọn Blank và chờ Expo download project (quá trình này tuỳ thuộc Internet).
- Chạy thử project, từ CMD đánh lệnh:

```
cd MOB306-ReactNative
npm start
```

2. Tạo các thư mục cho các LAB, bài tập theo hình và chép file App.js vào các thư mục Bai1, Bai2,...:



3. Tạo app: Open file App.js trong thư mục Bai1 và code như hình

```

JS App.js •
Users > lephamtuankiet > MOB306-ReactNative > App > Lab1 > Bai1 > JS App.js > ...
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 export default function App() {
5   return (
6     <View style={styles.container}>
7       <Text>WELCOME TO REACT NATIVE TEAM!</Text>
8     </View>
9   );
10 }
11
12 const styles = StyleSheet.create({
13   container: {
14     flex: 1,
15     backgroundColor: '#fff',
16     alignItems: 'center',
17     justifyContent: 'center',
18   },
19 });
20

```

4. Cấu hình chạy app: Open file App.js trong thư mục MOB306-ReactNative (project) và code như hình

```

import React from 'react';
import App from './src/Lab1/Bai1/App';
// './src/Lab1/Bai1/App' Thay đổi cho các bài khác

export default function Project() {
  return <App />;
}

```

Save file và xem kết quả.

Bài tập 2:

- *Mô tả bài tập:*
 - Tạo app hiển thị câu chúc mừng sinh viên.
 - Tạo app tính các phép toán.
- *Yêu cầu thực hiện:*
 1. Tạo app hiển thị câu chúc mừng sinh viên với tên sinh viên được truyền bằng props.
 2. Tạo app tính các phép toán + - * / hai số nhập vào từ InputText và hiển thị thông báo kết quả khi nhấn vào Button .
- *Hướng dẫn thực hiện:*
 1. Tạo app hiển thị câu chúc mừng sinh viên với tên sinh viên được truyền bằng props.
 - Viết component <Chucmung> có chứa props.name trong function App()

```

const Chucmung = (props) => {
  return (
    <View>
      <Text>Chúc mừng bạn {props.name}!</Text>
    </View>
  );
};

```

- Gọi component <Chucmung> trong return() của function App():

```
<Chucmung name="Kiet" />
```

2. Tạo app tính các phép toán + - * / hai số nhập vào:

- Import:

```

import React, { useState, useEffect } from 'react';
import { View, StyleSheet, TextInput, Button, Alert } from 'react-native';

```

- Khai báo State:

```

const [ n1, setN1 ] = useState(0);
const [ n2, setN2 ] = useState(0);
const [ tong, setTong ] = useState(null);
const [ hieu, setHieu ] = useState(null);
const [ tich, setTich ] = useState(null);
const [ thuong, setThuong ] = useState(null);

```

- Viết function calculate trong function App():

```

function calculate() {
  setTong(Number(n1) + Number(n2));
  setHieu(Number(n1) - Number(n2));
  setTich(Number(n1) * Number(n2));
  setThuong(Number(n1) / Number(n2));
}

```

- Dùng TextInput nhập 2 số:

```

<TextInput style={{ height: 40 }} placeholder="Nhập số thu 1" onChangeText={(text) => setN1(text)} />
<TextInput style={{ height: 40 }} placeholder="Nhập số thu 2" onChangeText={(text) => setN2(text)} />

```

- Nhấn vào Button gọi calculate() tính kết quả:

```
<Button title="Calculate" onPress={() => calculate()} />
```

- Dùng useEffect gọi Alert thông báo kết quả:

```

useEffect(
  () => {
    if (tong != null)
      Alert.alert(
        'Kết quả',
        'Tổng = ' + tong + '\n Hieu = ' + hieu + '\n Tich = ' + tich + '\n Thuong = ' + thuong
      );
  },
  [ tong ]
);

```

Bài tập 3:

- *Mô tả bài tập:*
 - Tạo app tính các phép toán phương trình bậc 2 với a, b, c nhập từ bàn phím
- *Yêu cầu thực hiện:*
 - Nhập a, b,c vào từ InputText
 - Viết function ptbac2() tính nghiệm phương trình bậc 2.
 - Hiển thị thông báo kết quả khi nhấn vào Button .
- *Hướng dẫn thực hiện:*
 - Tham khảo bài tập 2.
 - Thay function calculate() bằng function ptbac2(), tìm hiểu lại thuật toán tính phương trình bậc 2.
 - Thay nội dung hiển thị trong thông báo và State thay đổi trong useEffect().

