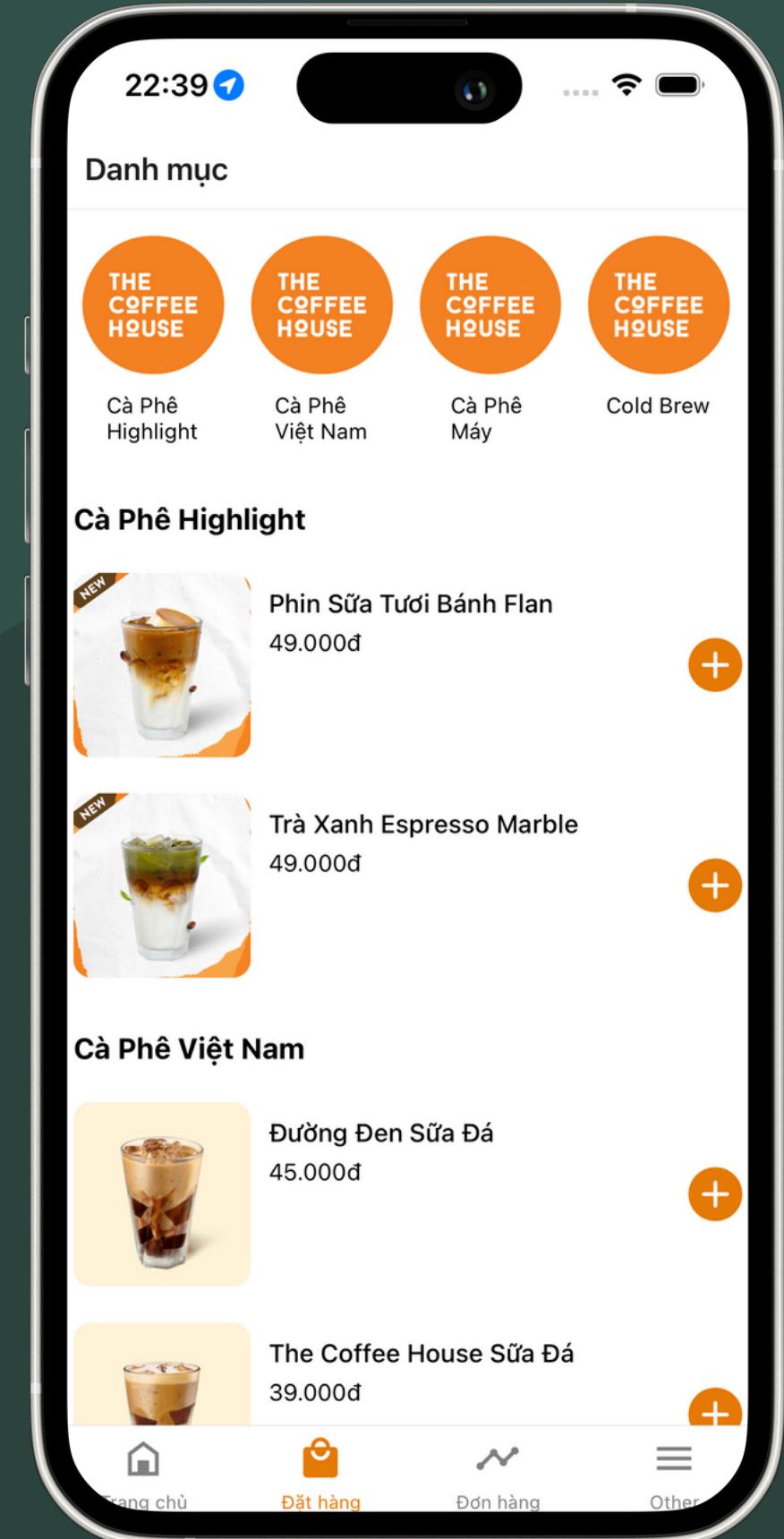


The Coffee House Mobile App

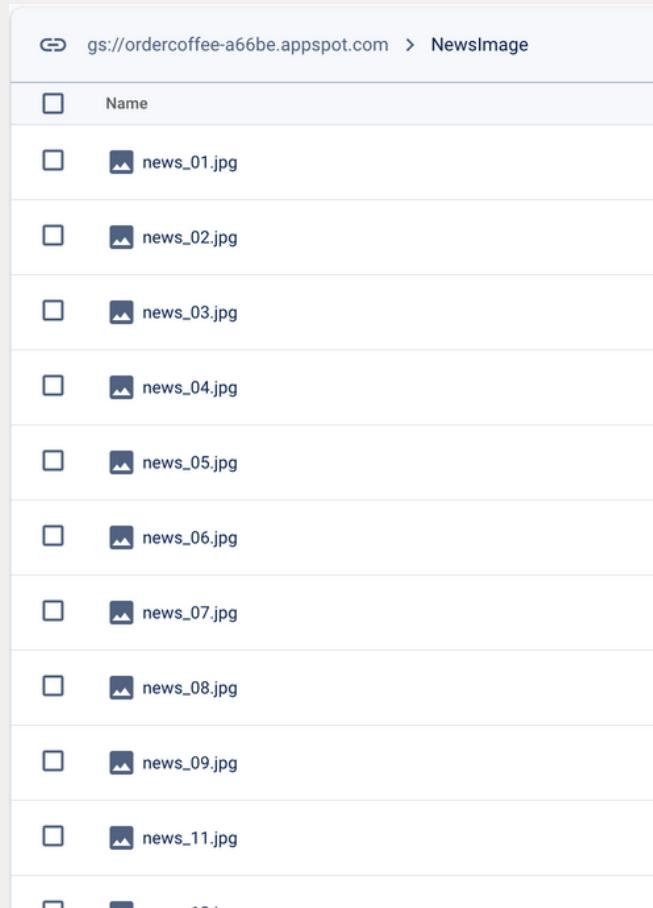
63130803 - VŨ MINH NGA



