

**BÁO CÁO DỰ ÁN CUỐI KỲ**  
**HỌC KỲ 2, NĂM HỌC 2023-2024**  
**CT484: PHÁT TRIỂN ỨNG DỤNG DI ĐỘNG**

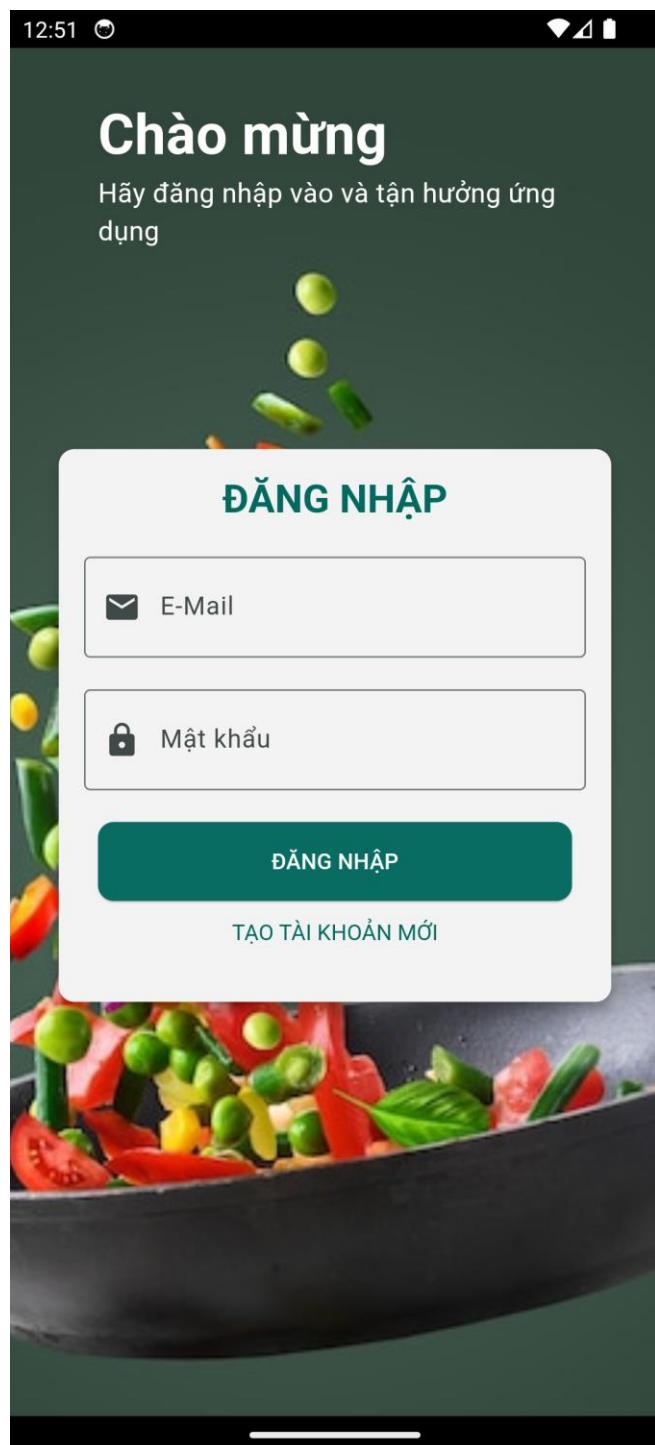
- **Tên dự án/ứng dụng:** **Ứng dụng nấu ăn**
- **Link GitHub mã nguồn:** <https://github.com/23-24Sem2-Courses/ct48403-project-doctorhuy.git>
- **MSSV 1:** **B2014569**
- **Họ tên SV 1:** **Giang Đại Huy**
- **Lớp học phần:** **03**

**I. Tổng quan**

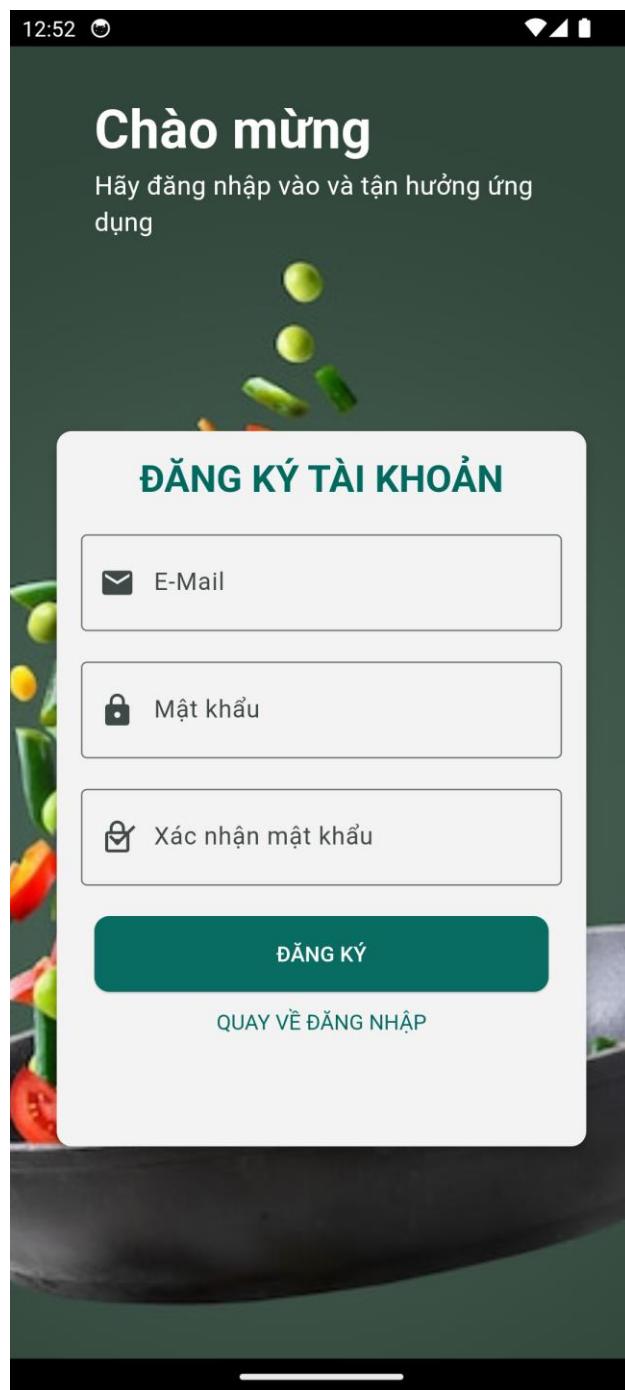
- **Miêu tả dự án/ứng dụng:** Ứng dụng nấu ăn là một ứng dụng cung cấp cho người sử dụng học những cách nấu ăn khác nhau. Trong ứng dụng sẽ cung cấp các công thức theo từng thể loại chế biến như món xào, món nướng, .v.vv. Hơn nữa những người dùng khác có thể thêm những công thức yêu thích của mình. Ngoài ra người dùng còn có thể tự tạo một công thức cho riêng mình và có thể công khai cho tất cả mọi người thấy hoặc là riêng tư.

**II. Chi tiết các chức năng**

- Chức năng/giao diện 1: Chức năng đăng nhập và đăng ký**
  - **Miêu tả chức năng/giao diện:** Chức năng này được dùng để xác thực người dùng trước khi bắt sử dụng ứng dụng đảm bảo an toàn trong việc sử dụng.
  - **Ảnh chức năng/giao diện:**
    - + Giao diện đăng nhập:



+ Giao diện đăng ký:



- **Chi tiết cài đặt cho chức năng đăng ký và đăng nhập:**
  - Các widget được sử dụng trong cả đăng ký và đăng nhập gồm: SizedBox, Column, Card, RoundedRectangleBorder, Container, BoxConstraints, Form, SingleChildScrollView, Text, CircularProgressIndicator, TextButton, ElevatedButton, TextFormField, InputDecoration, Stack, Icon, BoxDecoration, DecorationImage, NetworkImage, Flexible, ValueListenableBuilder, TextStyle,

- Scaffold, EdgeInsets.symmetric, MaterialTapTargetSize.shrinkWrap, BorderRadius.circular.
- Những thư viện/plugin mà các giao diện có sử dụng như: dart:async, dart:convert, flutter/foundation, flutter/material, provider/provider, http, shared\_preferences, flutter\_dotenv.
  - Vai trò của từng thư viện trên:
    - **Thư viện dart:async:** dùng để hỗ trợ cho lập trình bất đồng bộ với các lớp Future
    - **Thư viện dart:convert:** cung cấp các phương thức để mã hóa và giải mã dữ liệu sang định dạng JSON.
    - **Thư viện flutter/foundation:** Chứa các lớp và hàm tiện ích cơ bản cho việc phát triển ứng dụng flutter, bao gồm ChangeNotifier, một lớp cơ sở để triển khai mô hình quản lý trạng thái.
    - **Thư viện flutter/material:** thư viện chính của flutter, cung cấp các widget và công cụ để xây dựng giao diện người dùng theo Material Design.
    - **Thư viện provider/provider:** được sử dụng để triển khai mô hình quản lý trạng thái (state management) trong ứng dụng Flutter thông qua provider pattern. Cụ thể trong ứng dụng sử dụng provider để truy cập tới AuthManager để xác thực người dùng.
    - **Thư viện http:** được sử dụng để thực hiện các yêu cầu http, như gửi yêu cầu POST để xác thực người dùng thông qua API của Firebase Authentication.
    - **Thư viện shared\_preferences:** Được sử dụng để lưu trữ và truy xuất dữ liệu cục bộ như thông tin xác thực trong ứng dụng Flutter. Cụ thể hơn là truy xuất thông tin về token xác thực trong ứng dụng.
    - **Thư viện flutter\_dotenv:** được dùng để tải các biến môi trường từ tệp .env vào ứng dụng flutter. Hơn nữa nó giúp bảo mật thông tin nhạy cảm trong ứng dụng flutter như API Key.
  - Trong ứng dụng có sử dụng giải pháp quản lý trạng thái chia sẻ như ValueNotifier và Provider.
    - **Đối với ValueNotifier:** được sử dụng để quản lý trạng thái của quá trình submit trong màn hình đăng nhập hoặc đăng ký. Cụ thể sẽ có một biến \_isSubmitting là một ValueNotifier<bool> và giá trị hiện tại của nó được khởi tạo là false có nghĩa là người dùng chưa nhấn nút đăng nhập hoặc đăng ký. Khi người dùng nhấn nút đăng ký hoặc đăng nhập thì giá trị của biến này sẽ được gán bằng true đồng thời trong ValueListenableBuilder đang lắng nghe ValueNotifier này sẽ cập nhật lại UI dựa vào giá trị mà nó đang lắng nghe. Trong trường hợp này thì sẽ hiển thị một

CircularProgressingIndicator để ám chỉ rằng hệ thống đang xử lý yêu cầu đăng ký hoặc đăng nhập.