LAB 2: THIẾT KẾ GIAO DIỆN SCREEN

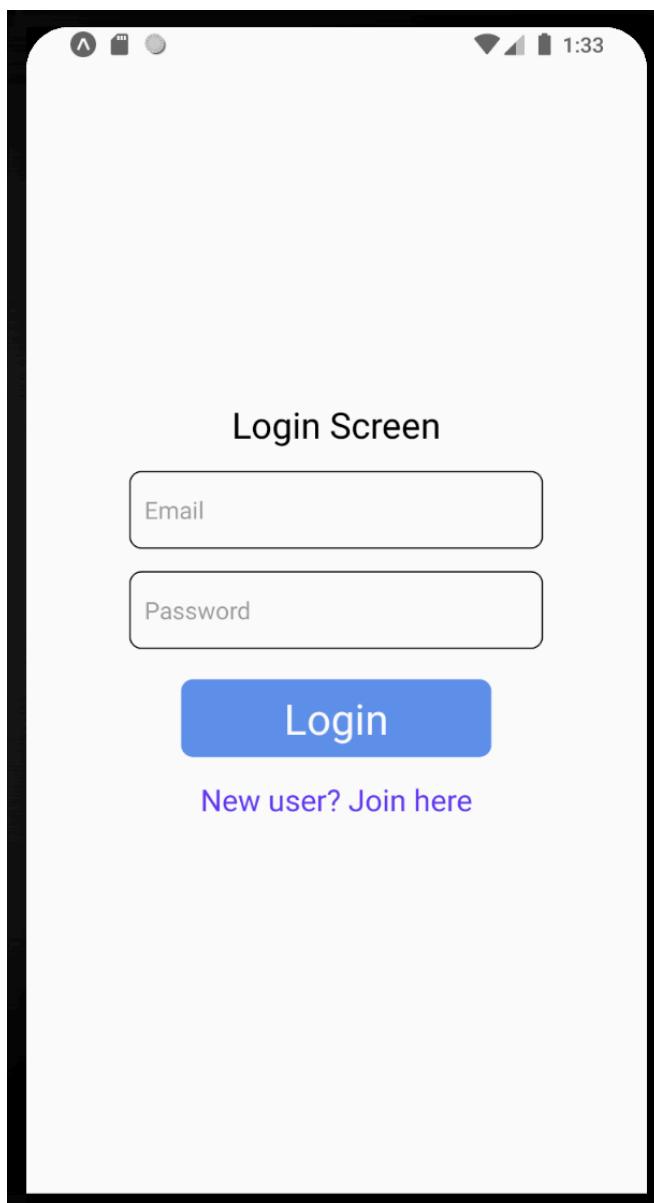
1. Chuẩn đầu ra

- Sử dụng Component cơ bản: Image, FlatList, TouchableOpacity
- Sử dụng Package swipeout hỗ trợ FlatList.
- Sử dụng StyleSheet

2. Nội dung các bài tập

Bài tập 1:

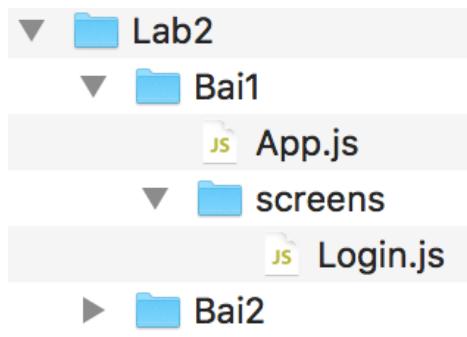
- *Mô tả bài tập:*
 - Thiết kế giao diện Login như hình sau:



- *Yêu cầu thực hiện:*
 - Tạo thư mục **screens** chứa file **Login.js**
 - Giao diện sử dụng các StyleSheet và các component cần thiết
 - Cấu hình chạy app sẽ xuất hiện giao diện vừa thiết kế

- Hướng dẫn thực hiện:

- Tạo thư mục **screens** chứa file **Login.js** như hình sau:



- Code file **Login.js**

```
import React, { useState } from 'react';
import { View, Text, StyleSheet, TouchableOpacity, TextInput, Alert, Dimensions } from 'react-native';

const Login = () => {
  const [ email, setEmail ] = useState('');
  const [ password, setPassword ] = useState('');
  return (
    <View style={styles.container}>
      <Text style={styles.text}> Login Screen </Text>
      <TextInput style={styles.input} value={email} placeholder="Email" onChangeText={(text) => setEmail(text)} />
      <TextInput
        style={styles.input}
        value={password}
        placeholder="Password"
        onChangeText={(text) => setPassword(text)}
        secureTextEntry={true}
      />
      <TouchableOpacity style={styles.buttonContainer} onPress={() => Alert.alert('ban vua nhan Login')}>
        <Text style={styles.buttonText}>Login</Text>
      </TouchableOpacity>
      <TouchableOpacity style={styles.navButton} onPress={() => Alert.alert('ban vua nhan Register')}>
        <Text style={styles.navButtonText}>New user? Join here</Text>
      </TouchableOpacity>
    </View>
  );
}
export default Login;
```

- Bổ sung style vào file Login.js:

```

const { width, height } = Dimensions.get('window');
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center'
  },
  text: {
    fontSize: 24,
    marginBottom: 10
  },
  navButton: {
    marginTop: 15
  },
  navButtonText: {
    fontSize: 20,
    color: '#6646ee'
  },
  input: {
    padding: 10,
    marginTop: 5,
    marginBottom: 10,
    width: width / 1.5,
    height: height / 15,
    fontSize: 16,
    borderRadius: 8,
    borderWidth: 1
  },
  buttonContainer: {
    marginTop: 10,
    width: width / 2,
    height: height / 15,
    backgroundColor: '#6495ed',
    padding: 10,
    alignItems: 'center',
    justifyContent: 'center',
    borderRadius: 8
  },
  buttonText: {
    fontSize: 28,
    color: '#ffffff'
  }
});