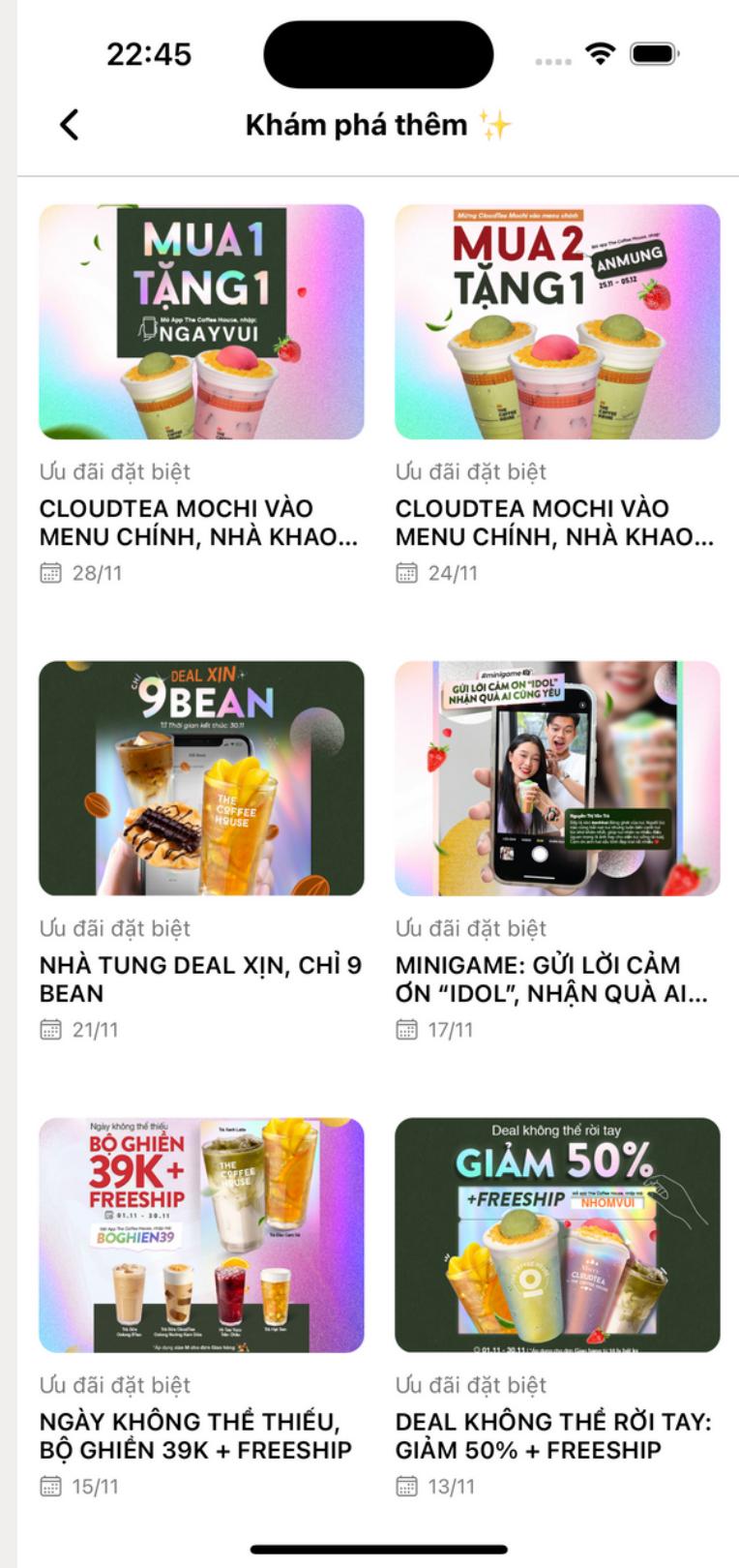
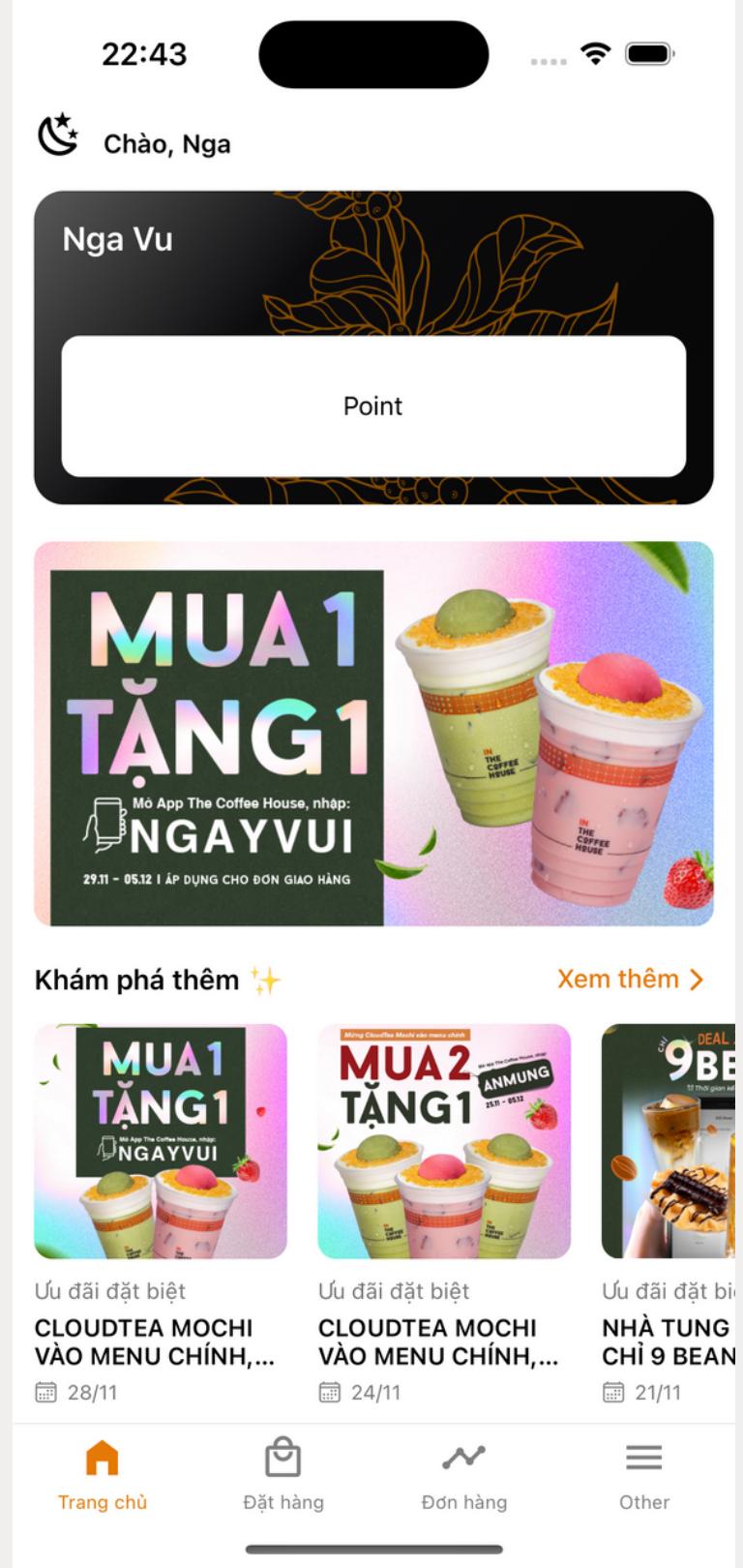
1. Giao diện xem tin tức của quán

