**BẢNG TỔNG HỢP CHỨC NĂNG ĐÃ THỰC HIỆN**

**DỰ ÁN : ỨNG DỤNG THÊM SỬA XÓA SẢN PHẨM THÍCH HỢP ĐA NỀN TẢNG**

Họ tên : Lê Công Giang

Mã Sinh viên : 21IT542

Link mã nguồn : git@github.com:Giang-203/LabGiuaKiDaNenTang.git

I, Khởi tạo dự án : ( Expo )

1. Nếu bạn chưa cài đặt Expo CLI, chạy lệnh này để cài đặt:

npm install -g expo-cli

1. Tạo một dự án Expo mới:

expo init my-new-project

1. Điều hướng đến thư mục dự án của bạn:

cd my-new-project

1. Khởi chạy dự án bằng cách chạy lệnh:

expo start

Dự án sẽ khởi động và bạn có thể mở nó trên trình giả lập hoặc bằng ứng dụng Expo Go trên điện thoại của mình.

II, Kết nối Firebase / Firestore

Để kết nối Firebase Firestore với dự án React Native sử dụng Expo, bạn có thể làm theo các bước sau:

**Bước 1: Cài đặt Firebase SDK**

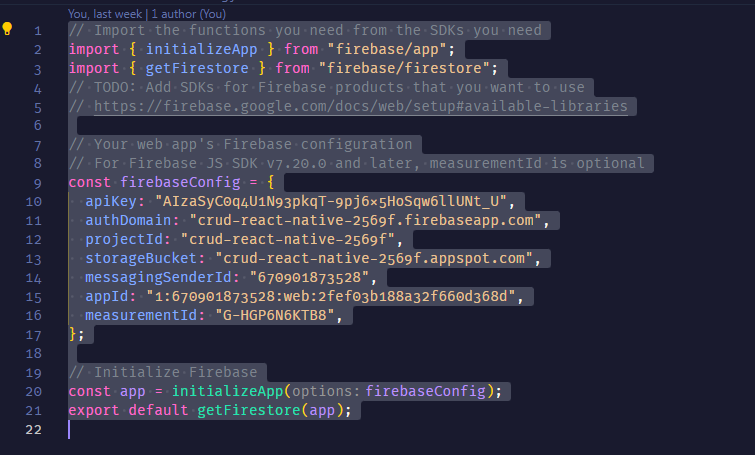
1. **Cài đặt Firebase trong dự án của bạn**: Trong thư mục gốc của dự án, chạy lệnh sau để cài đặt Firebase SDK:

npm install firebase

1. **Cấu hình Firebase**: Sau khi cài đặt xong, bạn cần cấu hình Firebase trong dự án bằng cách thêm các file cấu hình từ Firebase Console. Làm theo các bước:
   * Truy cập Firebase Console.
   * Tạo một dự án Firebase mới (hoặc chọn một dự án có sẵn).
   * Trong phần "Project settings" > "General", bạn thêm ứng dụng web (vì dự án Expo không cần native code ban đầu).
   * Lấy các thông tin cấu hình Firebase (apiKey, authDomain, projectId, v.v.) từ Firebase Console.

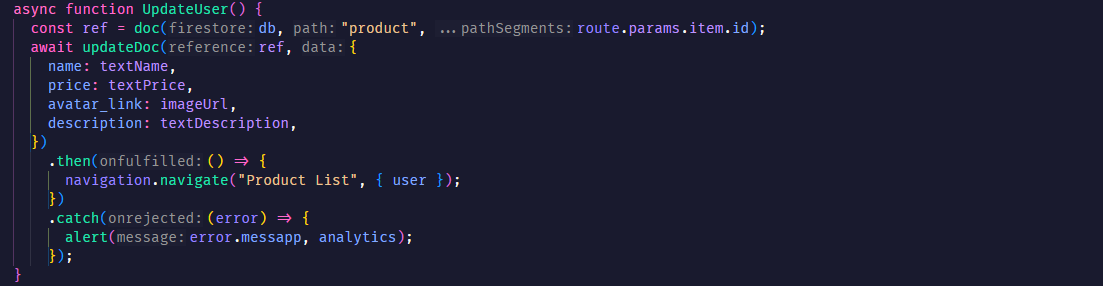
**Bước 2: Cấu hình Firebase trong dự án**

Tạo một file riêng để cấu hình Firebase (ví dụ: firebaseConfig.js) và đặt mã như sau:



**Bước 3: Sử dụng Firestore trong dự án**

Bây giờ, bạn có thể sử dụng Firestore trong các component React Native của mình. Ví dụ, đây là cách lấy dữ liệu từ một collection:

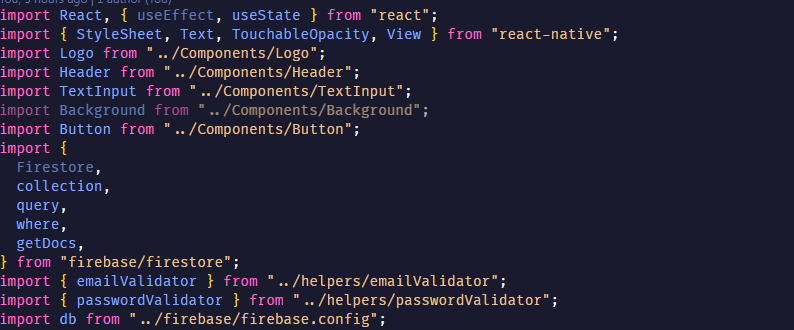


**Bước 4: Kiểm tra cấu hình Firebase**

Kiểm tra lại rằng dự án Firebase đã được kết nối thành công bằng cách thêm dữ liệu vào Firestore và kiểm tra xem nó có xuất hiện trong ứng dụng của bạn không.