```

- Cấu hình chạy app: Open file App.js trong thư mục MOB306-ReactNative (project) và code như hình (Save file và xem kết quả).

```

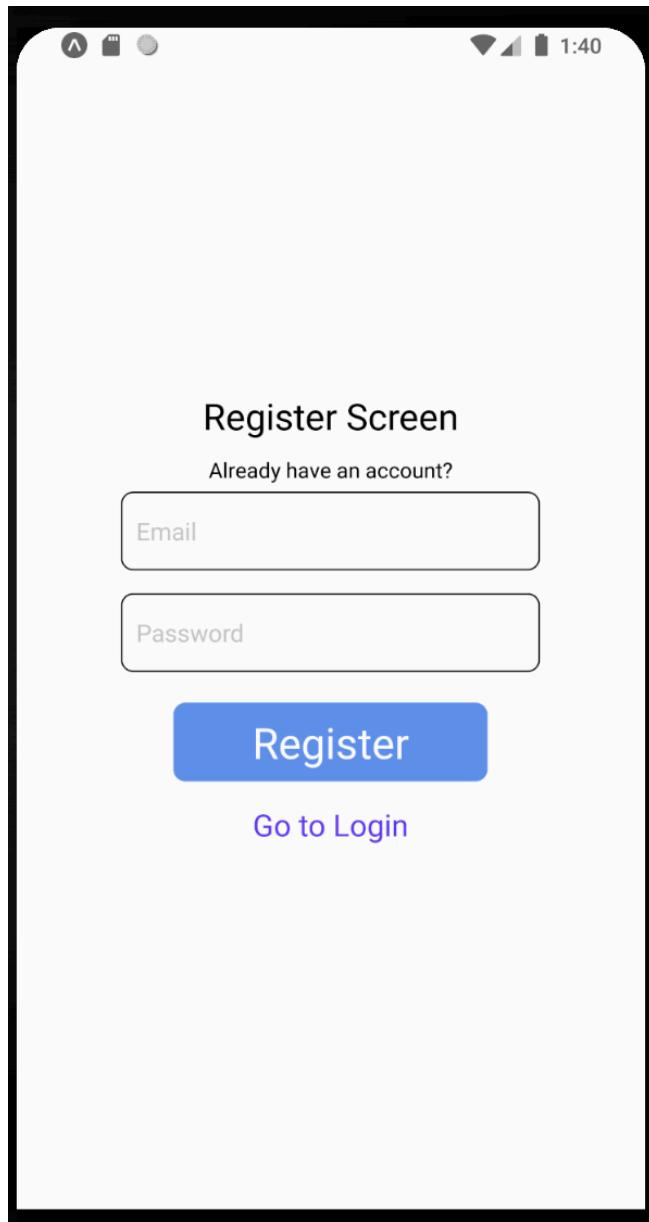
import React from 'react';
import App from './src/Lab2/Bai1/screens/Login';

export default function Project() {
  return <App />;
}

```

Bài tập 2:

- *Mô tả bài tập:*
 - Thiết kế giao diện Register như hình sau:



- *Yêu cầu thực hiện:*
 - Tạo thư mục **screens** chứa file **Register.js**
 - Giao diện sử dụng các StyleSheet và các component cần thiết
 - Cấu hình chạy app sẽ xuất hiện giao diện vừa thiết kế
- *Hướng dẫn thực hiện:*
 - Tạo thư mục **screens** chứa file **Register.js** tương tự Bài 1.
 - Code file **Register.js** tương tự với phần **return** như sau:

```

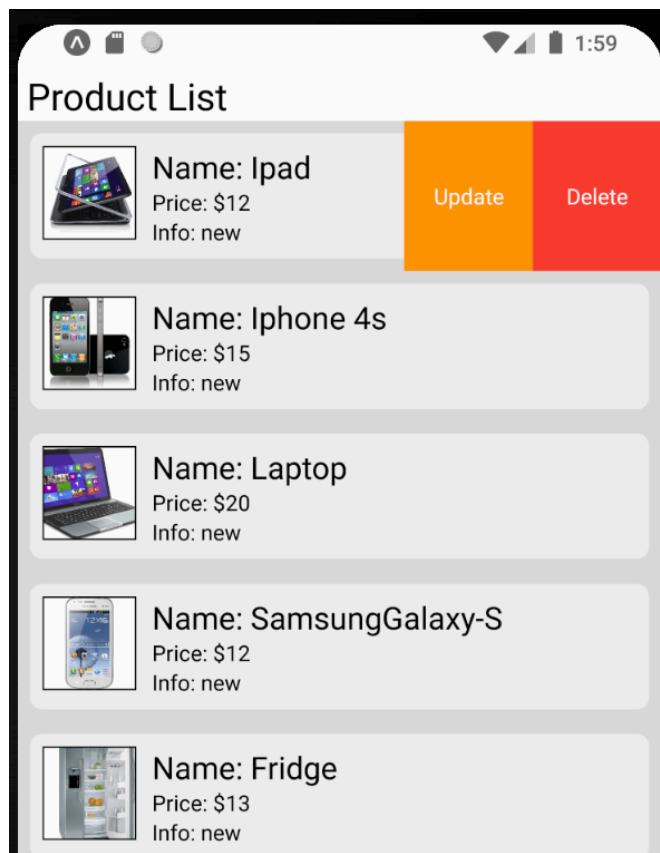
return (
    <View style={styles.container}>
        <Text style={styles.text}> Register Screen </Text>
        {/* Don't have an account */}
        <Text> Already have an account? </Text>
        <TextInput style={styles.input} value={email} placeholder="Email" onChangeText={(text) => setEmail(text)} />
        <TextInput
            style={styles.input}
            value={password}
            placeholder="Password"
            onChangeText={(text) => setPassword(text)}
            secureTextEntry={true}
        />
        <TouchableOpacity style={styles.buttonContainer} onPress={() => Alert.alert('ban vua nhan Register')}>
            <Text style={styles.buttonText}> Register </Text>
        </TouchableOpacity>
        <TouchableOpacity style={styles.navButton} onPress={() => Alert.alert('ban vua nhan Login')}>
            <Text style={styles.navButtonText}> Go to Login </Text>
        </TouchableOpacity>
    </View>
);

```

- Bổ sung style cần thiết.
- Cấu hình chạy app theo hướng dẫn Lab2-Bai1 và xem kết quả.