Trang chủ của App hiển thị thông tin cập nhật mới nhất và các khuyến mãi của quán.

- Sử dụng Firebase FireStore lưu trữ nội dung cập nhật từ QTV sau đó fetch lên App.
- Sử dụng FireStorage để lưu trữ hình ảnh



The screenshot shows the Firestore interface with the path `home > TblNews > news01`. It displays a list of documents from `news01` to `news15`. On the left, there are buttons for `+ Start collection`, `TblBills`, `TblMenus`, `TblNews` (which is selected), and `TblUsers`. On the right, there is a button `+ Add document` and a list of document IDs: news01, news02, news03, news04, news05, news06, news07, news08, news09, news11, news12, news13, news14, and news15.



1.Giao diện xem tin tức của quán

```
● ● ●
1 <SafeAreaView style={styles.container}>
2   <AnimatedScrollView
3     showsVerticalScrollIndicator={false}
4     refreshControl={
5       <RefreshControl refreshing={refreshing} onRefresh={onRefresh} />
6     }
7   <View style={styles.header}>
8     <WeatherIcon />
9     <View>
10    <Text style={styles.greeting}>{greetingMessage}</Text>
11  </View>
12  <MemberCard userData={userData} style={styles.MemberCard} />
13  {isLoading || refreshing ? (
14    <View style={styles.loadingContainer}>
15      <LottieView
16        source={require('../assets/animations/christmas.json')}
17        style={{width: windowHeight * 0.5, height: windowHeight * 0.5}}
18        autoPlay
19        loop
20        withTiming
21      />
22    </View>
23  ) : (
24    <>
25      <Advertisement userData={userData} images={images} />
26      <News newsData={newsData} newsImages={newsImages} />
27    </>
28  )}
29  </AnimatedScrollView>
30 </SafeAreaView>
```

1.ScrollView và AnimatedScrollView:

- ScrollView được sử dụng để đảm bảo rằng nội dung không bị che khuất bởi thanh tiêu đề hoặc thanh trạng thái của điện thoại.
- AnimatedScrollView là một ScrollView được kết hợp với Animated để thêm hiệu ứng chuyển động.

2.RefreshControl:

- refreshControl được sử dụng để thêm chức năng kéo xuống để làm mới nội dung.
- Khi người dùng kéo xuống, sự kiện onRefresh sẽ được kích hoạt để xử lý làm mới dữ liệu.

3.Header:

- Một phần header chứa một biểu tượng thời tiết (WeatherIcon) và một lời chào (greetingMessage).
- Các thành phần này được đặt trong một View để có thể tùy chỉnh kiểu dáng.

4.MemberCard:

- Một thẻ thành viên (MemberCard) được hiển thị thông tin về người dùng.

5.Loading Animation:

- Nếu đang tải dữ liệu hoặc đang làm mới, một animation sẽ hiển thị bằng cách sử dụng thư viện Lottie.
- Animation được hiển thị là một tệp JSON có tên là "christmas.json".

6.Advertisement và News:

- Nếu không có tình trạng tải hoặc làm mới, hiển thị hai thành phần: quảng cáo (Advertisement) và tin tức (News).
- Dữ liệu của thành phần này được truyền qua từ userData, images, newsData, và newsImages.