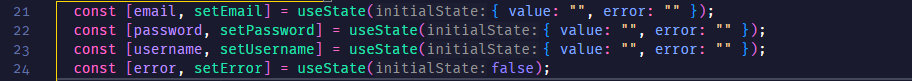
III, Xây dựng Tính năng Đăng nhập ( Login )

1. **Import các dependencies cần thiết**

****

* **React**: Dùng để xây dựng các component trong ứng dụng.
* **React Native Components**: Text, TouchableOpacity, View dùng để tạo giao diện.
* **Custom Components**: Bạn đang sử dụng các component tuỳ chỉnh từ thư mục Components, bao gồm: Logo, Header, TextInput, Background, và Button.
* **Firebase Firestore**: Bạn import collection, query, where, và getDocs từ Firestore để thực hiện truy vấn dữ liệu từ Firebase Firestore.
* **Validation**: Bạn có thêm các helper emailValidator và passwordValidator để kiểm tra tính hợp lệ của dữ liệu đầu vào.
* **Database Configuration**: Kết nối tới Firestore thông qua db được import từ firebase.config.

1. **Xây dựng các state cần thiết**

****

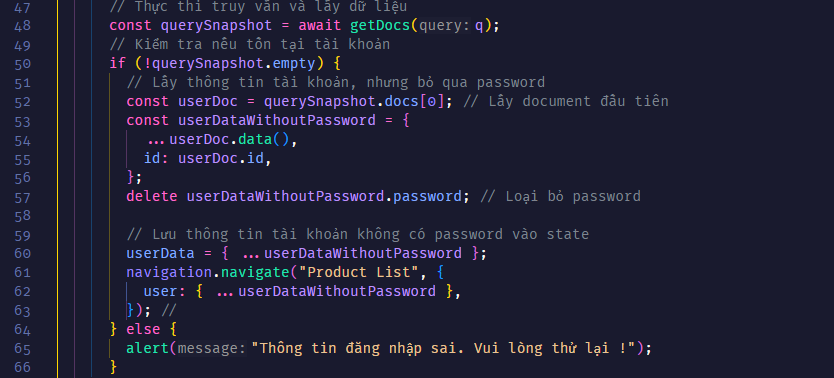
* **email, password, username**: Đây là ba state tương ứng với các trường dữ liệu của form đăng nhập, mỗi state có value để lưu giá trị và error để hiển thị lỗi nếu có.
* **error**: Một state boolean để kiểm soát việc hiển thị thông báo lỗi chung nếu dữ liệu nhập vào không hợp lệ.

1. **Xử lý khi người dùng nhấn nút đăng nhập**

****

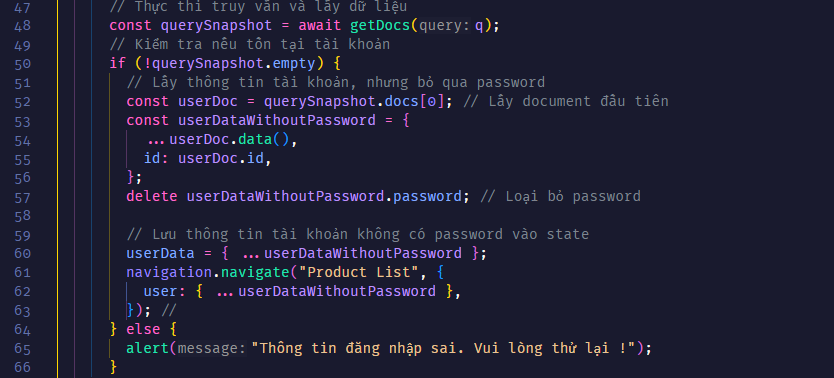
* **Validation**: Trước tiên, bạn kiểm tra tính hợp lệ của email, password, và username bằng cách sử dụng các hàm emailValidator và passwordValidator.
* Nếu có lỗi (email, password hoặc username không hợp lệ), các state tương ứng sẽ được cập nhật với thông báo lỗi, và hàm kết thúc ở đây.

1. **Truy vấn Firestore để kiểm tra thông tin đăng nhập**

****

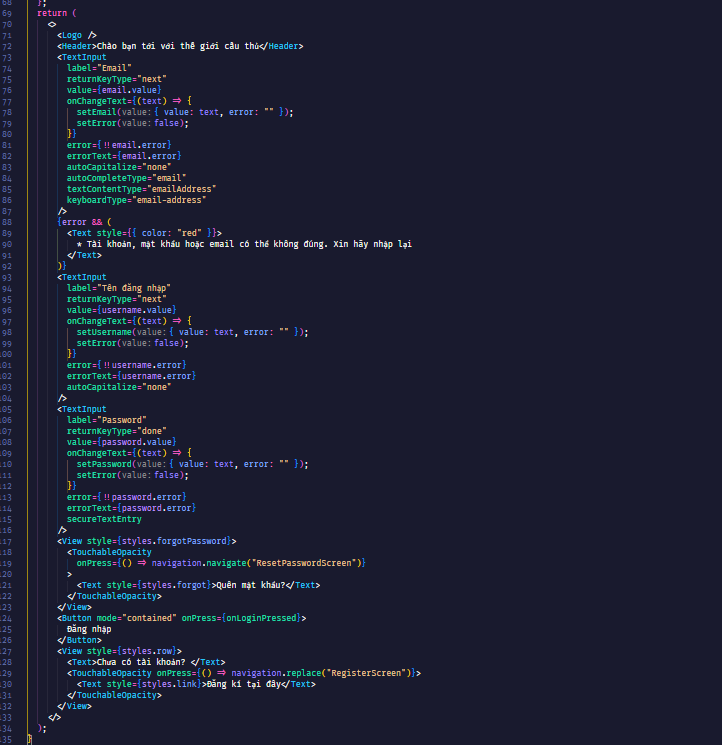
* Nếu dữ liệu hợp lệ, bạn tạo một truy vấn Firebase Firestore để tìm kiếm người dùng với các thông tin nhập vào: email, username và password.
* Truy vấn này sẽ tìm kiếm trong collection users trong Firestore.