- Đối với Provider: được sử dụng để cung cấp và quản lý đối tượng AuthManager, một lớp đảm nhận việc quản lý quá trình xác thực người dùng. AuthManager sẽ được định nghĩa ở mức cao nhất của cây, cụ thể là trong file main để có thể được truy cập ở bất cứ đâu trong cây. Khi người dùng đăng ký hoặc đăng nhập thì sẽ sử dụng context.read<AuthManager> () để truy cập đến các phương thức tương ứng để thực hiện hành động mong muốn. Khi thực hiện gọi phương thức login hoặc signup trong Provider thì sẽ thay đổi trạng thái đồng thời sẽ báo lại cho các widget khác lắng nghe cập nhật lại giao diện một cách tự động.
- Chức năng này thực hiện đọc và lưu trữ dữ liệu trong thiết bị cục bộ của người dùng bằng cách sử dụng SharedPreferences. Trong trường hợp này, dữ liệu được lưu trữ dưới dạng cấu trúc JSON. Trong trường hợp này cấu trúc Json được truyền và lưu trữ dữ liệu trả về từ API của Firebase Authentication và cấu trúc gồm các thuộc tính như {"idToken", "localId", "email", "expiresIn", "refreshToken", "kind", "registered"}

## 2. **Chức năng/giao diện 2:** Chức năng hiển thị công thức nấu ăn của người dùng

- **Miêu tả chức năng/giao diện:** Giao diện “Công thức nấu ăn của tôi” là nơi người dùng có thể xem danh sách các công thức mà họ đã tạo. Giao diện này bao gồm một thanh tìm kiếm và một toggle button để chuyển đổi chế độ hiển thị các công thức và mặc định chọn chế độ chi tiết.
- **Ảnh chức năng/giao diện:**



#### - Chi tiết cài đặt giao diện công thức nấu ăn của tôi:

- Các widget được sử dụng trong giao diện này gồm: Scaffold, AppBar, Text, IconButton, Icon, Column, Padding, FoodSearchField, EdgeInsets.all, Row, ToggleViewMode, Expanded, FutureBuilder, CircularProgressIndicator, RefreshIndicator, ValueListenableBuilder, UserFoodRecipeDetail, UserFoodRecipeLargeMode, FloatingActionButton.
- Các thư viện/plugin được sử dụng trong giao diện này gồm:  
flutter/foundation, flutter/material, go\_router, provider
- Vai trò của mỗi thư viện:
  - **Thư viện flutter/foundation:** Chứa các lớp và hàm tiện ích cơ bản cho việc phát triển ứng dụng flutter, bao gồm ChangeNotifier, một lớp cơ sở để triển khai mô hình quản lý trạng thái.

- **Thư viện flutter/material:** thư viện chính của flutter, cung cấp các widget và công cụ để xây dựng giao diện người dùng theo Material Design.
  - **Thư viện go\_router:** có vai trò cung cấp một cách tiện lợi và linh hoạt để quản lý các tuyến đường trong ứng dụng flutter. Trong ngữ cảnh của ứng dụng việc dùng go\_router giúp xác định các tuyến đường một cách dễ dàng đồng thời vẫn duy trì trạng thái của các trang.
  - **Thư viện provider/provider:** được sử dụng để triển khai mô hình quản lý trạng thái (state management) trong ứng dụng Flutter thông qua provider pattern. Cụ thể trong ứng dụng sử dụng provider để truy cập tới FoodRecipesManager để nạp và quan sát danh sách công thức và AuthManager để thực hiện việc đăng xuất.
- Trong giao diện “Công thức nấu ăn của tôi” có sử dụng provider để quan sát sự thay đổi trong danh sách công thức nấu ăn. Cụ thể trong lớp FoodRecipesManager sẽ định nghĩa một danh sách \_items dùng để chứa danh sách các công thức nấu ăn của người dùng tạo ra. Và tại nút root của cây đã định nghĩa provider này để có thể truy xuất ở mọi nút trong cây. Trong giao diện “Công thức nấu ăn của tôi” có sử dụng Consumer của provider để quan sát và cập nhật tự động danh sách \_items. Nếu người dùng thêm mới, chỉnh sửa, hoặc xóa một công thức nào đó thì việc thay đổi sẽ được phản ánh thông qua cập nhật UI tự động. Ngoài ra còn sử dụng một ValueNotifier để quan sát việc lựa chọn chế độ hiển thị và cập nhật dựa trên chế độ mà người dùng đã chọn. Ban đầu sẽ có một ValueNotifier có tên là \_isDetailView để lưu trữ trạng thái hiển thị và có giá trị khởi tạo là true, có nghĩa là sẽ hiển thị ở chế độ chi tiết theo mặc định. Tại giao diện chính sẽ sử dụng một ValueListenableBuilder và quan sát \_isDetailView để quyết định việc hiển thị.
- Giao diện này thực hiện đọc dữ liệu từ một dịch vụ lưu trữ của Firebase. Cấu trúc JSON sẽ có dạng như sau:
- ```
foodRecipes {
    Id công thức 1 (String) {
        creatorId (String)
        imageUrl (String)
        ingredient (String)
        isPublic (String)
        processing (String)
        title (String)
        type: (String)
    }
}
```

}

.....

**3. Chức năng/giao diện 3: Chế độ hiển thị công thức nấu ăn của người dùng**

- **Miêu tả chức năng/giao diện:** Chức năng này nằm trong trang “Công thức nấu ăn của tôi” với vai trò là kiểm soát việc hiển thị tùy theo sở thích của người dùng. Khi người dùng chọn chế độ chi tiết thì danh sách sẽ hiển thị dưới dạng danh sách liệt kê gồm các thông tin như ảnh minh họa, tên công thức, và các nguyên liệu. Khi người dùng chọn chế độ lớn thì sẽ hiển thị những thẻ lớn chứa hình ảnh, tên công thức và các nguyên liệu.
- **Ảnh chức năng/giao diện:**
  - **Giao diện khi chọn chế độ chi tiết:**



➤ Giao diện khi chọn chế độ thẻ lớn:



- **Chi tiết cài đặt:**

- Các widget được sử dụng trong chức năng này:
  - Đối với chế độ chi tiết: Consumer, ListView.builder, Card, EdgeInsets.only, InkWell, BorderRadius.all, ListTile, Text, Image, NetworkImage, BoxFit.fill, AlertDialog, Center, Column, Row, TextButton.icon, TextButton, TextStyle.
  - Đối với chế độ thẻ lớn: Card, EdgeInsets.only, BorderRadius.all, Column, SizedBox, ClipRRect, Image, NetworkImage, Padding, Text, AlertDialog, Row, Center, TextButton.icon, Icon, TextStyle, TextButton.

- Trong chức năng này có sử dụng một vài thư viện như: flutter/material, go\_router, provider.
- Vai trò của từng thư viện:
  - **Thư viện flutter/material:** thư viện chính của flutter, cung cấp các widget và công cụ để xây dựng giao diện người dùng theo Material Design.
  - **Thư viện go\_router:** có vai trò cung cấp một cách tiện lợi và linh hoạt để quản lý các tuyến đường trong ứng dụng flutter. Trong ngữ cảnh của chức năng này thì go router dùng để điều hướng đến trang chi tiết công thức bên trong trang “Công thức nấu ăn của tôi”.
  - **Thư viện provider/provider:** được sử dụng để triển khai mô hình quản lý trạng thái (state management) trong ứng dụng Flutter thông qua provider pattern. Cụ thể trong chức năng này dùng để quan sát sự thay đổi trong danh sách \_items của người dùng, khi danh sách thay đổi thì giao diện sẽ tự động cập nhật.
- Chức năng này có sử dụng provider để quản lý trạng thái chia sẻ. Cụ thể trong FoodRecipesManager đã định nghĩa một danh sách \_item chứa các công thức nấu ăn của người dùng tạo và được định nghĩa ở nút gốc của toàn cây. Trong giao diện chính của chức năng này thì có sử dụng một consumer để quan sát sự thay đổi của danh sách \_item, khi có bất kỳ thay đổi nào xảy ra thì widget sẽ tự động xây dựng lại và cập nhật thay đổi.
- Chức năng này có đọc dữ liệu từ dịch vụ lưu trữ của Firebase thông qua provider FoodRecipesManager. Cấu trúc JSON mà chức năng này đọc sẽ có dạng như sau:

```

foodRecipes {
    Id công thức 1 (String) {
        creatorId (String)
        imageUrl (String)
        ingredient (String)
        isPublic (String)
        processing (String)
        title (String)
        type: (String)
    }
    .....
}
  