7.Tổng quan:

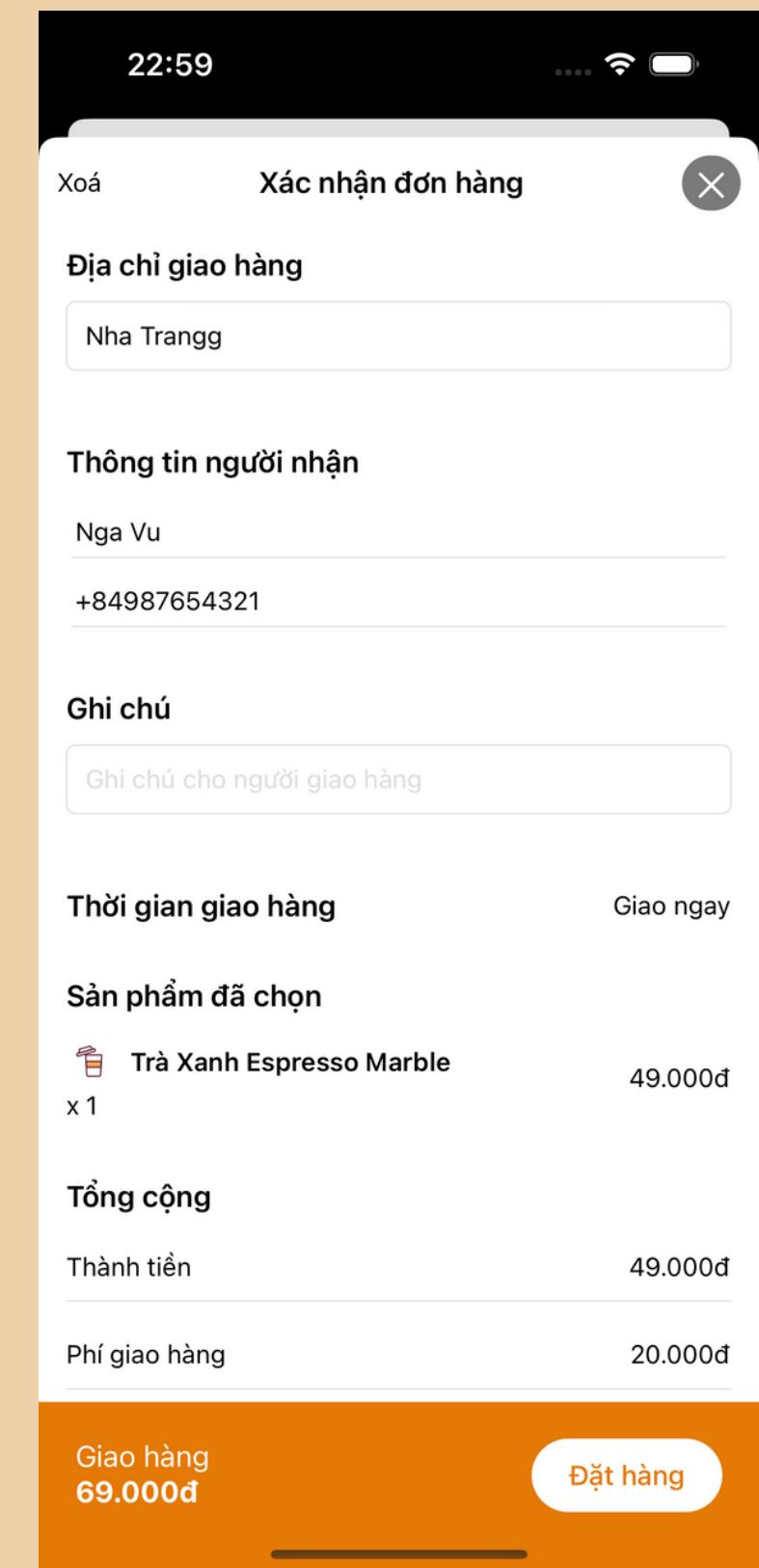
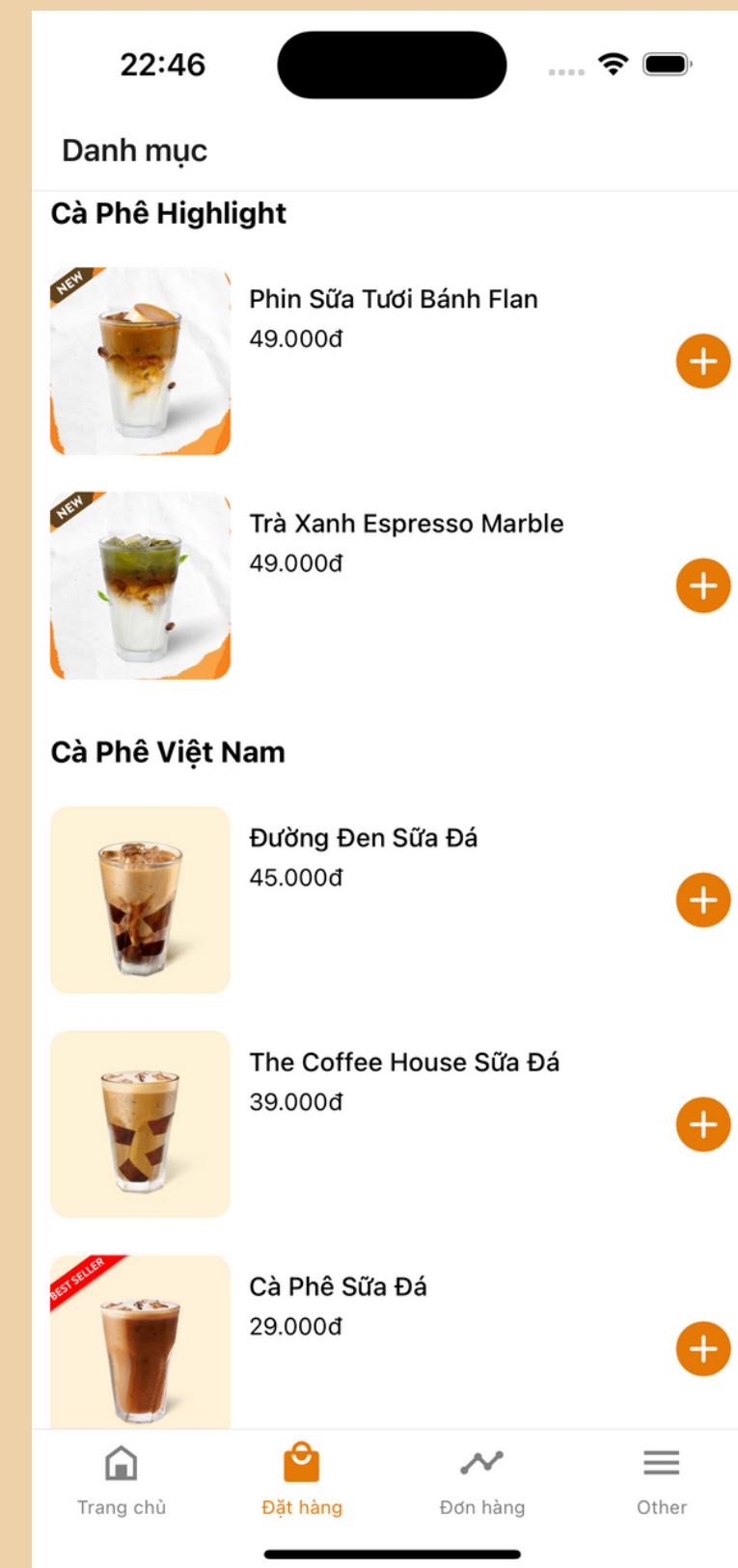
- Mã nguồn được tổ chức gọn gàng với sử dụng Animated cho các hiệu ứng chuyển động.
- Nếu có lỗi hoặc trạng thái làm mới (refreshing), sẽ hiển thị animation thích hợp.
- Nếu không có lỗi, hiển thị quảng cáo và tin tức.