1. **Xử lý kết quả truy vấn**

****

* Nếu kết quả truy vấn không rỗng (có người dùng khớp với thông tin nhập vào), bạn lấy document đầu tiên, xóa bỏ trường password để bảo mật, và lưu thông tin người dùng vào userData.
* Sau đó, bạn điều hướng người dùng đến màn hình "Product List" và truyền thông tin người dùng đã đăng nhập thành công.
* Nếu không có người dùng nào khớp, một thông báo lỗi sẽ xuất hiện để thông báo thông tin đăng nhập sai.

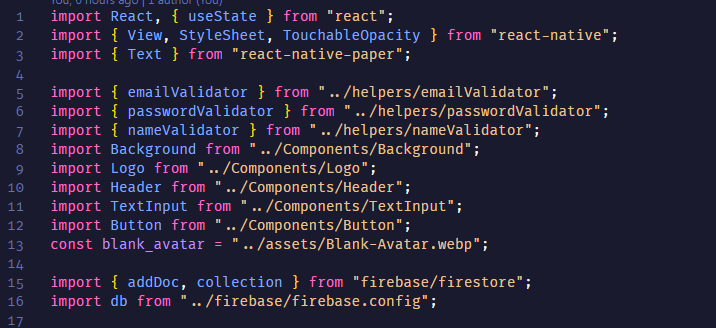
1. **Giao diện màn hình đăng nhập**

****

* **Logo và Header**: Hiển thị logo và tiêu đề.
* **TextInput**: Các trường nhập liệu bao gồm email, username, và password. Mỗi trường đều được liên kết với state tương ứng và hiển thị lỗi nếu có.
* **Button**: Nút đăng nhập được kích hoạt bằng cách gọi hàm onLoginPressed.
* **Điều hướng**: Người dùng có thể chọn quên mật khẩu để được điều hướng đến màn hình Reset Password, hoặc nếu chưa có tài khoản, có thể chọn đăng kí.

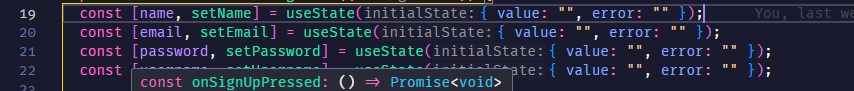
IV, Xây dựng tính năng Đăng kí ( Register )

**1. Import các dependencies cần thiết**

****

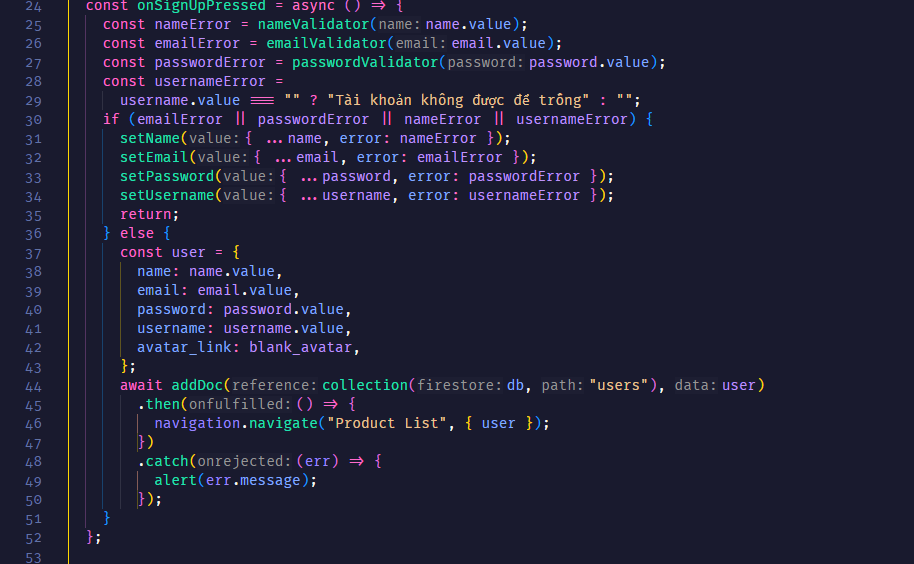
* **React & React Native**: Import các thư viện cần thiết để tạo giao diện và xử lý logic.
* **Firebase Firestore**: Bạn import addDoc và collection từ Firestore để lưu dữ liệu người dùng mới vào collection users.
* **Validation**: Import các hàm kiểm tra tính hợp lệ của các trường thông tin như email, password, và tên từ các file helper.
* **Custom Components**: Sử dụng các component tuỳ chỉnh như Background, Logo, Header, TextInput, và Button để xây dựng giao diện đăng ký.
* **blank\_avatar**: Đường dẫn đến hình ảnh mặc định cho avatar.

**2. Khai báo các state cần thiết**

****

* **name, email, password, username**: Đây là các state dùng để lưu giá trị và trạng thái lỗi của các trường thông tin khi người dùng nhập dữ liệu.

**3. Xử lý khi người dùng nhấn nút đăng ký**

****

* **Validation**: Khi người dùng nhấn đăng ký, hàm onSignUpPressed sẽ kiểm tra tính hợp lệ của từng trường thông tin (name, email, password, và username). Nếu có lỗi, thông báo lỗi sẽ được lưu vào các state tương ứng và hiển thị lên giao diện.
* **Tạo tài khoản**: Nếu tất cả dữ liệu hợp lệ, một đối tượng user sẽ được tạo với các trường dữ liệu, bao gồm cả avatar mặc định.
* **Lưu vào Firestore**: Dữ liệu user sẽ được lưu vào Firestore thông qua hàm addDoc và sau khi thành công, ứng dụng sẽ điều hướng người dùng đến màn hình "Product List".