```

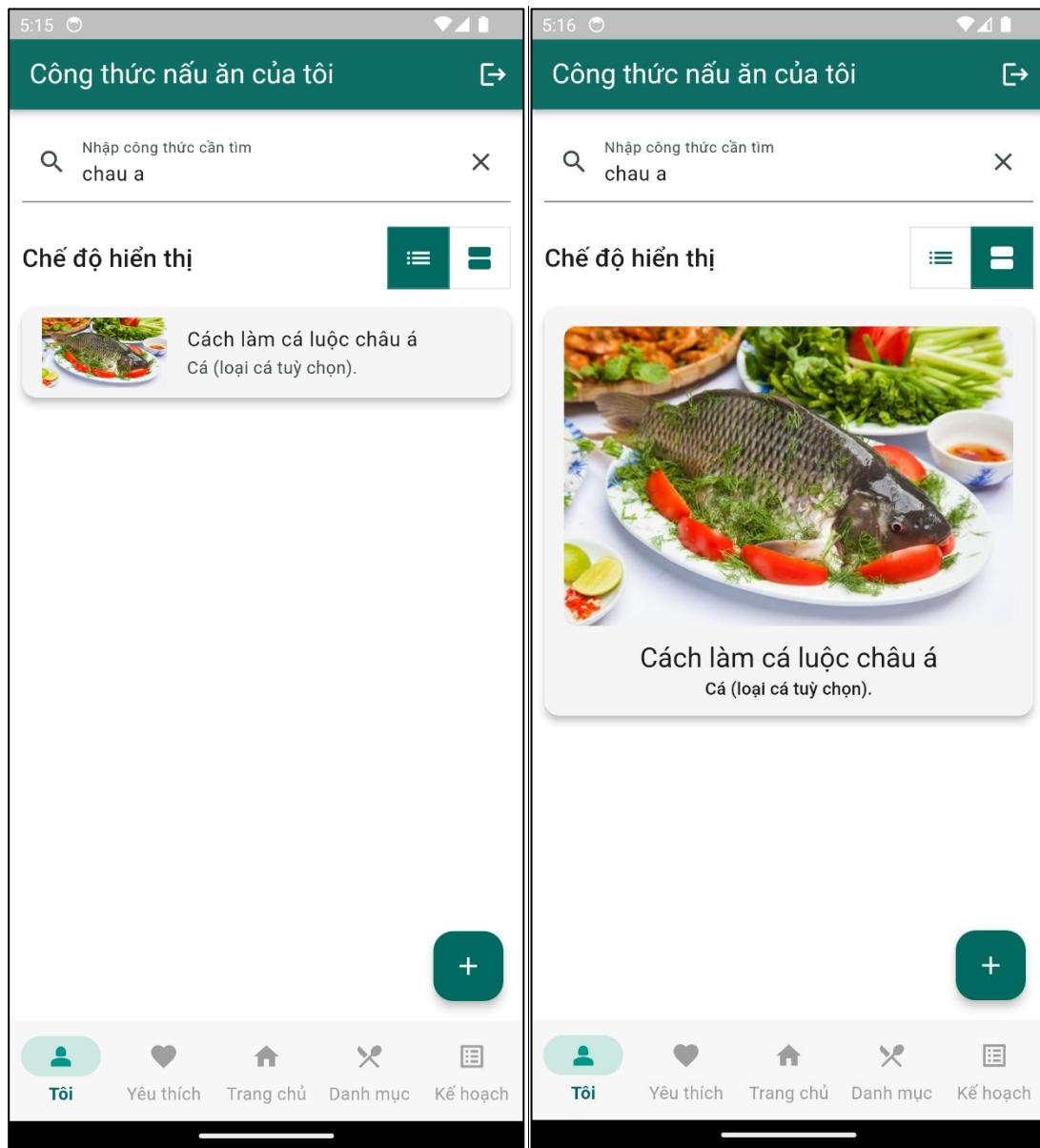
#### 4. Chức năng/giao diện 4: Chức năng tìm kiếm công thức

- **Miêu tả chức năng/giao diện:** Chức năng được sử dụng cho cả giao diện “Công thức nấu ăn của tôi” và “Trang chủ”. Ở chức năng này tùy thuộc vào giao diện mà người dùng đang chọn hiện tại mà cơ chế tìm kiếm sẽ khác nhau. Nếu người dùng

đang ở trang “Công thức nấu ăn của tôi” thì việc tìm kiếm sẽ trên danh sách sản phẩm của người dùng. Còn nếu người dùng đang ở “Trang chủ” thì việc tìm kiếm sẽ dựa vào danh sách toàn bộ sản phẩm.

- **Ảnh chức năng/giao diện:**

- Đối với trang “Công thức nấu ăn của tôi”:



- Đối với trang “Trang chủ”:



- Chi tiết cài đặt:

- Các widget được sử dụng trong chức năng này gồm: TextFormField, InputDecoration, UnderlineInputBorder, Icon, Box.
- Các thư viện được sử dụng trong chức năng gồm: flutter/material và provider.
- Vai trò của từng thư viện:
  - **Thư viện flutter/material:** thư viện chính của flutter, cung cấp các widget và công cụ để xây dựng giao diện người dùng theo Material Design.

- **Thư viện provider/provider:** được sử dụng để triển khai mô hình quản lý trạng thái (state management) trong ứng dụng Flutter thông qua provider pattern. Cụ thể trong chức năng tìm kiếm này thì được sử dụng để truy cập đến provider FoodRecipesManager
  - Chức năng này có sử dụng provider để quản lý trạng thái chia sẻ. Cụ thể lớp FoodRecipesManager sẽ chứa hai danh sách là `_items` của người dùng và `_allItems` chứa toàn bộ công thức. Provider này được định nghĩa ở nút gốc của cây để có thể truy xuất ở bất kỳ đâu trong cây. Trong chức năng tìm kiếm này. Ngoài ra bên trong FoodRecipesManager còn định nghĩa thêm `_searchText` để lưu trữ từ khóa tìm kiếm tại trang “Công thức nấu ăn của tôi” và `_searchHomeText` để lưu trữ từ khóa tìm kiếm tại trang “Trang chủ”. Thêm nữa còn định nghĩa thêm các hàm `setSearchText` và `setSearchHomeText` để thay đổi giá trị của `_searchText` và `_searchHomeText` và có `NotifyListeners()` để báo cho các widget lắng nghe cập nhật dữ liệu. Tại thanh tìm kiếm của chức năng tìm kiếm có sử dụng provider FoodRecipesManager để thực hiện hành động `setSearchText` hoặc `setSearchHomeText`. Tùy thuộc vào vị trí hiện tại của người dùng, nếu người dùng đang ở trang “Công thức nấu ăn của tôi” thì chức năng này sẽ gọi hành động `setSearchText`, còn nếu ở “Trang chủ” thì sẽ thực hiện hành động `setSearchHomeText`.
  - Chức năng này chỉ đọc dữ liệu từ dịch vụ lưu trữ của Firebase. Cấu trúc JSON sẽ có dạng như sau:
- ```
foodRecipes {
    Id công thức 1 (String) {
        creatorId (String)
        imageUrl (String)
        ingredient (String)
        isPublic (String)
        processing (String)
        title (String)
        type: (String)
    }
    .....
}
```

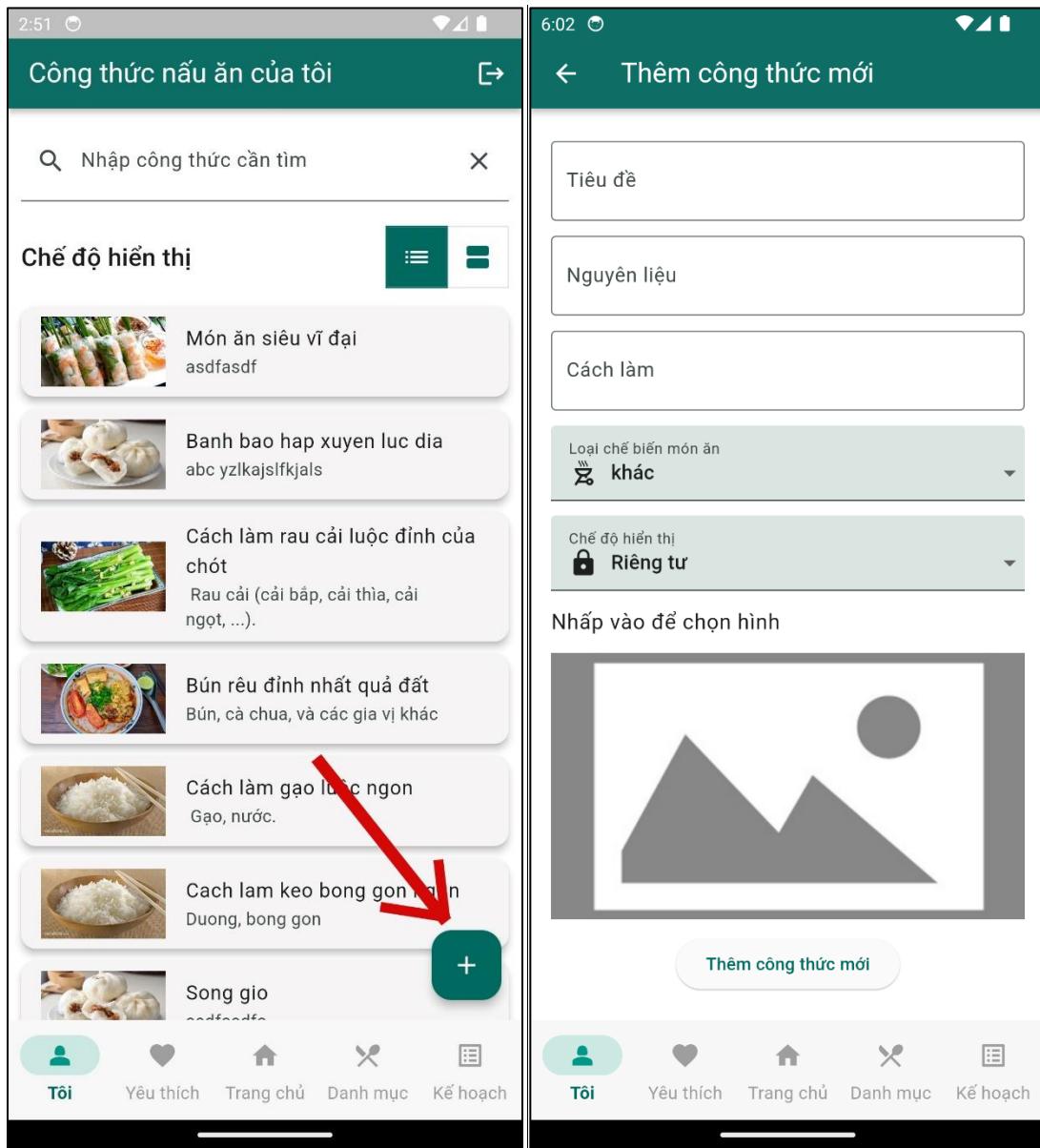
## 5. Chức năng/giao diện 5: Chức năng thêm/chỉnh sửa công thức nấu ăn của người dùng

- **Miêu tả chức năng/giao diện:** Chức năng này dùng để thêm mới một công thức nấu ăn mới hoặc chỉnh sửa một công thức nấu ăn đã có. Tùy thuộc vào hành động của người dùng mà dữ liệu hiển thị sẽ khác nhau. Tại giao diện “Công thức nấu ăn của tôi”, nếu người dùng chỉ muốn chỉnh sửa một công thức hiện có thì nhấn giữ vào công thức đó, sau đó sẽ xuất hiện một hộp thoại cho người dùng chọn hành