2. Giao diện đặt hàng

Giao diện đặt hàng của App hiển thị danh sách món và hỗ trợ đặt hàng.

- Sử dụng Firebase FireStore lưu trữ nội dung về món.
- Sử dụng FireStorage để lưu trữ hình ảnh



2.Giao diện đặt hàng

```
● ● ●  
1  {isLoading || refreshing ? (  
2    <View style={styles.loadingContainer}>  
3      <LottieView  
4        source={require '../../../../../assets/animations/christmas.json'}  
5        style={{width: windowHeight * 0.5, height: windowHeight * 0.5}}  
6        autoPlay  
7        loop  
8        withTiming  
9      />  
10     </View>  
11   ) : (  
12     <>  
13     <MenuList menus={menus} />  
14     <ProductList  
15       menus={menus}  
16       images={images}  
17       onProductPress={handleProductPress}  
18       onPlusIconPress={handlePlusIconPress}  
19       quantity={quantity}  
20     />  
21   </>  
22 )}
```

1.Kiểm tra isLoading và refreshing:

- Điều kiện isLoading || refreshing kiểm tra xem ứng dụng đang trong quá trình tải dữ liệu (isLoading) hoặc đang thực hiện làm mới dữ liệu (refreshing) không.
- Nếu một trong hai điều kiện đó đúng, nghĩa là ứng dụng đang trong trạng thái tải hoặc làm mới, thì sẽ hiển thị một animation giáng sinh.

2.Hiển thị Animation khi đang tải hoặc làm mới:

- Nếu ứng dụng đang trong trạng thái tải (isLoading) hoặc làm mới (refreshing), sẽ hiển thị một container với một animation Lottie giáng sinh.
- Animation được đặt trong một View có kiểu dáng được xác định bởi styles.loadingContainer.
- Kích thước của animation được xác định bằng cách sử dụng windowHeight và set chiều rộng, chiều cao là 0.5 của windowHeight.
- Animation được thiết lập để tự động phát và lặp lại với autoPlay và loop.

3.Hiển thị MenuList và ProductList khi không tải hoặc làm mới:

- Ngược lại, nếu không có trạng thái tải (isLoading) hoặc làm mới (refreshing), thì sẽ hiển thị hai thành phần: MenuList và ProductList.
- MenuList nhận dữ liệu menus và hiển thị danh sách các menu.
- ProductList nhận dữ liệu menus, images, quantity và các hàm xử lý sự kiện (onProductPress, onPlusIconPress) để hiển thị danh sách sản phẩm.

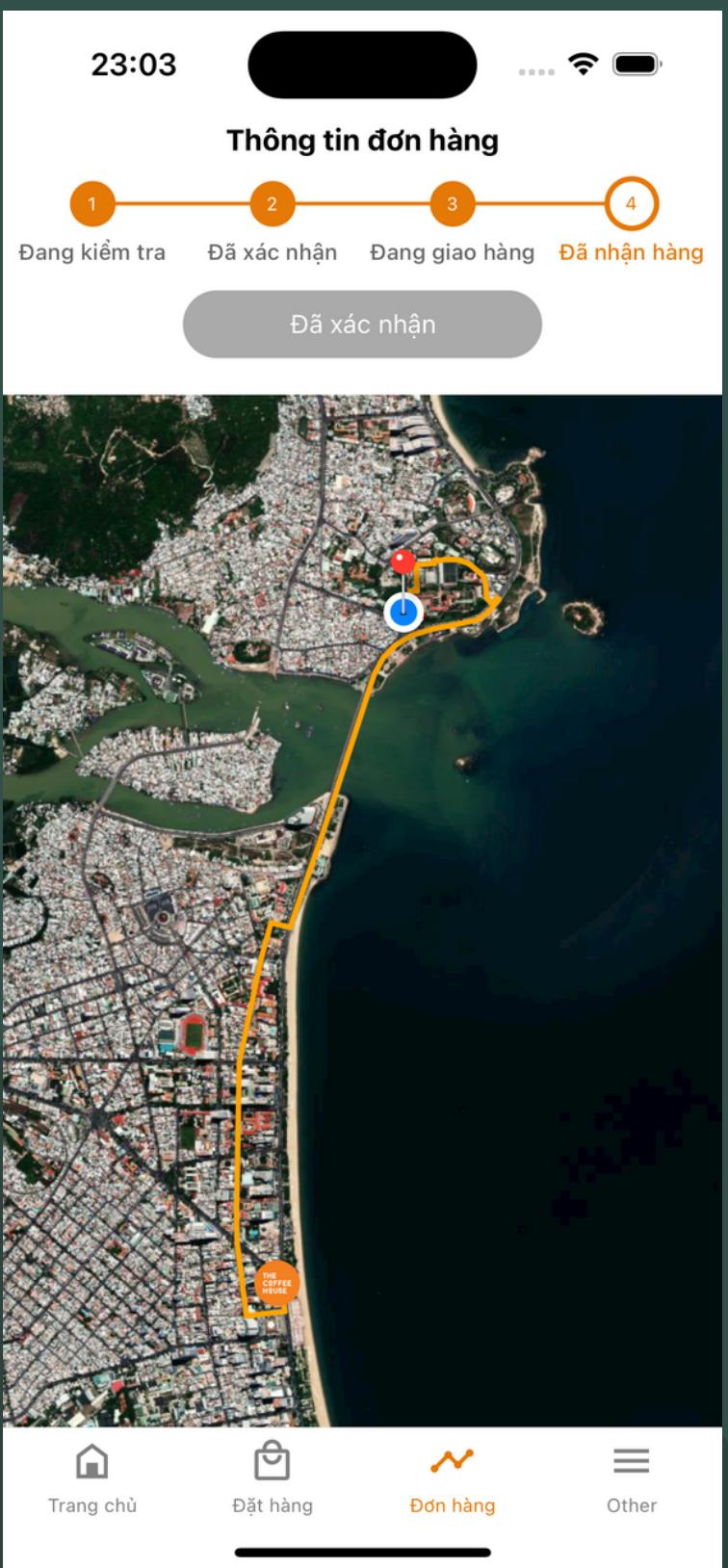
4.Dữ liệu đầu vào:

- menus là dữ liệu danh sách menu.
- images là dữ liệu hình ảnh liên quan đến sản phẩm.
- quantity là số lượng sản phẩm được chọn.

2.Giao diện Tracking đơn hàng

Giao diện tracking đơn hàng của App hiển thị thông tin giao hàng.

- Sử dụng OpenStreetMap để get location.



2.Giao diện Tracking đơn hàng

1.MapView:

- Thư viện MapView được sử dụng để tạo một thành phần hiển thị bản đồ trong ứng dụng.
- style={styles.map} đặt kiểu dáng cho bản đồ, được xác định trong đối tượng styles.
- region={region} thiết lập vùng hiển thị của bản đồ dựa trên một đối tượng region, có thể chứa thông tin về latitude, longitude và các thông số khác.

2.Polyline:

- routeSegments.map được sử dụng để vẽ các đoạn đường (Polyline) trên bản đồ.
- Mỗi đoạn đường được xác định bởi một mảng segment của các điểm có tọa độ (latitude, longitude).
- strokeWidth={3} và strokeColor="orange" đặt kiểu dáng cho đường vẽ.

3.Marker:

- Marker được sử dụng để đánh dấu các điểm cụ thể trên bản đồ.
- coordinate xác định vị trí của Marker dựa trên latitude và longitude.
- Đối với Marker thứ nhất, nó chỉ đơn giản là một điểm đánh dấu ở vị trí hiện tại của người dùng.
- secondMarkerCoordinate xác định vị trí và hình ảnh của Marker thứ hai, có hình dạng và ảnh đại diện cho một đối tượng cụ thể.
- movingIconCoordinate xác định vị trí của Marker thứ ba, sử dụng một biểu tượng (có thể là một hình ảnh hoặc biểu tượng tùy chỉnh).

4.Marker với Hình ảnh:

- Marker thứ hai chứa một hình ảnh (Image) được hiển thị trong một View.
- Hình ảnh này là biểu tượng của một đối tượng cụ thể và được hiển thị bên trong Marker với kích thước và kiểu dáng được xác định.

5.Biker Component:

- Marker thứ ba chứa một thành phần (Biker) thay vì một hình ảnh, người dùng có thể tạo một thành phần React Native tùy chỉnh để đại diện cho Marker này.