**4. Giao diện màn hình đăng ký**

return (

    <Background>

      <Logo />

      <Header>Đăng kí thành viên</Header>

      <TextInput

        label="Tên hiển thị"

        returnKeyType="next"

        value={name.value}

        onChangeText={(text) => setName({ value: text, error: "" })}

        error={!!name.error}

        errorText={name.error}

      />

      <TextInput

        label="Email"

        returnKeyType="next"

        value={email.value}

        onChangeText={(text) => setEmail({ value: text, error: "" })}

        error={!!email.error}

        errorText={email.error}

        autoCapitalize="none"

        autoCompleteType="email"

        textContentType="emailAddress"

        keyboardType="email-address"

      />

      <TextInput

        label="Tên đăng nhập"

        returnKeyType="next"

        value={username.value}

        onChangeText={(text) => setUsername({ value: text, error: "" })}

        autoCapitalize="none"

        autoCompleteType="email"

        textContentType="emailAddress"

        keyboardType="email-address"

      />

      <TextInput

        label="Password"

        returnKeyType="done"

        value={password.value}

        onChangeText={(text) => setPassword({ value: text, error: "" })}

        error={!!password.error}

        errorText={password.error}

        secureTextEntry

      />

      <Button

        mode="contained"

        onPress={onSignUpPressed}

        style={{ marginTop: 24 }}

      >

        Đăng kí

      </Button>

      <View style={styles.row}>

        <Text>Đã có tài khoản? </Text>

        <TouchableOpacity onPress={() => navigation.replace("LoginScreen")}>

          <Text style={styles.link}>Đăng nhập tại đây</Text>

        </TouchableOpacity>

      </View>

    </Background>

  );

}

* **Logo và Header**: Hiển thị logo và tiêu đề "Đăng kí thành viên".
* **TextInput**: Các trường nhập liệu bao gồm tên, email, tên đăng nhập, và mật khẩu. Mỗi trường đều có state tương ứng và hiển thị lỗi nếu có.
* **Button**: Nút đăng ký kích hoạt hàm onSignUpPressed.
* **Điều hướng**: Người dùng có thể chọn "Đăng nhập tại đây" nếu đã có tài khoản, và ứng dụng sẽ chuyển họ đến màn hình đăng nhập.

V, Xây dựng trang chủ Product List

**1. Import các dependencies cần thiết**

****

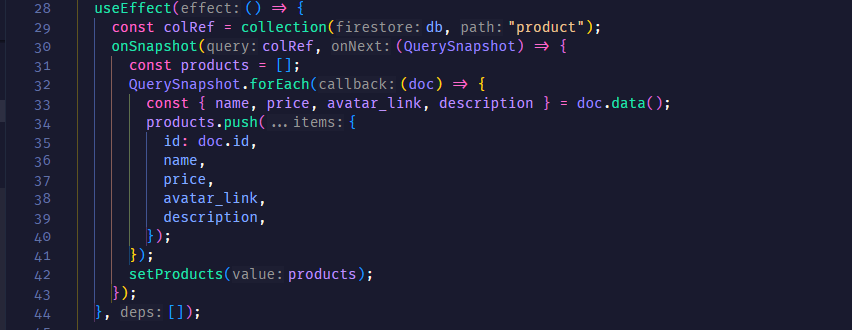
* **Firebase Firestore**: Bạn sử dụng onSnapshot để lắng nghe sự thay đổi dữ liệu trực tiếp từ Firestore.
* **React Native**: Sử dụng các thành phần như FlatList, View, Image, Pressable để hiển thị danh sách sản phẩm và điều hướng giữa các màn hình.
* **Helper functions**: formatPrice để định dạng giá sản phẩm hiển thị đúng kiểu VNĐ.
* **Assets**: Bạn dùng hình ảnh mặc định emptyList khi không có sản phẩm nào trong danh sách.

**2. Khai báo state**

****

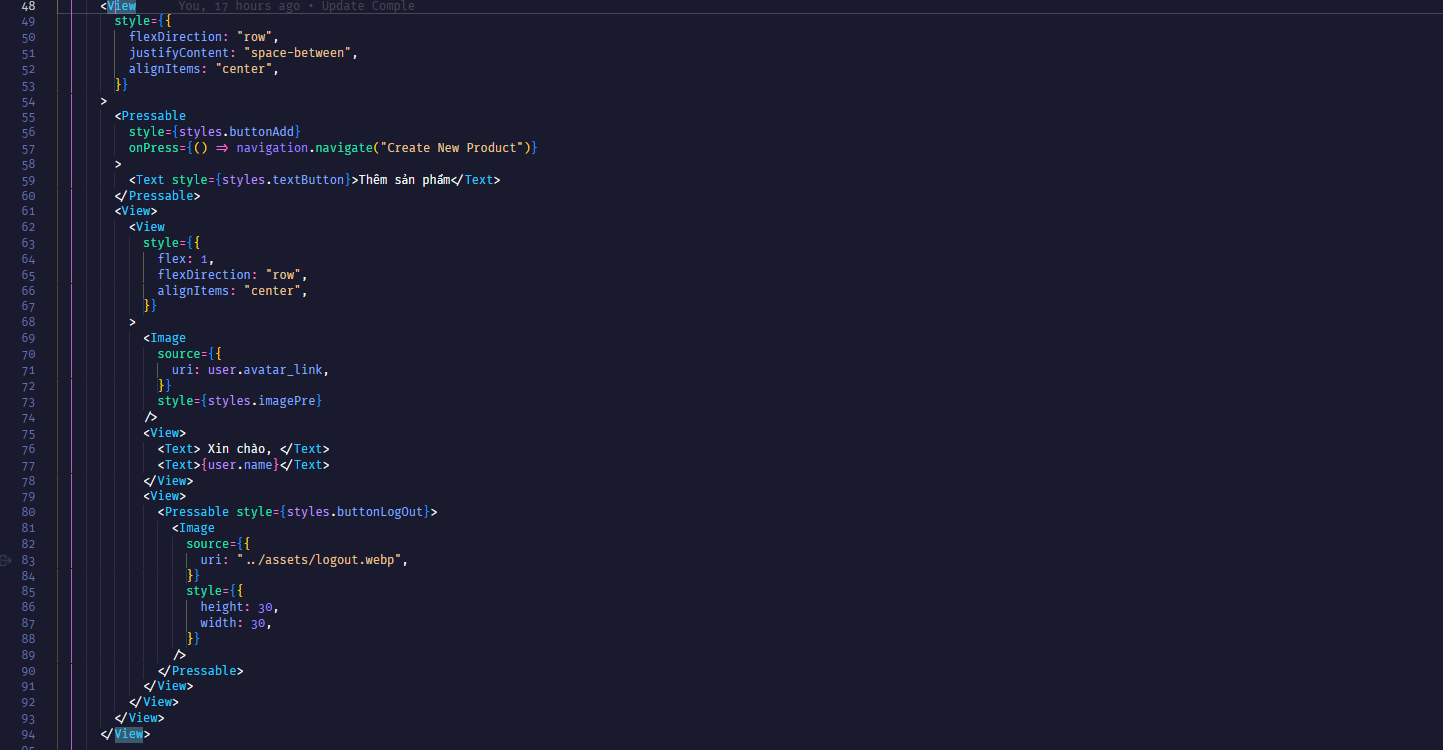
* **products**: useState để lưu trữ danh sách sản phẩm lấy từ Firestore.
* **isMobile**: Dùng để xác định nếu thiết bị đang chạy là Android hoặc iOS.
* **user**: Thông tin người dùng được truyền từ màn hình trước đó thông qua route.params.