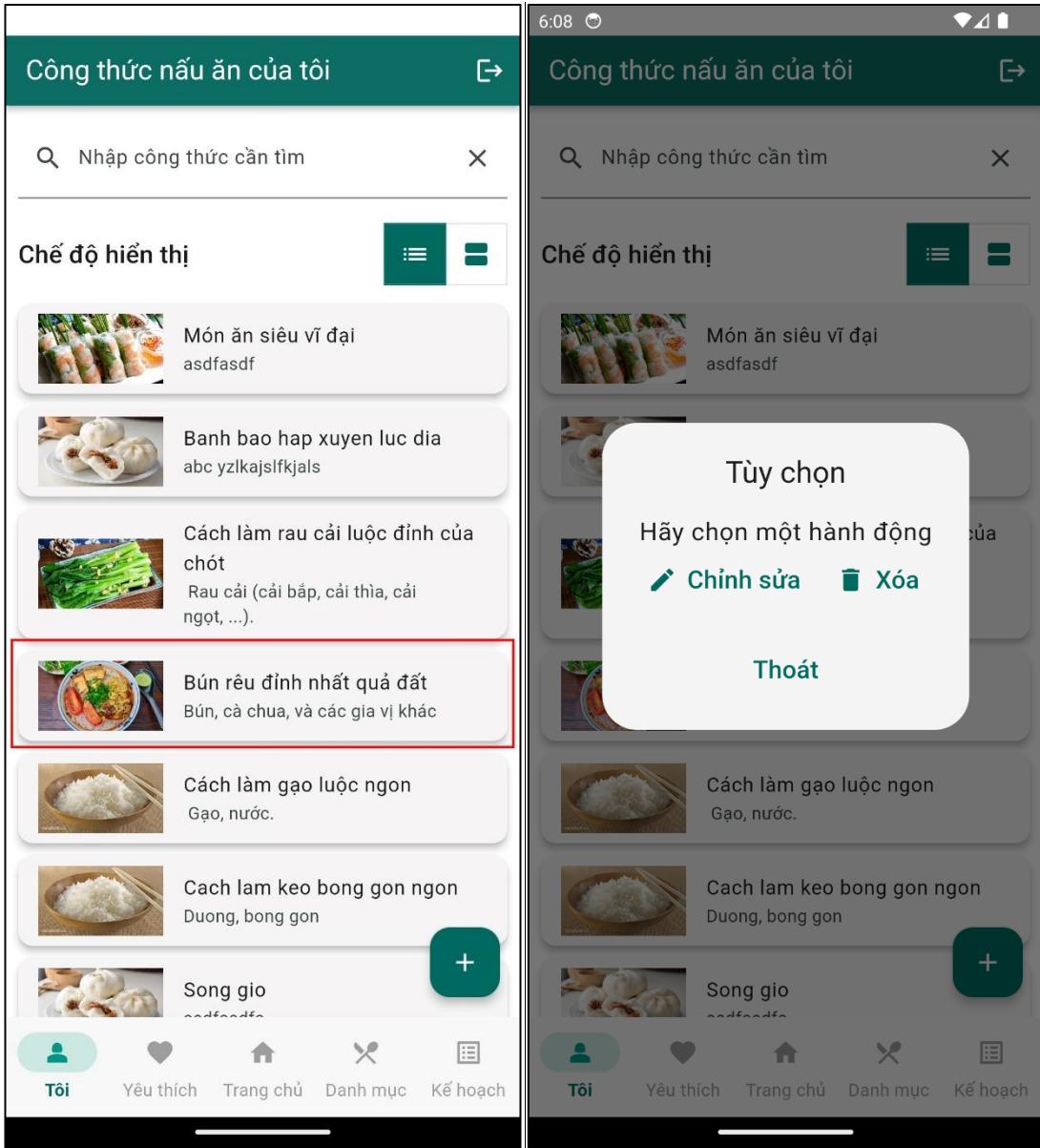
động. Nếu người dùng chọn “Chỉnh sửa” thì giao diện sẽ có dữ liệu tại các trường. Nếu người dùng muốn thêm mới thì nhấn vào FloatingActionButton tại trang “Công thức nấu ăn của tôi” để mở giao diện thêm mới. Đặc biệt là người dùng có thể chọn ảnh từ thư viện ảnh

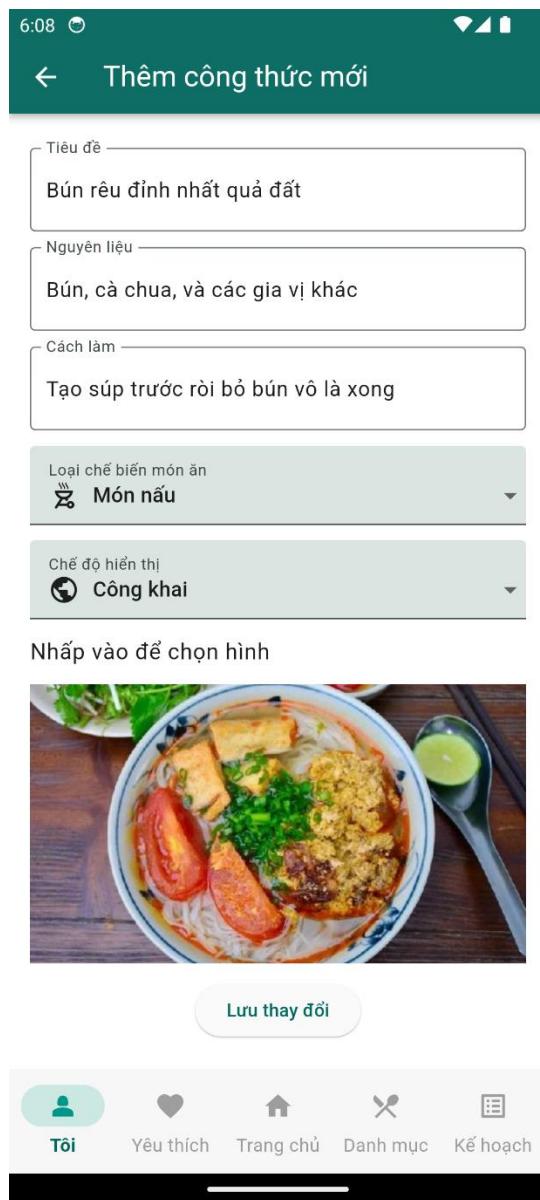
- **Ảnh chức năng/giao diện:**

➤ **Giao diện thêm công thức nấu ăn mới:**



➤ **Giao diện chỉnh sửa công thức nấu ăn hiện có:**





- **Chi tiết cài đặt chức năng thêm/chỉnh sửa công thức nấu ăn:**
  - Các widget được sử dụng trong chức năng này gồm: Scaffold, AppBar, Padding, EdgeInsets.all, SingleChildScrollView, Column, SizedBox, Container, Text, Stack, Image, NetworkImage, Positioned.fill, Material, InkWell, ElevatedButton, TextFormField, InputDecoration, OutlineInputBorder, DropdownButtonFormField, DropdownMenuItem, Icon.
  - Các thư viện được sử dụng gồm: flutter/material, image\_picker, provider, dart:developer, dart:io, firebase\_storage.
  - Vai trò của từng thư viện:

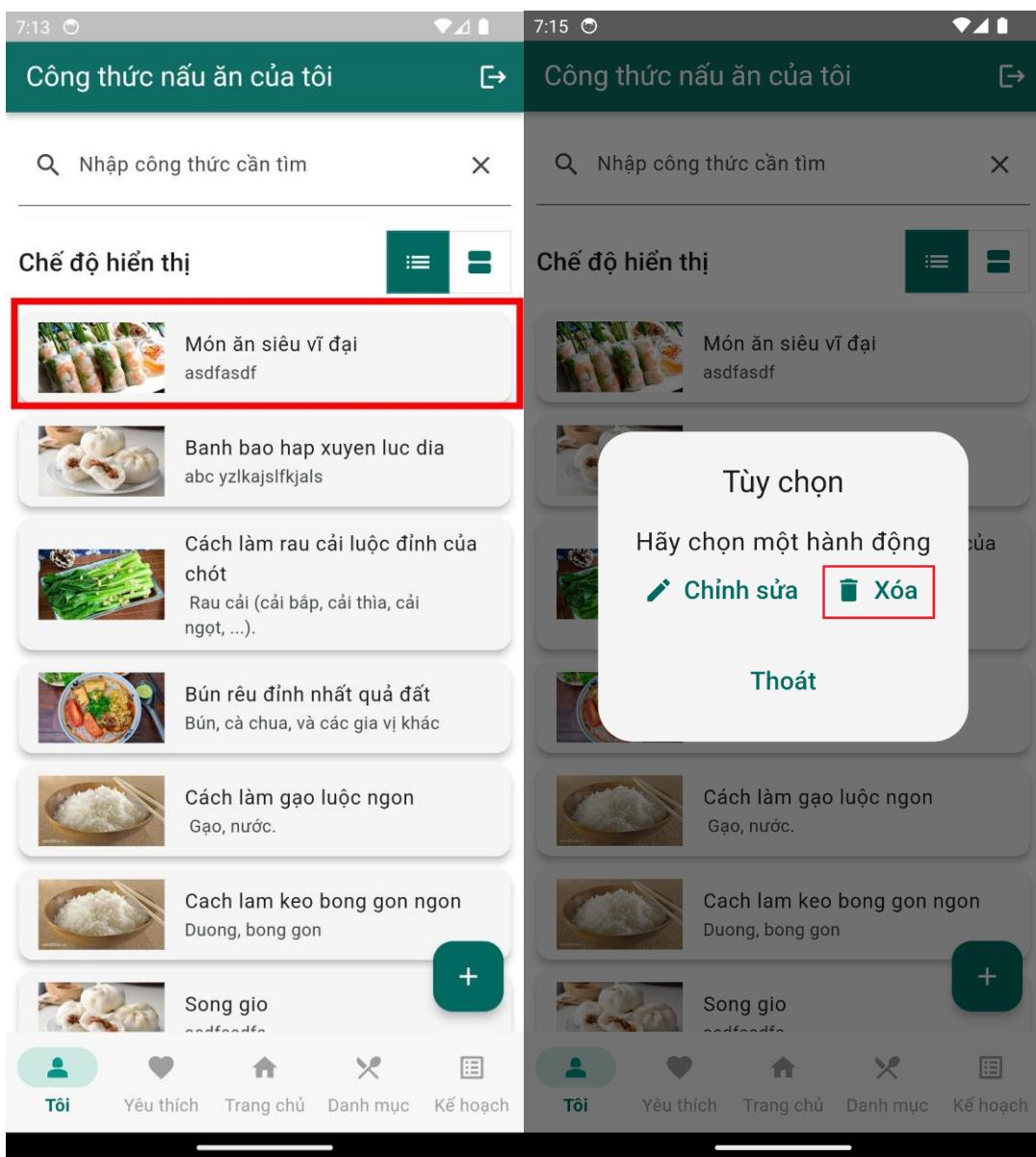
- **Thư viện flutter/material:** thư viện chính của flutter, cung cấp các widget và công cụ để xây dựng giao diện người dùng theo Material Design.
  - **Thư viện image\_picker:** cung cấp các công cụ để chọn hình ảnh từ thư viện ảnh của thiết bị hoặc chụp hình mới bằng camera. Cụ thể trong chức năng thêm/chỉnh sửa này thì dùng để chọn hình ảnh có sẵn từ thư viện ảnh trong máy.
  - **Thư viện provider/provider:** được sử dụng để triển khai mô hình quản lý trạng thái (state management) trong ứng dụng Flutter thông qua provider pattern. Cụ thể trong chức năng này thì dùng để truy xuất tới provider FoodRecipesManager để thực hiện hành động thêm mới công thức hoặc cập nhật công thức lên firebase.
  - **Thư viện dart:developer:** cung cấp các công cụ hỗ trợ phát triển ứng dụng như logging, profiling và debugging.
  - **Thư viện dart:io:** cung cấp các công cụ để tương tác với các tác vụ I/O như đọc ghi tập tin,vv.vv. Trong ngữ cảnh của ứng dụng thì dùng để tạo một File dựa vào đường dẫn khi người dùng chọn ảnh.
  - **Thư viện firebase\_storage:** được dùng để tương tác với dịch vụ lưu trữ của Firebase Storage. Cụ thể trong ngữ cảnh của chức năng này thì nó có nhiệm vụ là dùng để tải các file ảnh lên storage.
- Chức năng này có sử dụng provider để quản lý trạng thái chia sẻ. Cụ thể là lớp FoodRecipesManager sẽ định nghĩa thêm các phương thức thêm mới công thức, cập nhật công thức và các phương thức đều có NotifyListeners để báo cho các widget khác lắng nghe. Trong chức năng này thì khi người dùng hoàn thành thao tác thêm hoặc cập nhật thì sẽ gọi phương thức tương ứng trong provider FoodRecipesManager.
- Chức năng này có thực hiện đọc và lưu trữ dữ liệu lên dịch vụ lưu trữ của Firebase. Cấu trúc JSON của dữ liệu khi đọc và lưu trữ có dạng như sau:
- JSON khi đọc:**
- ```
Id công thức 1 (String) {
    imageUrl (String)
    ingredient (String)
    isPublic (String)
    processing (String)
    title (String)
    type: (String)
}
```
- JSON khi lưu trữ lên Firebase:**
- ```
Id công thức 1 (String) {
    creatorId (String)
```