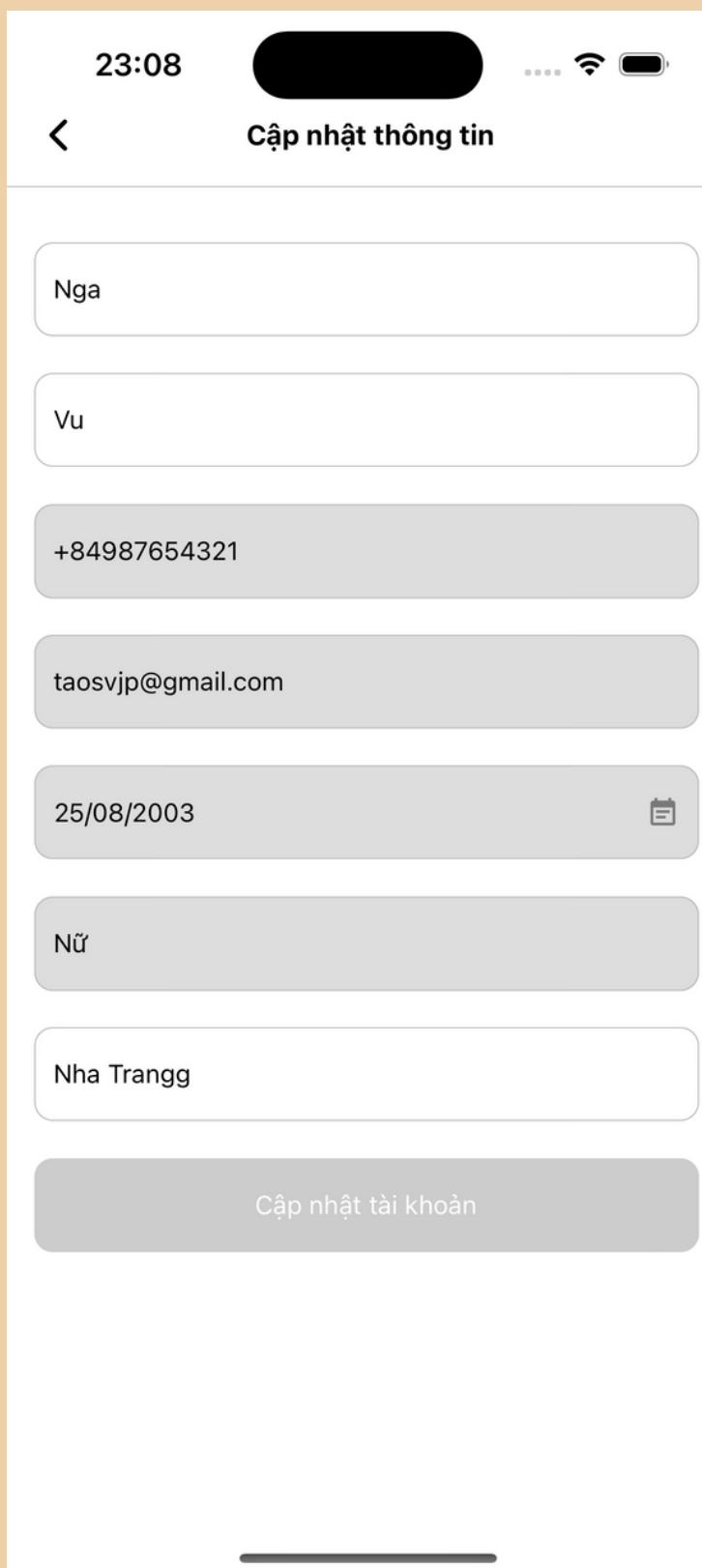
```
1 <MapView
2   style={styles.map}
3   region={region}
4   showsUserLocation={true}
5   zoomControlEnabled={true}
6   mapType="satellite">
7   {routeSegments.map((segment, index) => (
8     <Polyline
9       key={index}
10      coordinates={segment}
11      strokeWidth={3}
12      strokeColor="orange"
13    />
14  ))}
15
16   <Marker
17     coordinate={{
18       latitude: region.latitude,
19       longitude: region.longitude,
20     }}
21   />
22   <Marker coordinate={secondMarkerCoordinate}>
23     <View
24       style={{
25         justifyContent: 'center',
26         alignItems: 'center',
27         borderRadius: 50,
28       }}>
29       <Image
30         source={require('../assets/images/tch.png')}
31         style={{
32           width: 25,
33           height: 25,
34           borderRadius: 50,
35         }}
36         resizeMode="contain"
37       />
38     </View>
39   </Marker>
40
41   <Marker coordinate={movingIconCoordinate}>
42     <Biker />
43   </Marker>
44 </MapView>
```



2. Giao diện thông tin

Giao diện thông tin người dùng và lịch sử đặt hàng.

- Sử dụng Firebase FireStore lưu trữ nội dung.



Lịch sử đơn hàng		
	Trà Xanh Espresso Marble, The Coffee House Sữa Đá	108.000đ
	Trạng thái: Đã giao 03/01/2024 - 23:01:53	
	Trà Xanh Espresso Marble, Cà Phê Sữa Đá, The Coffee House Sữa Đá	137.000đ
	Trạng thái: Đã giao 03/01/2024 - 22:05:58	
	Trà Xanh Espresso Marble	69.000đ
	Trạng thái: Chưa giao 03/01/2024 - 22:13:34	
	Trà Xanh Espresso Marble, The Coffee House Sữa Đá, Bạc Siêu	195.000đ
	Trạng thái: Chưa giao 03/01/2024 - 22:00:54	
	Phin Sữa Tươi Bánh Flan, Trà Xanh Espresso Marble, Đường Đen Sữa Đá	163.000đ
	Trạng thái: Đã giao 03/01/2024 - 22:25:09	
	Đường Đen Sữa Đá	65.000đ
	Trạng thái: Đã giao 21/12/2023 - 21:05:25	

2.Giao diện đặt hàng

1.View và styles.billContainer:

- Một View chứa toàn bộ thành phần đơn hàng.
- styles.billContainer được sử dụng để xác định kiểu dáng của container.

2.View (70% width):

- Một View con với chiều rộng chiếm 70% của billContainer.
- flexDirection: 'row' được sử dụng để xác định hướng chia các phần tử con theo chiều ngang.

3.Receipt Component:

- Một thành phần React Native được gọi là Receipt được hiển thị, có thể là biểu tượng hóa hóa đơn hoặc biểu tượng tương tự.

4.View (marginLeft và Text):

- Một View con để chứa các thông tin chi tiết của đơn hàng.
- marginLeft: 10 để thêm khoảng cách giữa Receipt và các phần tử text khác.
- Một Text element hiển thị danh sách các mục trong đơn hàng. Dữ liệu này được trích xuất từ thuộc tính items của item và được nối thành một chuỗi bằng cách sử dụng join(', ').
- Một Text element hiển thị trạng thái của đơn hàng, dựa trên thuộc tính done. Nếu done là true, hiển thị "Đã giao", ngược lại hiển thị "Chưa giao".
- Một Text element hiển thị thời gian tạo đơn hàng, sử dụng hàm formatDate(item.createdAt) để định dạng thời gian.

5.View (30% width):

- Một View con với chiều rộng chiếm 30% của billContainer.
- alignItems: 'flex-end' và justifyContent: 'center' được sử dụng để căn chỉnh văn bản theo cạnh phải và canh giữa.

6.Text (Giá trị đơn hàng):

- Một Text element hiển thị giá trị đơn hàng. Dữ liệu này được định dạng bằng hàm formatPrice(item.totalBill) để hiển thị số tiền có dạng chuỗi và thêm đơn vị đồng (đ) vào cuối.