Bài tập 3:

- *Mô tả bài tập:*
 - Thiết kế giao diện Product List như hình sau:



- *Yêu cầu thực hiện:*
 - Tạo thư mục screens chứa file **Product.js**
 - Giao diện sử dụng các StyleSheet và các component cần thiết
 - Cấu hình chạy app sẽ xuất hiện giao diện vừa thiết kế

- *Hướng dẫn thực hiện:*

- Tạo thư mục **screens** chứa file **Product.js** tương tự Bài 1.
- Tạo thư mục **Images** chứa hình ảnh cần thiết.
- Cài đặt package **swipeout**:

```
npm i --save react-native-swipeout
```

- Code file **Product.js** như sau:

- Import swipeout vào:

```
import Swipeout from 'react-native-swipeout';
```

- Khai báo cứng data:

```
const data = [
  {
    name: 'Ipad',
    image: require('../Images/ipad.jpg'),
    info: 'new',
    price: '$12'
  },
  {
    name: 'Iphone 4s',
    image: require('../Images/iphone4s-1.jpg'),
    info: 'new',
    price: '$15'
  },
  {
    name: 'Laptop',
    image: require('../Images/laptop.jpg'),
    info: 'new',
    price: '$20'
  }
];
```

- Code return giao diện:

```
return (
  <View style={styles.container}>
    <Text style={styles.text}> Product List </Text>
    <FlatList data={data} renderItem={({ item }) => <ListItem item={item} />} />
  </View>
);
```

- Code component **ListItem** đã gọi trong giao diện:

```

//Item FlatList function
const ListItem = (props) => {
  // Cấu hình swipeout
  const swipeoutSettings = {
    autoClose: true,
    onClose: () => {
      console.log('Close swipeout');
    },
    onOpen: () => {
      console.log('Open swipeout');
    },
    right: [
      {
        text: 'Update',
        type: 'secondary',
        onPress: () => {
          console.log('Update');
        }
      },
      {
        text: 'Delete',
        type: 'delete',
        onPress: () => {
          console.log('Delete');
        }
      }
    ]
  ];
  return (
    <Swipeout {...swipeoutSettings}>
      <View style={styles.listContainer}>
        <Image
          source={props.item.image}
          style={{ borderWidth: 1, borderColor: 'black', width: 60, height: 60 }}
        />
        <View>
          <Text style={{ marginLeft: 10, fontSize: 20 }}>Name: {props.item.name}</Text>
          <Text style={{ marginLeft: 10 }}>Price: {props.item.price}</Text>
          <Text style={{ marginLeft: 10 }}>Info: {props.item.info}</Text>
        </View>
      </View>
    </Swipeout>
  );
};

```

- Khai báo style:

```

const { width, height } = Dimensions.get('window');
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center'
  },
  text: {
    fontSize: 24,
    marginTop: 30
  },
  listContainer: {
    backgroundColor: '#f1f1f1',
    flexDirection: 'row',
    margin: width * 3.6 / 187.5,
    padding: width * 3.6 / 187.5,
    borderRadius: width * 3.6 / 187.5
  }
});

```

- Cấu hình chạy app theo hướng dẫn Lab2-Bài1 và xem kết quả.

LAB 3: NAVIGATION - FIREBASE AUTH

1. Chuẩn đầu ra

- Biết điều hướng với Navigation.
- Biết cách kết nối Firebase.
- Sử dụng Firebase Auth.

2. Nội dung các bài tập

Bài tập 1:

- *Mô tả bài tập:*
 - Sử dụng 3 screens ở Lab 2 và thực hiện điều hướng theo logic của app
- *Yêu cầu thực hiện:*
 - Cài đặt package Navigation v5.x
 - Thực hiện điều hướng như sau:
 - Tại màn hình Login:
 - Chuyển sang Register Screen khi nhấn vào New User
 - Chuyển sang Product Screen khi nhấn vào Login (với điều kiện email = 'admin@gmail.com' và password = 'admin')
 - Tại màn hình Register:
 - Chuyển sang Login Screen khi nhấn vào Login
- *Hướng dẫn thực hiện:*
 - Cài đặt Navigation:
npm install @react-navigation/native @react-navigation/stack
expo install react-native-reanimated react-native-gesture-handler react-native-screens react-native-safe-area-context @react-native-community/masked-view
 - Tạo file **App.js**, thư mục **screens** và thư mục **Images** trong Lab3.
 - Chép hình ảnh vào **Images**, 3 screen Lab2 vào thư mục **screens**
 - Code file **App.js** như hình:

```

import * as React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';
import LoginScreen from './screens/Login';
import RegisterScreen from './screens/Register';
import ProductScreen from './screens/Product';

const Stack = createStackNavigator();

const App = () => {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="Login" screenOptions={{ headerShown: false }}>
        <Stack.Screen name="Login" component={LoginScreen} options={{ title: 'Login' }} />
        <Stack.Screen name="Register" component={RegisterScreen} options={{ title: 'Register' }} />
        <Stack.Screen name="Product" component={ProductScreen} options={{ title: 'Product' }} />
      </Stack.Navigator>
    </NavigationContainer>
  );
};

export default App;