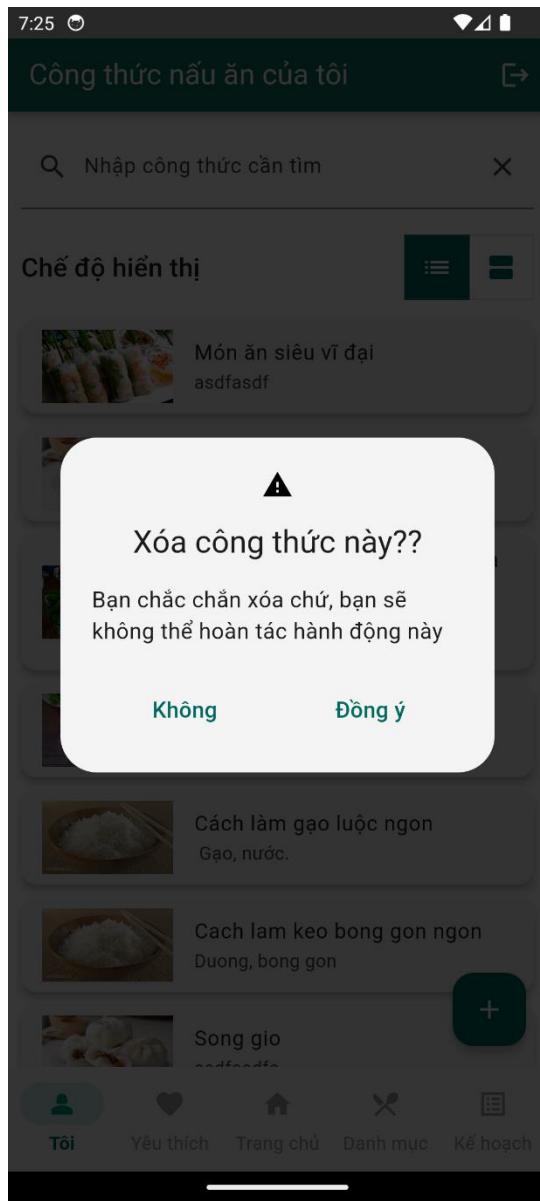
```
imageUrl (String)  
ingredient (String)  
isPublic (String)  
processing (String)  
title (String)  
type: (String)
```

```
}
```

## 6. Chức năng/giao diện 6: Chức năng xóa một công thức nấu ăn của người dùng

- **Miêu tả chức năng/giao diện:** Chức năng này dùng để xóa một công thức nấu ăn cụ thể nào đó của người dùng tại trang “Công thức nấu ăn của tôi”.
- **Ảnh chức năng/giao diện:**



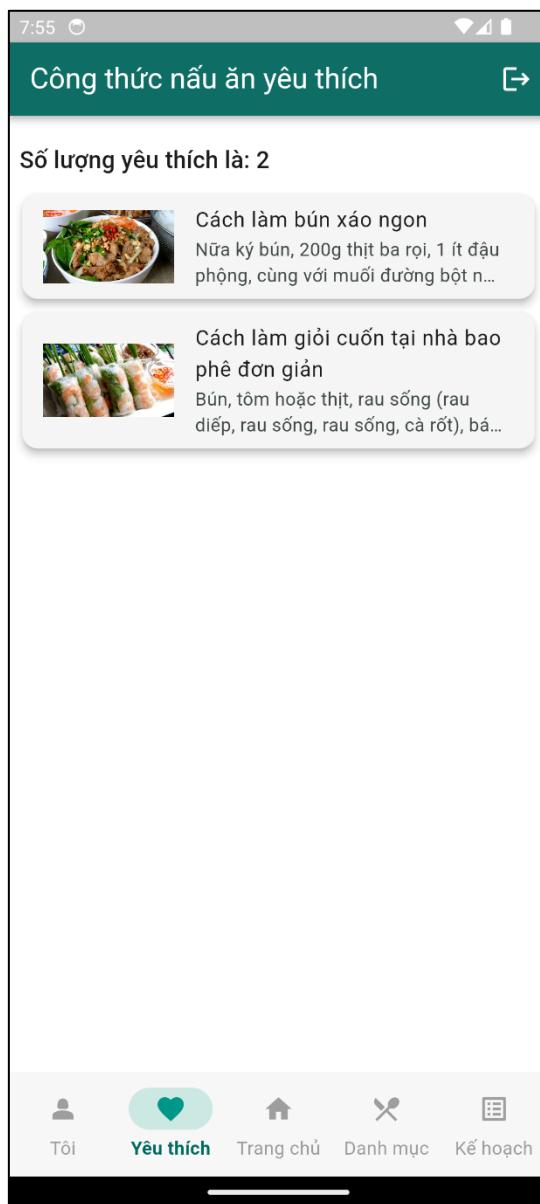


- **Chi tiết cài đặt chức năng xóa công thức nấu ăn:**

- Các widget được sử dụng cho chức năng này gồm: showDialog, AlertDialog, Icon, Text, TextStyle, Row, Expanded, TextButton.
- Chức năng chỉ sử dụng thư viện material để xây dựng hộp thoại chọn chức năng xóa và xác nhận.
- Chức năng này sử dụng provider để quản lý trạng thái chia sẻ. Cụ thể là dùng để truy cập đến phương thức xóa trong provider FoodRecipesManager để thực hiện thay đổi dữ liệu trên firebase.
- Chức năng này có thực hiện việc lưu trữ dữ liệu trên dịch vụ lưu trữ Firebase. Cụ thể là yêu cầu thực hiện hành động xóa trên firebase.

## **7. Chức năng/giao diện 7: Chức năng hiển thị công thức nấu ăn yêu thích**

- **Miêu tả chức năng/giao diện:** Chức năng này dùng để hiển thị tất cả các công thức nấu ăn yêu thích của người dùng, đồng thời cũng hiển thị bao nhiêu công thức được yêu thích.
- **Ảnh chức năng/giao diện:**



- **Chi tiết cài đặt chức năng hiển thị công thức nấu ăn yêu thích:**

- Các widget được sử dụng trong chức năng này gồm: AppBar, Text, IconButton, FutureBuilder, Center, CircularProgressIndicator,