7.styles.billText:

- Một số kiểu dáng chung được áp dụng cho văn bản, như font weight và font size, thông qua styles.billText.

```
1 <View style={styles.billContainer}>
2   <View
3     style={{
4       width: '70%',
5       flexDirection: 'row',
6     }>
7     <Receipt />
8     <View
9       style={{
10        marginLeft: 10,
11      }>
12     <Text
13       style={[
14         styles.billText,
15         {
16           fontWeight: '700',
17           fontSize: 13,
18         },
19       ]}>
20       {item.items.map(orderItem => orderItem.title).join(' ', )}
21     </Text>
22     <Text style={styles.billText}>
23       Trạng thái:
24       {item.done ? 'Đã giao' : 'Chưa giao'}
25     </Text>
26     <Text style={styles.billText}>{formatDateTime(item.createdAt)}</Text>
27   </View>
28 </View>
29 <View
30   style={{
31     width: '30%',
32     alignItems: 'flex-end',
33     justifyContent: 'center',
34   }}>
35   <Text
36     style={[
37       styles.billText,
38       {
39         fontWeight: '700',
40         fontSize: 15,
41       },
42     ]}>
43     {formatPrice(item.totalBill)}đ
44   </Text>
45 </View>
46 </View>
47 </View>
```

2.Giao diện đặt hàng

1. ScrollView và styles.body:

- ScrollView được sử dụng để tạo một khu vực cuộn cho các thành phần bên trong nếu nội dung vượt quá kích thước màn hình.
- styles.body là kiểu dáng được áp dụng cho ScrollView.

2.Các TouchableOpacity và TextInput:

- Có nhiều TouchableOpacity chứa TextInput để người dùng có thể nhập thông tin vào các trường cụ thể.
- activeOpacity={0.5} làm cho thành phần trở nên trong suốt khi được nhấn.
- styles.inputField là kiểu dáng chung được áp dụng cho các thành phần nhập liệu.

3.Các Trường Thông Tin Người Dùng:

- Mỗi TouchableOpacity chứa một TextInput để nhập thông tin.
- value và onChangeText được sử dụng để liên kết giá trị và sự kiện thay đổi của TextInput với các biến và hàm tương ứng (ví dụ: firstName, handleFirstNameChange).

4.Các Trường Không Thể Chính Sửa:

- Có một số TouchableOpacity được thiết lập để không thể chỉnh sửa (editable={false}) và có màu nền khác để chỉ ra rằng chúng không thể được chỉnh sửa.
- Các TextInput bên trong chứa dữ liệu tương ứng từ userData như phone, email, dob và gender.

5.Trường Giới Tính với Icon:

- Một TouchableOpacity chứa TextInput không thể chỉnh sửa, hiển thị giới tính từ userData.gender.
- Có một biểu tượng MaterialCommunityIcons (biểu tượng văn bản) được sử dụng để hiển thị thông tin giới tính.

6.Trường Địa Chỉ và Nút Cập Nhật:

- Một TouchableOpacity chứa TextInput cho địa chỉ người dùng, với giá trị được liên kết với biến address và sự kiện thay đổi được xử lý bởi handleAddressChange.
- styles.submitBtn là một kiểu dáng cho nút cập nhật (Update button).
- isButtonActive được sử dụng để kiểm soát tính khả dụng của nút (khi true, nút có thể được nhấn).
- Nút được kích hoạt hoặc vô hiệu hóa dựa trên giá trị của isButtonActive.
- Sự kiện onPress được liên kết với hàm handleUpdate để xử lý khi người dùng nhấn nút cập nhật.

```
1  <ScrollView style={styles.body}>
2    <TouchableOpacity activeOpacity={0.5} style={styles.inputField}>
3      <TextInput value={firstName} onChangeText={handleFirstNameChange} />
4    </TouchableOpacity>
5
6    <TouchableOpacity activeOpacity={0.5} style={styles.inputField}>
7      <TextInput value={lastName} onChangeText={handleLastNameChange} />
8    </TouchableOpacity>
9
10   <TouchableOpacity activeOpacity={0.5}
11     style={[styles.inputField, {backgroundColor: '#dddddd'}]}>
12     <TextInput editable={false}>{userData.phone}</TextInput>
13   </TouchableOpacity>
14
15   <TouchableOpacity
16     activeOpacity={0.5}
17     style={[styles.inputField, {backgroundColor: '#dddddd'}]}>
18     <TextInput editable={false}>{userData.email}</TextInput>
19   </TouchableOpacity>
20
21   <TouchableOpacity
22     activeOpacity={0.5}
23     style={[styles.inputField, {backgroundColor: '#dddddd'}]}>
24     <View
25       style={{
26         flexDirection: 'row',
27         width: '100%',
28         justifyContent: 'space-between',
29       }}>
30       <TextInput editable={false}>{userData.dob}</TextInput>
31       <MaterialCommunityIcons
32         name="calendar-text"
33         size={18}
34         color={colors.darkGray}
35       />
36     </View>
37   </TouchableOpacity>
38
39   <TouchableOpacity
40     activeOpacity={0.5}
41     editable={false}
42     style={[styles.inputField, {backgroundColor: '#dddddd'}]}>
43     <Text>{userData.gender ? 'Nữ' : 'Nam'}</Text>
44   </TouchableOpacity>
45
46   <TouchableOpacity activeOpacity={0.5} style={styles.inputField}>
47     <TextInput value={address} onChangeText={handleAddressChange} />
48   </TouchableOpacity>
49
50   <TouchableOpacity
51     activeOpacity={0.5}
52     style={[
53       styles.submitBtn,
54       {backgroundColor: isButtonActive ? colors.mainColor : '#cccccc'},
55     ]}
56     disabled={!isButtonActive}
57     onPress={handleUpdate}>
58       <Text style={{color: 'white'}}>Cập nhật tài khoản</Text>
59     </TouchableOpacity>
60
61 </ScrollView>
```