```

- Sửa code file **Login.js** như hình:

```

const Login = ({ navigation }) => {
  const [ email, setEmail ] = useState('');
  const [ password, setPassword ] = useState('');
  // thêm function kiểm tra login
  const checkLogin = (email, password) => {
    if (email == 'admin@gmail.com' && password == 'admin') {
      navigation.navigate('Product');
      ToastAndroid.show('Login success!', ToastAndroid.SHORT);
    } else {
      ToastAndroid.show('Wrong email or password!', ToastAndroid.SHORT);
    }
  };
  return (
    <View style={styles.container}>
      <Text style={styles.text}> Login Screen </Text>
      <TextInput style={styles.input} value={email} placeholder="Email" onChangeText={(text) => setEmail(text)} />
      <TextInput
        style={styles.input}
        value={password}
        placeholder="Password"
        onChangeText={(text) => setPassword(text)}
        secureTextEntry={true}
      />
      <TouchableOpacity style={styles.buttonContainer} onPress={() => checkLogin(email, password)}>
        <Text style={styles.buttonText}>Login</Text>
      </TouchableOpacity>
      <TouchableOpacity style={styles.navButton} onPress={() => navigation.navigate('Register')}>
        <Text style={styles.navButtonText}>New user? Join here</Text>
      </TouchableOpacity>
    </View>
  );
};
export default Login;

```

- Sửa code Register.js như hình:

```

const Register = ({ navigation }) => {
  const [ email, setEmail ] = useState('');
  const [ password, setPassword ] = useState('');
  return (
    <View style={styles.container}>
      <Text style={styles.text}> Register Screen </Text>
      {/* Don't have an account */}
      <Text>Already have an account?</Text>
      <TextInput style={styles.input} value={email} placeholder="Email" onChangeText={(text) => setEmail(text)} />
      <TextInput
        style={styles.input}
        value={password}
        placeholder="Password"
        onChangeText={(text) => setPassword(text)}
        secureTextEntry={true}
      />
      <TouchableOpacity style={styles.buttonContainer} onPress={() => Alert.alert('ban vua nhan Register')}>
        <Text style={styles.buttonText}>Register</Text>
      </TouchableOpacity>
      <TouchableOpacity style={styles.navButton} onPress={() => navigation.navigate('Login')}>
        <Text style={styles.navButtonText}>Go to Login</Text>
      </TouchableOpacity>
    </View>
  );
}
export default Register;

```

- Cấu hình chạy app: Open file App.js trong thư mục MOB306-ReactNative (project) và code như hình (Save file và xem kết quả).

```

import React from 'react';
import App from './src/Lab3/App';

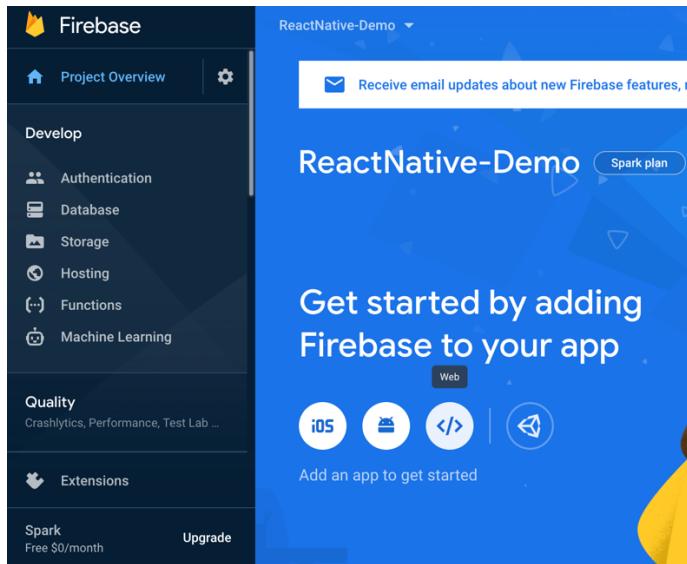
export default function Project() {
  return <App />;
}

```

Bài tập 2:

- *Mô tả bài tập:*
 - Sử dụng Auth Firebase
 - Đăng ký tài khoản với Auth Firebase
- *Yêu cầu thực hiện:*
 - Tạo project trên Firebase.
 - Kết nối Project.
 - Thực hiện code đăng ký tài khoản với Auth Firebase sử dụng email và password.
- *Hướng dẫn thực hiện:*
 - Tạo project trên Firebase:
Vào link: <https://console.firebaseio.google.com/> và thực hiện các bước theo hướng dẫn.

- Kết nối Project: chọn WEB như hình



- Nhập tên App và lưu lại thông tin sau:

```
var firebaseConfig = {
  apiKey: "AIzaSyBNWsGgn6_Uj9GJzQM5H_wai79sm5W5HXg",
  authDomain: "reactnative-demo-eb10d.firebaseio.com",
  databaseURL: "https://reactnative-demo-eb10d.firebaseio.com",
  projectId: "reactnative-demo-eb10d",
  storageBucket: "reactnative-demo-eb10d.appspot.com",
  messagingSenderId: "434748492970",
  appId: "1:434748492970:web:4fef845014b41b644a14d3",
  measurementId: "G-F78KN1BVWH"
};
```

- Bật Auth Firebase sử dụng email và password:
 - Chọn Authentication, và chọn tab Sign-in method
 - Enable Email/Password
- Kết nối Project ReactNative với Firebase:
 - Tạo folder firebase chứa file firebase.js.
 - Code file firebase.js như hình: (copy thông tin lưu trữ bên trên)

```
import * as firebase from 'firebase'; // npm i --save firebase
var Config = {
  apiKey: 'AIzaSyBNWsGgn6_Uj9GJzQM5H_wai79sm5W5HXg',
  authDomain: 'reactnative-demo-eb10d.firebaseio.com',
  databaseURL: 'https://reactnative-demo-eb10d.firebaseio.com',
  projectId: 'reactnative-demo-eb10d',
  storageBucket: 'reactnative-demo-eb10d.appspot.com',
  messagingSenderId: '434748492970',
  appId: '1:434748492970:web:4fef845014b41b644a14d3',
  measurementId: 'G-F78KN1BVWH'
};
// Initialize Firebase
export default (firebaseConfig = firebase.initializeApp(Config));
```