**3. Lấy dữ liệu sản phẩm từ Firestore**

****

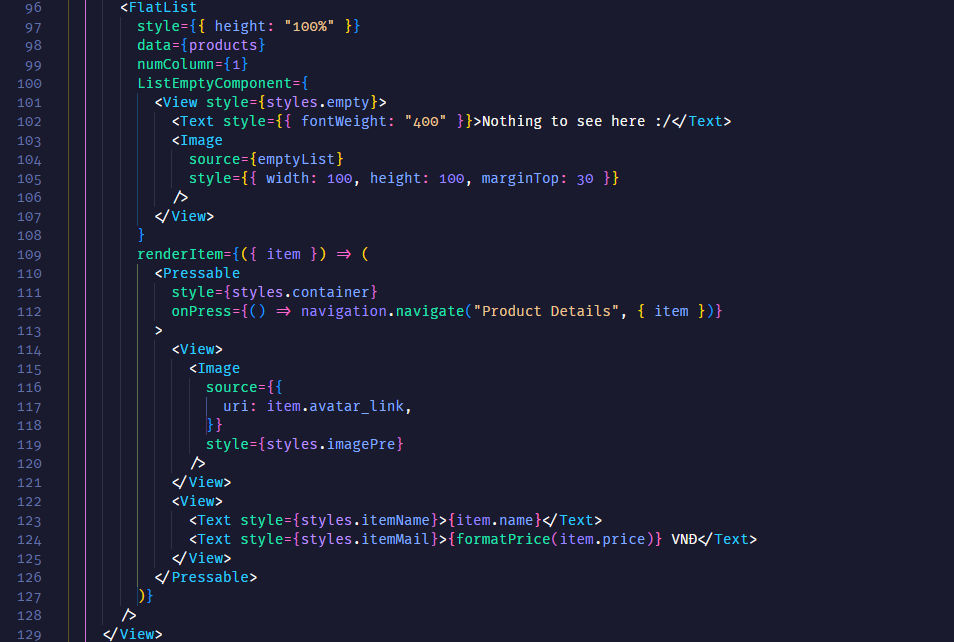
* **useEffect**: Khi component được render, useEffect sẽ được kích hoạt để lắng nghe các thay đổi từ Firestore trong collection product.
* **onSnapshot**: Lấy dữ liệu từ Firestore và theo dõi các thay đổi. Mỗi khi có thay đổi, dữ liệu mới được cập nhật và setProducts được gọi để cập nhật danh sách sản phẩm.

**4. Giao diện đầu trang**

****

* **Thêm sản phẩm**: Nút Pressable cho phép người dùng điều hướng đến màn hình "Create New Product" để thêm sản phẩm mới.
* **Thông tin người dùng**: Hiển thị avatar và tên người dùng từ thông tin được truyền vào route.params.
* **Log out**: Nút log out, mặc dù hiện chưa có hành động xử lý đi kèm.

**5. Danh sách sản phẩm (FlatList)**

****

* **FlatList**: Hiển thị danh sách sản phẩm. Nếu không có sản phẩm nào, sẽ hiển thị component ListEmptyComponent với hình ảnh và thông báo "Nothing to see here :/".
* **renderItem**: Mỗi sản phẩm sẽ hiển thị tên, giá và avatar. Khi người dùng nhấn vào một sản phẩm, họ sẽ được chuyển đến màn hình chi tiết sản phẩm (Product Details).