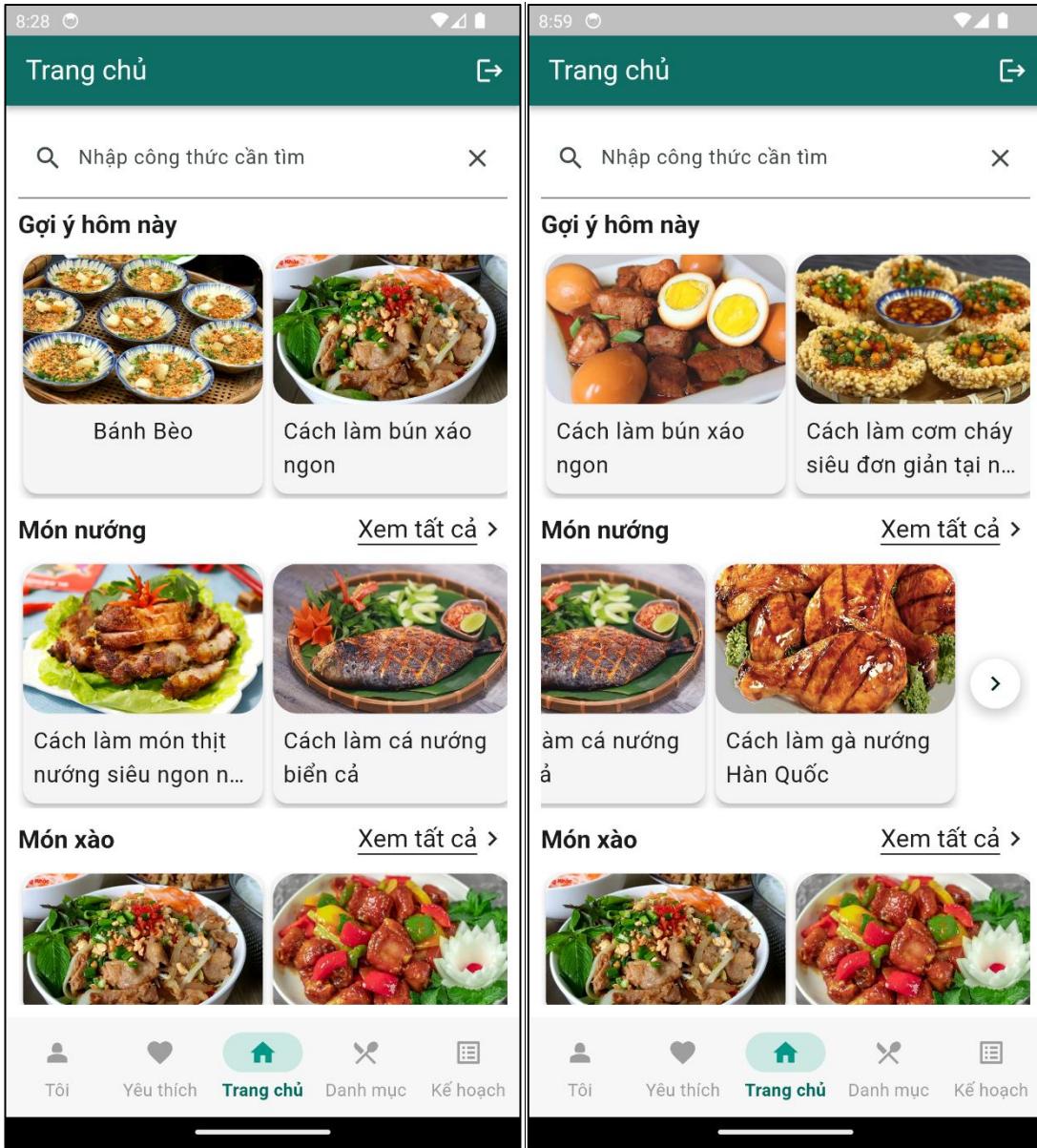
RefreshIndicator, Column, SizedBox, Consumer, Container, EdgeInsets.all, Expanded, UserFoodRecipesDetailMode(true).

- Trong chức năng này chỉ sử dụng các thư viện như material và provider để xây dựng giao diện và quản lý trạng thái chia sẻ.
- Chức năng này có sử dụng provider để quản lý trạng thái chia sẻ. Cụ thể là theo dõi sự thay đổi của favoriteItems được định nghĩa trong provider FoodRecipesManager. Khi người dùng yêu thích thêm một công thức thì favoriteItems sẽ thay đổi đồng thời cũng báo lại cho các widget lắng nghe cập nhật UI. Và trong chức năng này cũng sử dụng Consumer để lắng nghe thay đổi và cập nhật giao diện ngay lập tức.
- Chức năng có đọc dữ liệu từ dịch vụ lưu trữ của Firebase. Cấu trúc của một JSON khi đọc dữ liệu sẽ có dạng như sau:

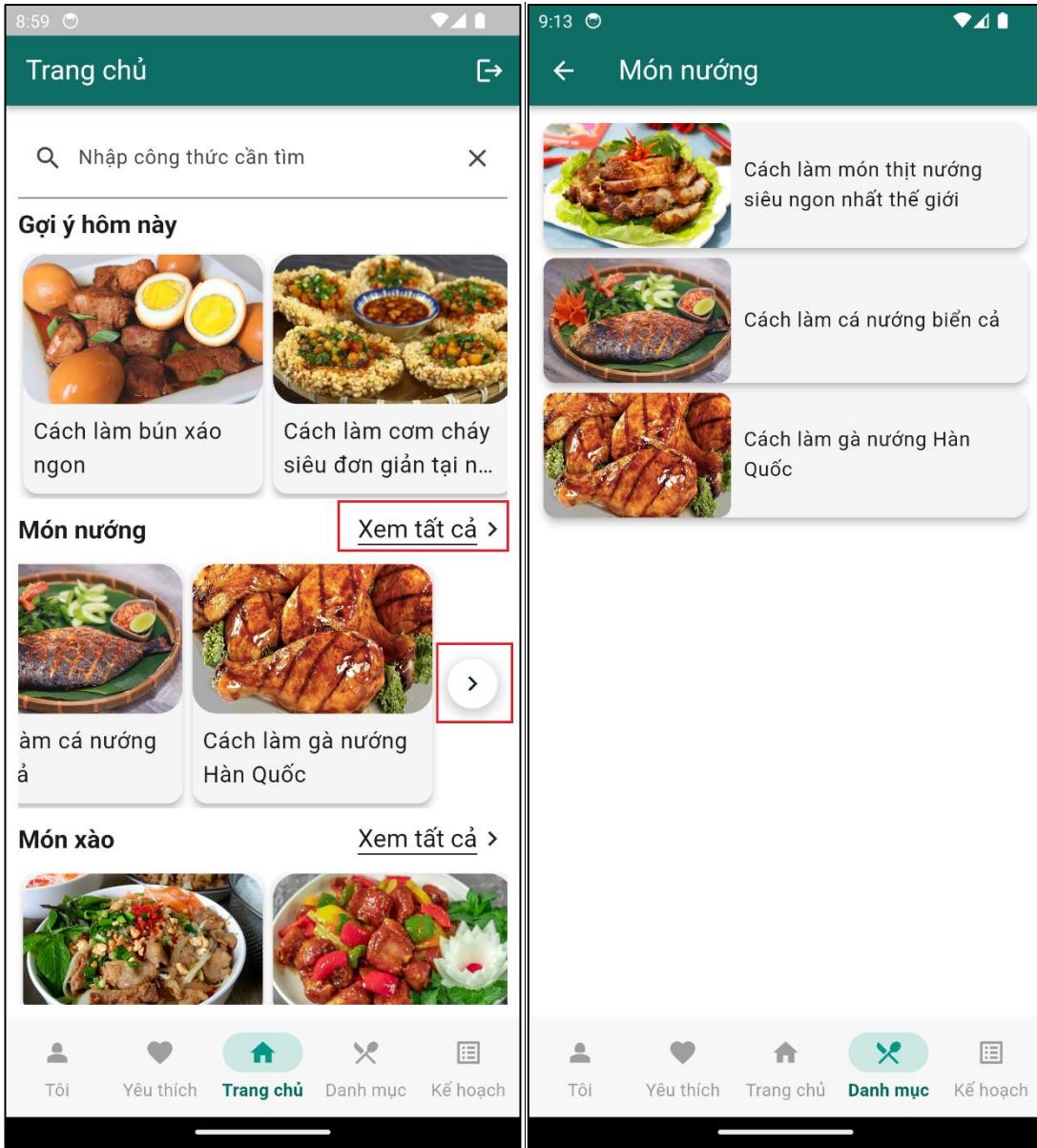
```
Id công thức 1 (String) {  
    imageUrl (String)  
    ingredient (String)  
    isPublic (String)  
    processing (String)  
    title (String)  
    type: (String)  
}
```

## 8. Chức năng/giao diện 8: Giao diện trang chủ

- **Miêu tả chức năng/giao diện:** Tại trang chủ sẽ hiển thị các 5 công thức nấu ăn ngẫu nhiên cho hôm nay. Đồng thời cũng sẽ hiển thị 5 công thức nấu ăn đại diện cho mỗi loại cách chế biến. Nếu người dùng muốn xem đầy đủ thì nhấp vào nút “Xem tất cả” hoặc biểu tượng mũi tên bên trong danh sách của mỗi loại cách chế biến. Sau khi nhấp xong thì sẽ được chuyển hướng đến trang hiển thị các công thức nấu ăn của loại cách chế biến cụ thể. Ngoài ra tại trang chủ khi nhấp vào một vào một sản phẩm cũng có thể xem chi tiết công thức và có thể tìm kiếm một công thức cụ thể.
- **Ảnh chức năng/giao diện:**



Nhấn vào nút “xem tất cả” hoặc biểu tượng “mũi tên tròn” thì sẽ chuyển hướng đến giao diện xem tất cả các công thức theo thể loại cách chế biến đã chọn.



- **Chi tiết cài đặt:**

- Các widget được sử dụng trong trang chủ gồm: Scaffold, AppBar, Text, IconButton, Center, Padding, EdgeInsets.all, Column, FoodSearchField, FutureBuilder, CircularProgressIndicator, RefreshIndicator, Consumer, ListView.builder, StaggeredGridView, FoodSearchScreen. Trong đó FoodSearchScreen gồm các widget như Center, Text, GridView.builder, EdgeInsets.only, StaggeredGridTile, SliverGridDelegateWithFixedCrossAxisCount. StaggeredGridTile lại gồm các widget như Container, Card, Stack, Column, SizedBox, ClipRRect, Image, NetworkImage, BorderRadius.circular, Padding, Text, Positioned.fill, Material, InkWell. Và StaggeredGridView gồm

BoxDecoration, BoxShadow, Column, Row, Padding, TextButton, Row, Container, EdgeInsets.only, TextStyle, Text, Icon, SizedBox, ListView.builder, IconButton, CircleAvatar và StaggeredGridTile.