- Code đăng ký tài khoản với Auth Firebase sử dụng email và password (Sửa file Register.js):
 - Import firebase config:

```
import firebaseConfig from '../firebase/firebase';
```

- Code function _register (ngoài funtion Register):

```
const _register = async (email, password) => {
  await firebaseConfig
    .auth()
    .createUserWithEmailAndPassword(email, password)
    .then((user) => {
      console.log('Register success!');
      ToastAndroid.show('Register success!', ToastAndroid.SHORT);
    })
    .catch((error) => {
      const { code, message } = error;
      console.log('Error: ' + message);
      ToastAndroid.show('Register fail!', ToastAndroid.SHORT);
    });
};
```

- Gọi function register:

```
<TouchableOpacity style={styles.buttonContainer} onPress={() => _register(email, password)}>
  <Text style={styles.buttonText}>Register</Text>
</TouchableOpacity>
```

Bài tập 3:

- *Mô tả bài tập:*
 - Đăng nhập tài khoản với Auth Firebase
- *Yêu cầu thực hiện:*
 - Thực hiện code đăng nhập tài khoản với Auth Firebase sử dụng email và password.
- *Hướng dẫn thực hiện:*
 - Code đăng nhập tài khoản với Auth Firebase sử dụng email và password (Sửa file Login.js):
 - Import firebase config tương tự file Register.js
 - Code function _login (ngoài funtion Login):

```
const _login = async (email, password) => {
  await firebaseConfig
    .auth()
    .signInWithEmailAndPassword(email, password)
    .then((user) => {
      console.log('Login success!');
      return true;
    })
    .catch((error) => {
      const { code, message } = error;
      console.log('Error: ' + message);
      return false;
    });
};
```

- Sửa code funtion checklogin: kiểm tra true false của funtion _login.

LAB 4: FIREBASE DATABASE

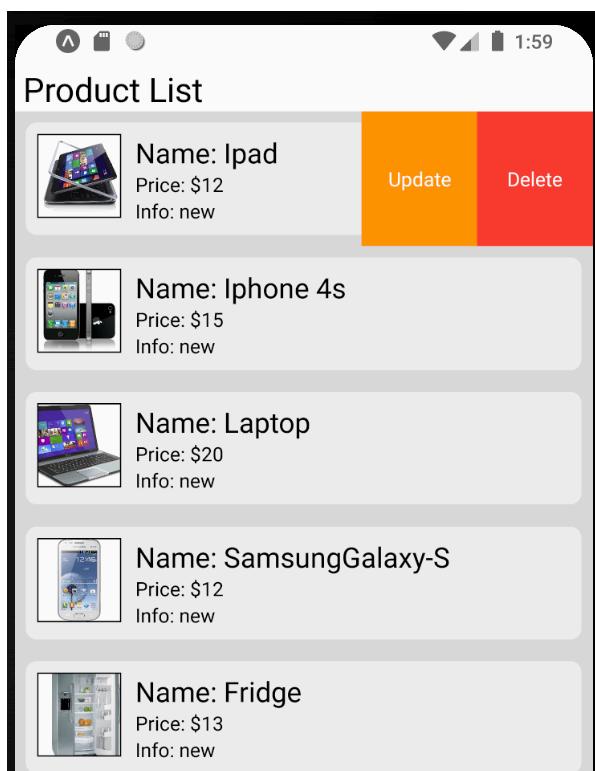
1. Chuẩn đầu ra

- Sử dụng firebase database Realtime
- Sử dụng firebase storage.
- Component cơ bản: Modal.

2. Nội dung các bài tập

Bài tập 1:

- *Mô tả bài tập:*
 - Sử dụng firebase Realtime Database.
 - Show product trong database lên Flatlist
 - Xoá product trong database
- *Yêu cầu thực hiện:*
 - Enable Realtime Database.
 - Show product trong database lên Flatlist bao gồm hình ảnh và có dùng swipeout.
 - Xoá product trong database dùng button trong swipeout.



- *Hướng dẫn thực hiện:*

- Enable Realtime Database:
 - Vào link: <https://console.firebaseio.google.com/> và thực hiện chọn Database
 - Chọn Create Realtime Database, chọn Test mode
- Show product:
 - Import data mẫu vào để demo (file data.json trong tài nguyên)

- Sửa code file Product.js như hình:

```

import React, { useState, useEffect } from 'react';
import { View, StyleSheet, FlatList, Dimensions, Image, Text, Alert } from 'react-native';
import Swipeout from 'react-native-swipeout'; //npm i --save react-native-swipeout
import firebaseConfig from '../firebase.firebaseio.js';
const Product = () => {
    const [ data, setData ] = useState([]);
    // Function getAll Products
    const getAllProduct = () => {
        firebaseConfig.database().ref().child('products').on('value', (snap) => {
            var items = [];
            snap.forEach((child) => {
                let item = {
                    key: child.key,
                    name: child.val().name,
                    price: child.val().price,
                    info: child.val().info,
                    image: child.val().image
                };
                items.push(item);
            });
            setData(items);
        });
    };
    useEffect(() => {
        getAllProduct();
    }, []);
    return (
        <View style={styles.container}>
            <Text style={styles.text}> Product List </Text>
            <FlatList data={data} renderItem={({ item }) => <ListItem item={item} />} />
        </View>
    );
}
export default Product;