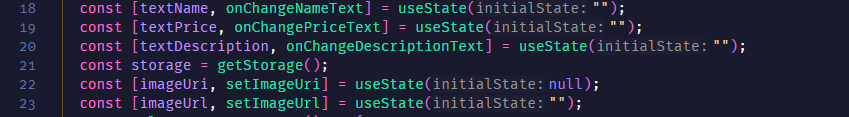
VI, Xây dựng tính năng them sản phẩm

**1. Imports các dependencies cần thiết**



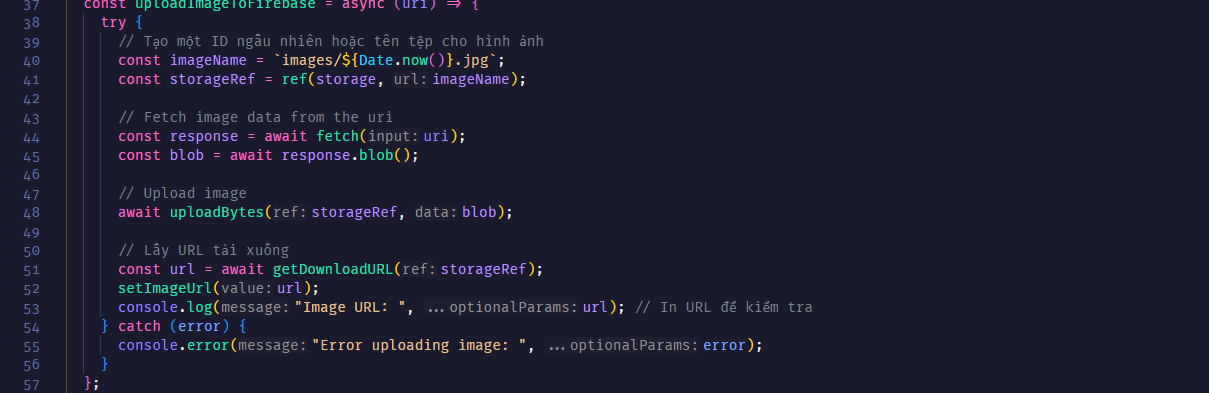
* **Firebase Firestore**: Dùng addDoc để thêm dữ liệu sản phẩm mới vào collection product trong Firestore.
* **Firebase Storage**: uploadBytes, getDownloadURL để tải hình ảnh lên Firebase Storage và lấy đường dẫn URL của ảnh.
* **React Native Image Picker**: Dùng để mở thư viện ảnh của thiết bị và chọn ảnh.

**2. Khai báo state**

****

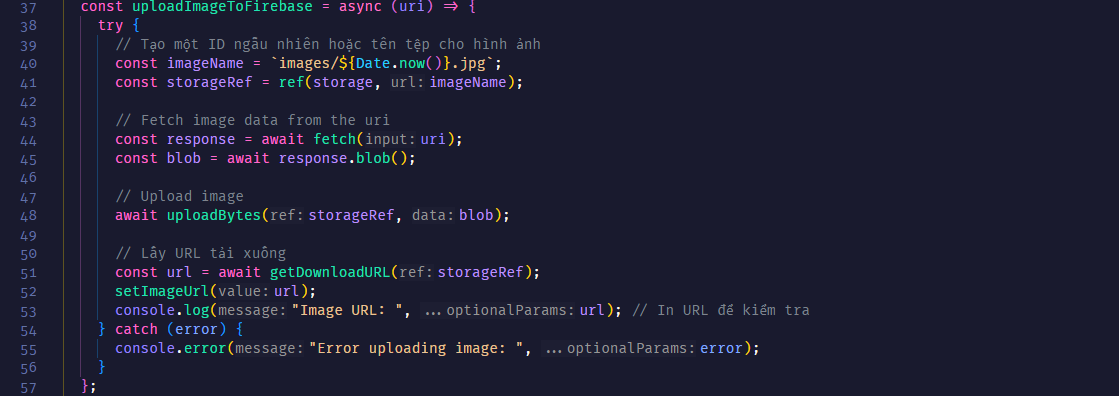
* **textName, textPrice, textDescription**: Các state lưu trữ tên, giá, và mô tả sản phẩm.
* **imageUri**: Lưu trữ đường dẫn cục bộ của hình ảnh được chọn từ thiết bị.
* **imageUrl**: Lưu trữ đường dẫn URL của hình ảnh sau khi được tải lên Firebase Storage.

**3. Chọn và tải lên hình ảnh**

****

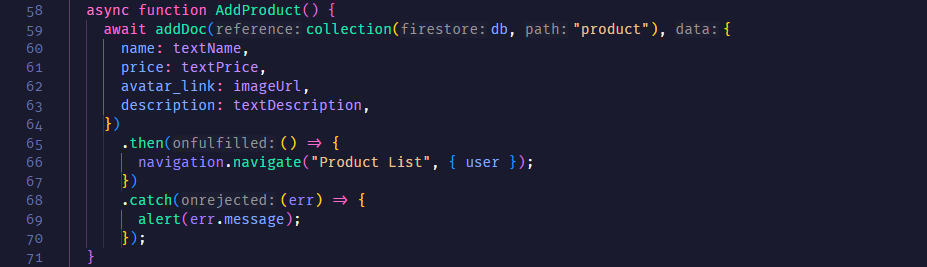
* **selectImage**: Dùng launchImageLibrary để mở thư viện ảnh trên thiết bị và cho phép người dùng chọn một ảnh. Nếu người dùng không hủy và chọn ảnh thành công, đường dẫn ảnh (uri) được lưu vào imageUri, sau đó hàm uploadImageToFirebase sẽ tải ảnh lên Firebase Storage.

**4. Tải ảnh lên Firebase Storage**

****

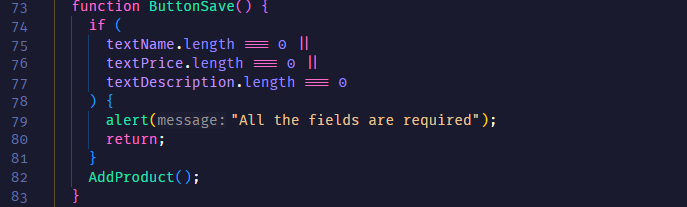
* **uploadImageToFirebase**: Tạo tên tệp ngẫu nhiên cho hình ảnh, sau đó tải ảnh lên Firebase Storage dưới dạng blob. Khi tải xong, lấy URL tải xuống và lưu vào imageUrl.