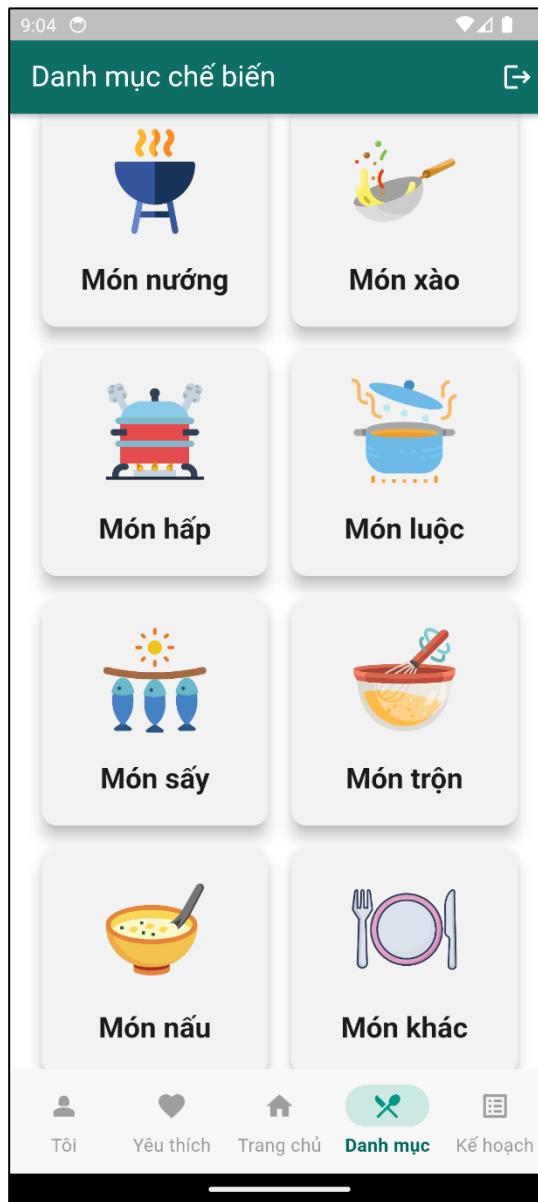
- Tại trang chủ có sử dụng một số thư viện như: flutter/material, go\_router, provider, dart:async.
- Vai trò của từng thư viện:
  - **Thư viện flutter/material:** thư viện chính của flutter, cung cấp các widget và công cụ để xây dựng giao diện người dùng theo Material Design.
  - **Thư viện provider/provider:** được sử dụng để triển khai mô hình quản lý trạng thái (state management) trong ứng dụng Flutter thông qua provider pattern. Cụ thể tại trang chủ này dùng để truy cập đến allItem được định nghĩa trong FoodRecipesManager để hiển thị các công thức nấu ăn
  - **Thư viện go\_router:** có vai trò cung cấp một cách tiện lợi và linh hoạt để quản lý các tuyến đường trong ứng dụng flutter. Trong ngữ cảnh trang chủ thì nó dùng để chuyển hướng đến trang hiển thị tất cả công thức nấu ăn theo loại cách chế biến hoặc để chuyển hướng đến chi tiết công thức của một công thức nấu ăn.
  - **Thư viện dart:async:** dùng để hỗ trợ cho lập trình bất đồng bộ với các lớp Future
- Tại giao diện trang chủ có sử dụng đến provider để hiển thị quản lý trạng thái ứng dụng. Cụ thể provider FoodRecipesManager được cấp sẵn ở nút gốc của cây và đã định nghĩa sẵn các danh sách cần thiết. Tại trang chủ này thì sẽ dùng Consumer để quan sát sự thay đổi của danh sách \_allItems. Mỗi khi có sự thay đổi trong danh sách thì giao diện sẽ cập nhật UI dựa vào Map foodByTypeList. foodByTypeList là một hàm được cung cấp trong provider FoodRecipesManager với nhiệm vụ là tạo một map gồm có key là tên loại cách chế biến, còn value là một danh sách đầy đủ của loại cách chế biến đó. Và giá trị trả về của danh sách này vẫn dựa vào \_allItems, nếu \_allItems bị thay đổi thì foodByTypeList cũng sẽ được cập nhật. Lúc này giao diện sẽ hiển thị dựa vào foodByTypeList và truyền tên loại cách chế biến và danh sách của loại cách chế biến đó vào widget StaggeredGridView để hiển thị cho người dùng. StaggeredGridView là một widget hiển thị theo chiều ngang để cho người dùng có thể lướt sang hai bên để xem.

## 9. Chức năng/giao diện 9: Giao diện danh mục chế biến

- **Miêu tả chức năng/giao diện:** Giao diện danh mục chế biến sẽ liệt kê ra tất cả các loại cách chế biến món ăn được phân theo từng danh mục riêng. Khi chọn một

danh mục cụ thể thì sẽ hiển thị ra tất cả các công thức chế biến theo loại danh mục đó.

- **Ảnh chức năng/giao diện:**



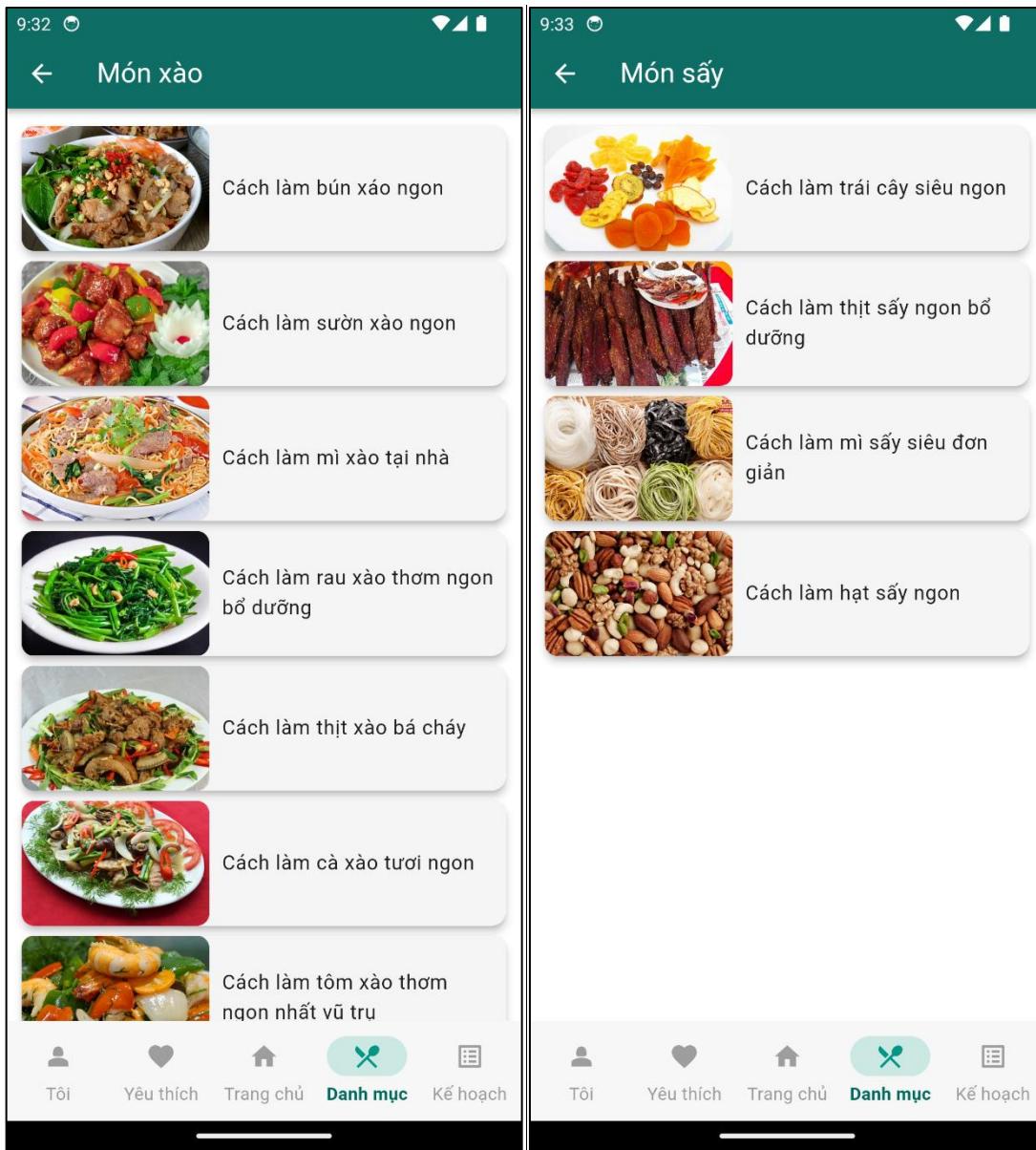
- **Chi tiết cài đặt giao diện danh mục chế biến:**

- Các widget được sử dụng trong giao diện này gồm: Scaffold, AppBar, Text, IconButton, FutureBuilder, Center, CircularProgressIndicator, GridView.count, EdgeInsets.all, Card, Stack, Column, Image.asset, Positioned.fill, Material, InkWell, BorderRadius.all.
- Giao diện này sử dụng một số thư viện như material, go\_router, và provider.
- Vai trò của từng thư viện:

- **Thư viện flutter/material:** thư viện chính của flutter, cung cấp các widget và công cụ để xây dựng giao diện người dùng theo Material Design.
  - **Thư viện go\_router:** có vai trò cung cấp một cách tiện lợi và linh hoạt để quản lý các tuyến đường trong ứng dụng flutter. Trong ngữ cảnh của trang này thì dùng để chuyển hướng đến route con của trang này là trang hiển thị tất cả công thức theo loại cách chế biến.
  - **Thư viện provider/provider:** được sử dụng để triển khai mô hình quản lý trạng thái (state management) trong ứng dụng Flutter thông qua provider pattern. Cụ thể là trong trang này dùng để nạp lại danh sách tất cả các sản phẩm một lần duy nhất trong FutureBuilder khi ứng dụng được chạy và cùng để gọi hàm lọc lại danh sách theo loại đã chọn.
- Trong giao diện này có sử dụng provider để nạp danh sách tất cả các công thức một lần duy nhất bên trong FutureBuilder. Cụ thể, mỗi khi ứng dụng được chạy và FoodRecipesManager cũng đã định nghĩa danh sách tất cả công thức và hàm nạp dữ liệu từ Firebase. Khi lần đầu truy xuất đến trang này thì FutureBuilder sẽ có nhiệm vụ nạp lại danh sách tất cả công thức một lần duy nhất. Hơn nữa tại mỗi một mục đều truy cập đến hàm setFoodType của provider với nhiệm vụ là lọc lại danh sách theo loại của mục đã chọn. Điều này giúp cho việc hiển thị trang xem tất cả công thức của một loại cụ thể được chính xác.
- Giao diện này có đọc dữ liệu từ dịch vụ lưu trữ của Firebase một lần duy nhất khi lần đầu tiên truy cập vào trang. Cấu trúc JSON sẽ có dạng như sau:
- ```
foodRecipes {
    Id công thức 1 (String) {
        creatorId (String)
        imageUrl (String)
        ingredient (String)
        isPublic (String)
        processing (String)
        title (String)
        type: (String)
    }
    .....
}
```

#### **10. Chức năng/giao diện 10: Giao diện hiển thị tất cả các công thức nấu ăn của một loại cách chế biến cụ thể**

- **Miêu tả chức năng/giao diện:** Giao diện này sẽ hiển thị tất cả các công thức nấu ăn của một loại cách chế biến khi người dùng chọn tại trang “Danh mục chế biến”.
- **Ảnh chức năng/giao diện:**



#### - Chi tiết cài đặt giao diện:

- Các widget được sử dụng cho giao diện này gồm: Scaffold, AppBar, Text, Padding, EdgeInsets.only, RefreshIndicator, Consumer, ListView.builder, HorizontalFoodRecipe. Trong đó widget HorizontalFoodRecipe gồm Padding, EdgeInsets.symmetric, Card, Stack, Row, EdgeInsets.only, ClipRRect, BorderRadius
- Chức năng có sử dụng một số thư viện như material, go\_router, provider.
- Vai trò của từng thư viện:
  - **Thư viện flutter/material:** thư viện chính của flutter, cung cấp các widget và công cụ để xây dựng giao diện người dùng theo Material Design.