```

- o Xoá product:

- Tạo folder DAO chứa file ProductDAO.js và code như hình:

```

import { ToastAndroid } from 'react-native';
import firebaseConfig from '../firebase.firebaseio.js';
module.exports.delete = (key) => {
    firebaseConfig
        .database()
        .ref()
        .child('products')
        .child(key)
        .remove()
        .then(() => {
            console.log('Delete success!');
            ToastAndroid.show('Delete success!', ToastAndroid.SHORT);
        })
        .catch((error) => {
            console.log('Delete fail! ' + error);
            ToastAndroid.show('Delete fail!', ToastAndroid.SHORT);
        });
}

```

- Sửa code function ListItem file Product.js:

- Import ProductDAO.js:

```
import ProductDAO from '../DAO/ProductDAO';
```

- Sửa code function ListItem:

```
const ListItem = (props) => {
    // Cấu hình swipeout
    const swipeoutSettings = {
        autoClose: true,
        onClose: () => {
            console.log('Close swipeout');
        },
        onOpen: () => {
            console.log('Open swipeout');
        },
        right: [
            {
                text: 'Update',
                type: 'secondary',
                onPress: () => {
                    console.log('Update');
                }
            },
            {
                text: 'Delete',
                type: 'delete',
                onPress: () => {
                    Alert.alert(
                        'Delete',
                        'Are you want to delete product ' + props.item.name + '?',
                        [
                            { text: 'No', onPress: () => console.log('Cancel Delete'), type: 'cancel' },
                            { text: 'Yes', onPress: () => ProductDAO.delete(props.item.key) }
                        ],
                        { cancelable: true }
                    );
                }
            }
        ]
    };
    return (
        <Swipeout {...swipeoutSettings}>
            <View style={styles.listContainer}>
                <Image
                    source={{ uri: props.item.image, width: 60, height: 60 }}
                    style={{ borderWidth: 1, borderColor: 'black' }}
                />
                <View>
                    <Text style={{ marginLeft: 10, fontSize: 20 }}>Name: {props.item.name}</Text>
                    <Text style={{ marginLeft: 10 }}>Price: {props.item.price}</Text>
                    <Text style={{ marginLeft: 10 }}>Info: {props.item.info}</Text>
                </View>
            </View>
        </Swipeout>
    );
};
```

Bài tập 2:

- *Mô tả bài tập:*
 - Sử dụng firebase storage.
 - Insert product vào database
- *Yêu cầu thực hiện:*
 - Enable firebase storage.
 - Insert product vào database và hình ảnh sản phẩm vào storage.
- *Hướng dẫn thực hiện:*
 - Enable firebase storage.
 - Vào link: <https://console.firebaseio.google.com/> và thực hiện chọn Storage
 - Chọn Get start, Next.
 - Insert product vào database và hình ảnh sản phẩm vào storage:
 - Install expo-image-picker vào project:
npm i -save expo-image-picker
 - Sửa code file ProductDAO.js như sau:

- Thêm function upimage:

```
//function up ảnh fireStorage
const _uploadImage = async (name, uri) => {
    const path = 'images/' + name + '.jpg';
    return new Promise(async (res, rej) => {
        const response = await fetch(uri);
        const file = await response.blob();

        let upload = firebaseConfig.storage().ref(path).put(file);

        upload.on(
            'state_changed',
            (snapshot) => {},
            (err) => {
                rej(err);
            },
            async () => {
                const url = await upload.snapshot.ref.getDownloadURL();
                res(url);
            }
        );
    });
};
```

- Thêm function insert:

```
module.exports.insert = async (name, price, info, image) => {
  const remoteUri = await _uploadImage(name, image);
  firebaseConfig
    .database()
    .ref()
    .child('products')
    .push({
      name: name,
      price: price,
      info: info,
      image: remoteUri
    })
    .then(() => {
      console.log('Insert success!');
      ToastAndroid.show('Insert success!', ToastAndroid.SHORT);
    })
    .catch((error) => {
      console.log('Insert fail! ' + error);
      ToastAndroid.show('Insert fail!', ToastAndroid.SHORT);
    });
};
```

- Sửa code file Product.js như sau:

- Import ImagePicker:

```
import * as ImagePicker from 'expo-image-picker';
```

- Sửa function Product(), khai báo State, function local và thêm Modal trong return như sau:

Khai báo State, function local

```
const [ modalVisible, setModalVisible ] = useState(false);
_hideDialog = () => {
  setModalVisible(false);
};
_showDialog = () => {
  setModalVisible(true);
};
```

Thêm Modal trong return

```
return (
  <View style={styles.container}>
    <Text style={styles.text}> Product List </Text>
    <FlatList
      data={data}
      renderItem={({ item }) => <ListItem item={item} _showDialog={_showDialog} _setCurrent={_setCurrent} />}
    />

    {/* Tao nut them Product */}
    <TouchableOpacity
      style={styles.fab}
      onPress={() => {
        _showDialog();
      }}
    >
      <Text style={styles.text}>+</Text>
    </TouchableOpacity>

    {/* Neu co item duoc chon se mo Update */}
    <Modal animationType="slide" transparent={true} visible={modalVisible}>
      {currentItem ? (
        <ProductUpdate item={currentItem} _hideDialog={_hideDialog} />
      ) : (
        <ProductInsert _hideDialog={_hideDialog} />
      )}
    </Modal>
  </View>
);
```