**5. Thêm sản phẩm mới vào Firestore**

****

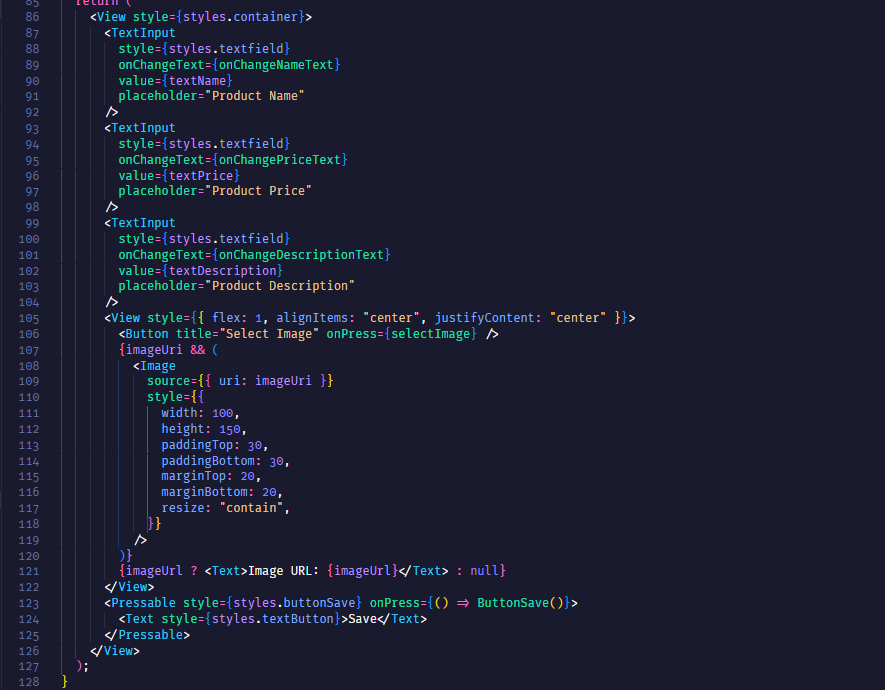
* **AddProduct**: Dùng addDoc để thêm một sản phẩm mới vào Firestore, bao gồm name, price, avatar\_link (URL của hình ảnh), và description. Sau khi thêm thành công, sẽ điều hướng người dùng trở lại màn hình "Product List".

**6. Button "Save" để lưu sản phẩm**

****

* **ButtonSave**: Kiểm tra xem tất cả các trường thông tin (tên, giá, mô tả) đã được điền hay chưa. Nếu chưa, hiển thị thông báo yêu cầu điền đủ thông tin. Nếu đã đầy đủ, gọi hàm AddProduct để lưu sản phẩm.

**7. Giao diện (UI)**

****

* **TextInput**: Các trường nhập liệu cho tên sản phẩm, giá, và mô tả.
* **Button**: Nút "Select Image" để mở thư viện ảnh, hiển thị hình ảnh đã chọn và URL của ảnh nếu đã tải lên thành công.
* **Pressable**: Nút "Save" để lưu sản phẩm.

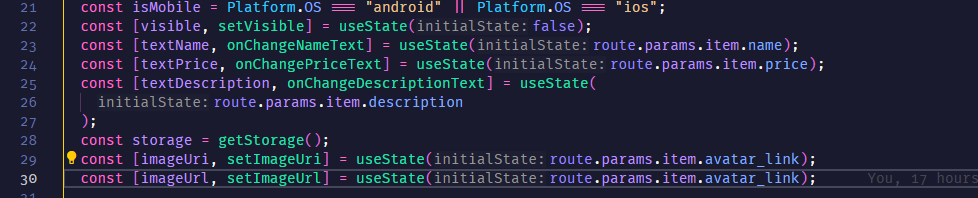
VI, Xây dựng tính năng sửa xóa sản phẩm

**1. Imports các dependencies cần thiết**

****

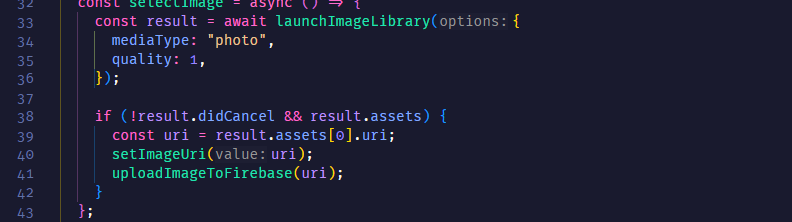
* **Firebase Firestore**: updateDoc để cập nhật thông tin sản phẩm và deleteDoc để xóa sản phẩm khỏi Firestore.
* **Firebase Storage**: uploadBytes, getDownloadURL để tải hình ảnh lên Firebase Storage và lấy URL của hình ảnh.
* **React Native Image Picker**: Dùng để mở thư viện ảnh trên thiết bị và chọn ảnh.
* **Dialog và AwesomeAlert**: Dùng để hiển thị hộp thoại xác nhận xóa sản phẩm.

**2. Khai báo state**

****

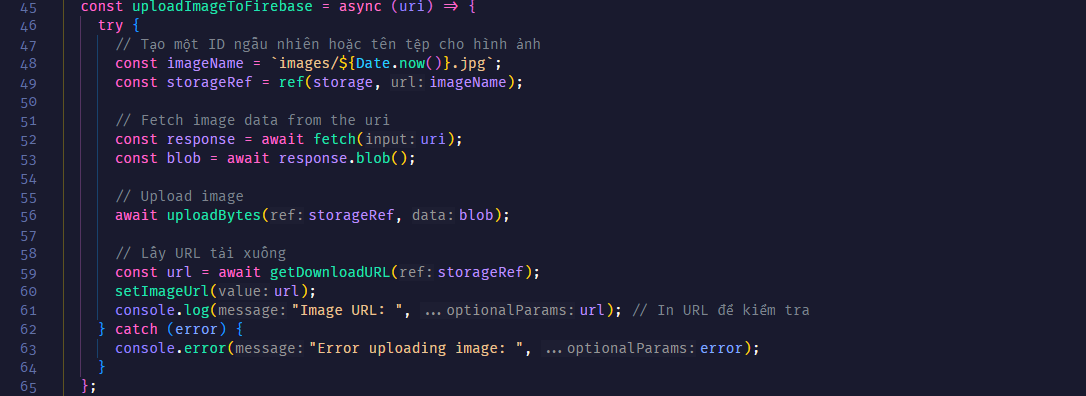
* **textName, textPrice, textDescription**: Khởi tạo với giá trị từ sản phẩm hiện tại, để người dùng có thể chỉnh sửa thông tin này.
* **imageUri, imageUrl**: Dùng để lưu trữ đường dẫn hình ảnh cục bộ và đường dẫn URL của hình ảnh trên Firebase.