- Thêm function ProductInsert ngoài function Product():

```

//Insert Product function
const ProductInsert = (props) => {
  const [ name, setName ] = useState('');
  const [ price, setPrice ] = useState('');
  const [ info, setInfo ] = useState('');
  const [ image, setImage ] = useState('https://reactjs.org/logo-og.png');

  //function chọn ảnh
  const _chooseImage = async () => {
    const result = await ImagePicker.launchImageLibraryAsync({
      mediaTypes: ImagePicker.MediaTypeOptions.Images,
      allowsEditing: true,
      aspect: [ 4, 3 ]
    });

    if (!result.cancelled) {
      console.log(image);
      setImage(result.uri);
    }
  };
  return (
    <View style={styles.centeredView}>
      <View style={styles.modalView}>
        <Text style={styles.modalText}>Insert Product</Text>
        <TouchableWithoutFeedback onPress={() => _chooseImage()}>
          <Image
            source={{ uri: image, width: 150, height: 150 }}
            style={{ borderWidth: 1, borderColor: 'black' }}
          />
        </TouchableWithoutFeedback>
        <View style={styles.lineDialog}>
          <Text style={styles.textDialog}>Name: </Text>
          <TextInput style={styles.TextInputDialog} value={name} onChangeText={(text) => setName(text)} />
        </View>

        <View style={styles.lineDialog}>
          <Text style={styles.textDialog}>Price: </Text>
          <TextInput style={styles.TextInputDialog} value={price} onChangeText={(text) => setPrice(text)} />
        </View>

        <View style={styles.lineDialog}>
          <Text style={styles.textDialog}>Info: </Text>
          <TextInput style={styles.TextInputDialog} value={info} onChangeText={(text) => setInfo(text)} />
        </View>

        <View style={styles.modelButton}>
          <TouchableOpacity
            style={{ ...styles.openButton, backgroundColor: '#2196F3' }}
            onPress={() => {
              props._hideDialog();
            }}
          >
            <Text style={styles.textStyle}>Cancel</Text>
          </TouchableOpacity>
          <TouchableOpacity
            style={{ ...styles.openButton, backgroundColor: '#2196F3' }}
            onPress={() => {
              props._hideDialog();
              ProductDAO.insert(name, price, info, image);
            }}
          >
            <Text style={styles.textStyle}>Insert</Text>
          </TouchableOpacity>
        </View>
      </View>
    </View>
  );
};

```

Bài tập 3:

- *Mô tả bài tập:*
 - Update product trong database
- *Yêu cầu thực hiện:*
 - Update product trong database và hình ảnh sản phẩm trong storage.
- *Hướng dẫn thực hiện:*
 - Thêm function update trong file ProductDAO.js

```
module.exports.update = async (key, name, price, info, image) => {
  const remoteUri = await _uploadImage(name, image);
  firebaseConfig
    .database()
    .ref()
    .child('products')
    .child(key)
    .set({
      name: name,
      price: price,
      info: info,
      image: remoteUri
    })
    .then(() => {
      console.log('Update success!');
      ToastAndroid.show('Update success!', ToastAndroid.SHORT);
    })
    .catch((error) => {
      console.log('Update fail! ' + error);
      ToastAndroid.show('Update success!', ToastAndroid.SHORT);
    });
};
```

- Sửa code file Product.js
 - Khai báo State và function local:

```
const [ currentItem, setCurrentItem ] = useState(null);
_setCurrent = async (item) => {
  await setCurrentItem(item);
};
```

- Thêm function ProductUpdate ngoài function Product() tương tự ProductInsert nhưng cần có giá trị item chọn:

```
// Update Product function
const ProductUpdate = (props) => {
  const [ key, setKey ] = useState(props.item.key);
  const [ name, setName ] = useState(props.item.name);
  const [ price, setPrice ] = useState(props.item.price);
  const [ info, setInfo ] = useState(props.item.info);
  const [ image, setImage ] = useState(props.item.image);

  //function chọn ảnh
  const _chooseImage = async () => {
```

- Sửa function ListItem để setItemCurrent và mở Modal khi chọn update

```
//Item FlatList function
const ListItem = (props) => {
  // Cấu hình swipeout
  const swipeoutSettings = {
    autoClose: true,
    onClose: () => {
      props._setCurrent(null);
    },
    onOpen: () => {
      props._setCurrent(props.item);
      console.log('Open swipeout');
    },
    right: [
      {
        text: 'Update',
        type: 'secondary',
        onPress: () => {
          props._showDialog();
        }
      },
    ],
  };
}
```

3. Tổ chức thực hành và cách thức đánh giá (soạn theo gợi ý bên dưới)

- SV thực hành theo nhóm 4-6 SV/ nhóm.
- GV hướng dẫn, theo dõi, đánh giá trong quá trình SV thực hành và đánh giá sản phẩm cuối cùng.
- Tuỳ theo bài Lab, GV có thể chấm tại lớp hay chấm ngoài giờ với sản phẩm SV nộp cuối buổi/hoàn tất bài Lab.

4. Thang điểm đánh giá (10)

STT	Hạng mục đánh giá	Thang điểm	Ghi chú
	Lab 1	...	
1.1	Bài tập 1	
1.2	Bài tập 2	
....	
	Lab 2	
2.1	Bài tập 1	
2.2	Bài tập 2	
....	
	Lab	
....	Bài tập 1	
....	Bài tập 2	
...	
	Thái độ - chuyên cần	1.0	Thái độ làm bài, tích cực hỏi GV, thảo luận nhóm