**3. Chọn và tải lên hình ảnh**

****

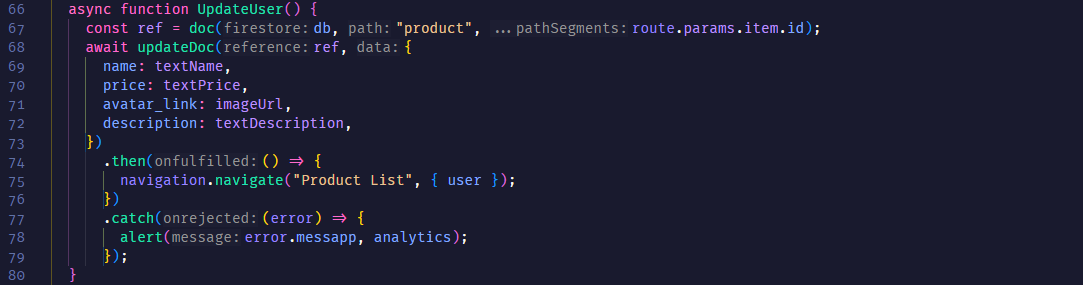
* **selectImage**: Dùng launchImageLibrary để mở thư viện ảnh của thiết bị và cho phép người dùng chọn ảnh. Nếu chọn thành công, uri của ảnh sẽ được lưu và hàm uploadImageToFirebase sẽ tải ảnh lên Firebase Storage.

**4. Tải ảnh lên Firebase Storage**

****

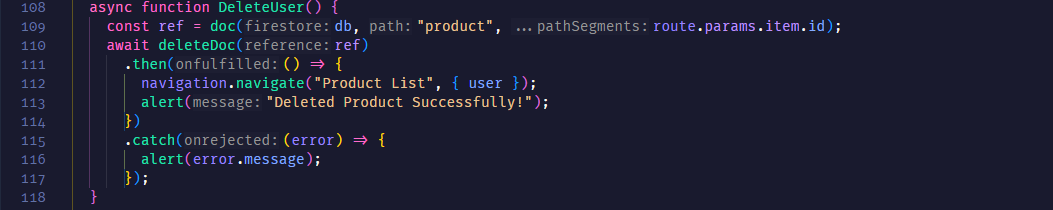
* **uploadImageToFirebase**: Tải ảnh lên Firebase Storage dưới dạng blob. Sau khi hoàn thành, URL của ảnh được lấy về và lưu vào imageUrl.

**5. Cập nhật sản phẩm**

****

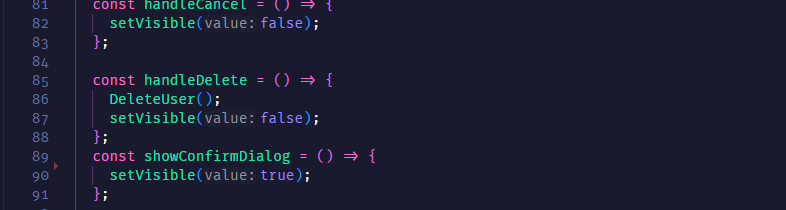
* **UpdateUser**: Cập nhật sản phẩm hiện tại trong Firestore với thông tin mới, bao gồm tên, giá, URL hình ảnh, và mô tả sản phẩm.

**6. Xóa sản phẩm**

****

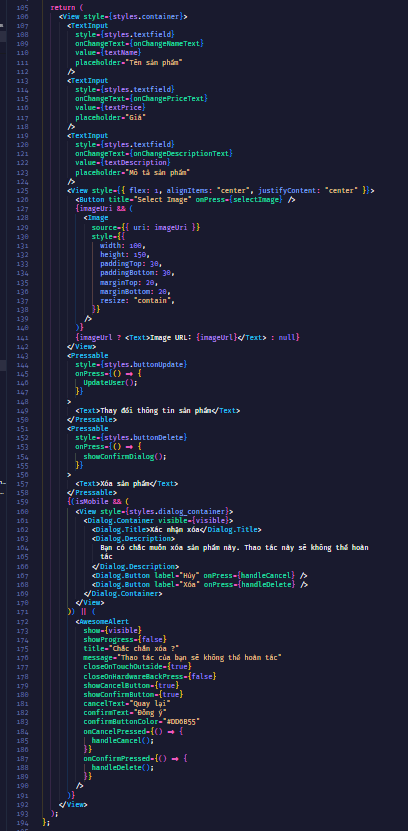
* **DeleteUser**: Xóa sản phẩm khỏi Firestore và hiển thị thông báo sau khi xóa thành công.

**7. Hộp thoại xác nhận xóa**

****

* **showConfirmDialog**: Hiển thị hộp thoại xác nhận khi người dùng bấm nút "Xóa sản phẩm".
* **Dialog**: Dùng trên thiết bị mobile để hiển thị hộp thoại xác nhận xóa.
* **AwesomeAlert**: Dùng trên web để hiển thị thông báo xác nhận.

**8. Giao diện (UI)**

****

* **TextInput**: Các trường nhập liệu để chỉnh sửa thông tin sản phẩm (tên, giá, mô tả).
* **Button**: Nút để chọn hình ảnh và hiển thị ảnh đã chọn.
* **Pressable**: Nút "Thay đổi thông tin sản phẩm" và "Xóa sản phẩm".