- **Thư viện provider/provider:** được sử dụng để triển khai mô hình quản lý trạng thái (state management) trong ứng dụng Flutter thông qua provider pattern. Cụ thể trong ứng dụng sử dụng provider để quan sát danh sách itemsByType và hiển thị chính xác công thức theo thể loại.
- Giao diện này có sử dụng provider để quản lý trạng thái chia sẻ. Cụ thể là trong provider FoodRecipesManager đã định nghĩa một hàm getter itemsByType có nhiệm vụ trả về danh sách theo loại cách chế biến. Và danh sách này được tạo ra bằng cách theo dõi biến \_foodType được định nghĩa trong FoodRecipes. Khi người dùng đang ở trang danh mục chế biến và chọn một mục cụ thể. Khi chọn thì nó sẽ gọi hàm setFoodType để đặt lại giá trị cho \_foodType và báo lại cho các widget lắng nghe biết sự thay đổi. Khi đó itemsByType sẽ lọc danh sách dựa trên \_foodType đã nhận được và trả về cho giao diện “Xem tất cả sản phẩm của một loại cách chế biến”. Trong trang này có sử dụng consumer để quan sát itemsByType nên nếu người dùng chọn danh mục khác thì giao diện cũng sẽ cập nhật hợp lý.

➤ Chức năng này có đọc dữ liệu từ dịch vụ lưu trữ của Firebase. Cấu trúc JSON có dạng như sau:

```
foodRecipes {
    Id công thức 1 (String) {
        creatorId (String)
        imageUrl (String)
        ingredient (String)
        isPublic (String)
        processing (String)
        title (String)
        type: (String)
    }
    .....
}
```

## 11. Chức năng/giao diện 11: Giao diện hiển thị chi tiết công thức nấu ăn

- **Miêu tả chức năng/giao diện:** Dùng để hiển thị các thông tin chi tiết của một công thức cụ thể. Người dùng có thể truy cập đến trang chi tiết công thức nấu ăn ở bất kỳ đâu trong các trang “Công thức nấu ăn của tôi”, “Công thức nấu ăn yêu thích”, “Trang chủ”, “Trang tất cả công thức của một loại cách chế biến”. Tại các trang đó đều có hiển thị cách công thức và khi người dùng nhấn vào thì sẽ chuyển hướng đến trang hiển thị chi tiết công thức.
- **Ảnh chức năng/giao diện:**
  - Xem chi tiết công thức có thể tại trang “Công thức nấu ăn của tôi”:

Công thức nấu ăn của tôi

Nhập công thức cần tìm

Chế độ hiển thị

- Món ăn siêu vĩ đại  
asdfasdf
- Banh bao hap xuyen luc dia  
abc yzlkajslfkjals
- Cách làm rau cải luộc đinh của chót**  
Rau cải (cải bắp, cải thìa, cải ngọt, ...).
- Bún rêu đinh nhất quả đất  
Bún, cà chua, và các gia vị khác
- Cách làm gạo luộc ngon  
Gạo, nước.
- Cach lam keo bong gon ngon  
Duong, bong gon
- Song gio

+ Tối Yêu thích Trang chủ Danh mục Kế hoạch

← Chi tiết công thức nấu ăn

**CÁCH LÀM RAU CẢI LUỘC ĐỈNH CỦA CHÓT** ❤️

Loại chế biến: món hấp

Nguyên liệu:  
Rau cải (cải bắp, cải thìa, cải ngọt, ...).

Cách chế biến:  
Rửa sạch rau, đặt vào nồi, đổ nước và một ít

Tối Yêu thích Trang chủ Danh mục Kế hoạch

➤ Xem chi tiết công thức có thể tại trang “Công thức nấu ăn yêu thích”:

Công thức nấu ăn yêu thích

Số lượng yêu thích là: 2

 Cách làm bún xáo ngon  
Nữa kỹ bún, 200g thịt ba rọi, 1 ít đậu phộng, cùng với muối đường bột n...

 Cách làm giòi cuốn tại nhà bao phê đơn giản  
Bún, tôm hoặc thịt, rau sống (rau diếp, rau sống, rau sống, cà rốt), bá...

← Chi tiết công thức nấu ăn



**CÁCH LÀM GIÒI CUỐN TẠI NHÀ BAO PHÊ ĐƠN GIẢN** ❤️

Loại chế biến: khác

Nguyên liệu:  
Bún, tôm hoặc thịt, rau sống (rau diếp, rau sống, rau sống, cà rốt), bánh tráng.

Cách chế biến:

Tôi  Yêu thích Trang chủ Danh mục Kế hoạch

Tôi  Yêu thích Trang chủ Danh mục Kế hoạch

➤ Xem chi tiết công thức có thể tại “Trang chủ”:

Trang chủ

Nhập công thức cần tìm

Gợi ý hôm nay

Cách làm canh chua miền tây

Món nướng

Xem tất cả >

Cách làm món thịt nướng siêu ngon n...

Cách làm cá nướng biển cá

Món xào

Xem tất cả >

Tôi Yêu thích Trang chủ Danh mục Kế hoạch

← Chi tiết công thức nấu ăn

**CÁCH LÀM CANH CHUA MIỀN TÂY**

Loại chế biến: món nấu

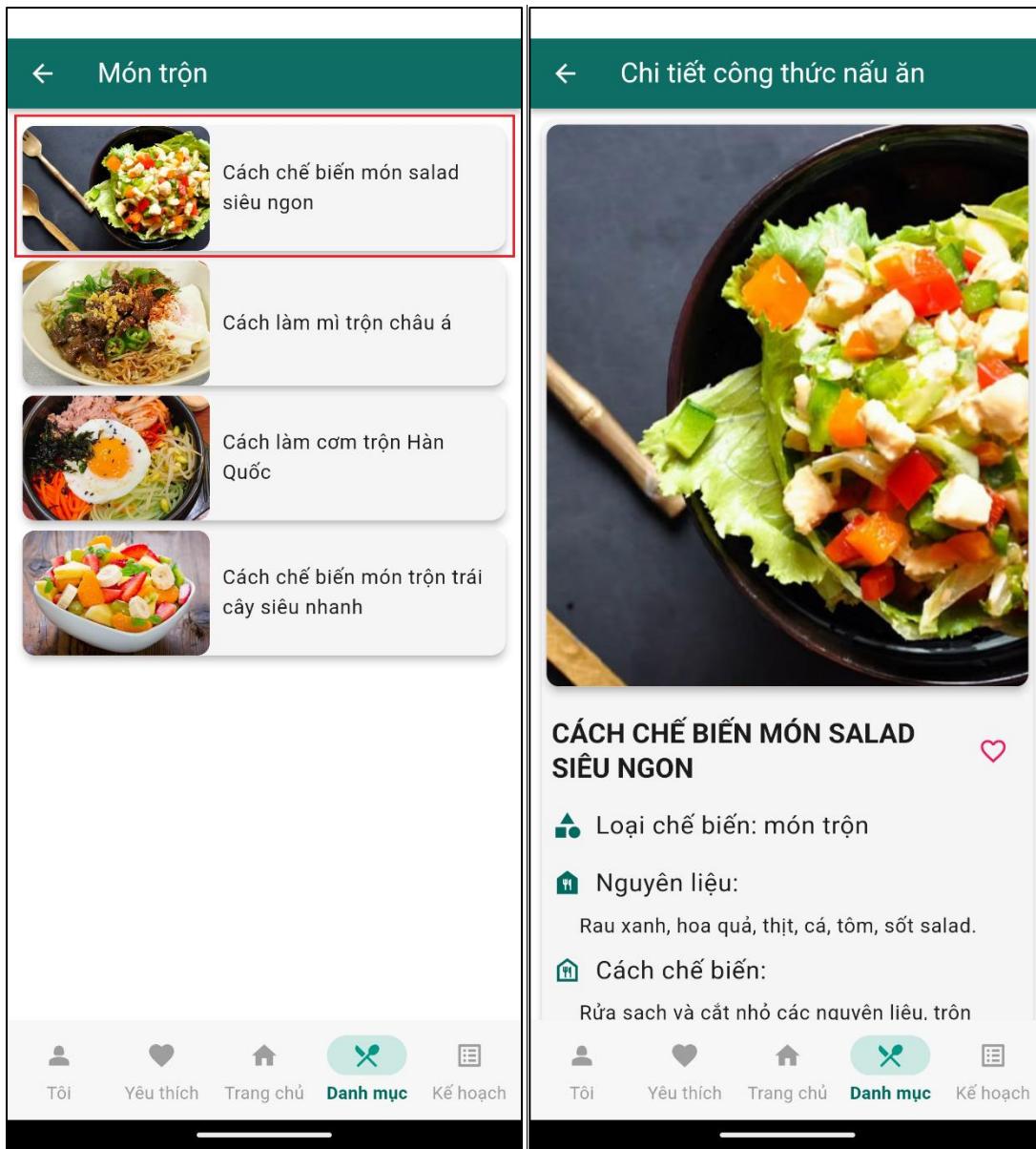
Nguyên liệu:

Cá, tôm hoặc thịt gà, cà chua, nước mắm, đường, dưa chuột, rau giá.

Cách chế biến:

Tôi Yêu thích Trang chủ Danh mục Kế hoạch

- Xem chi tiết công thức có thể tại trang “Hiển thị công thức theo loại cách chế biến”:

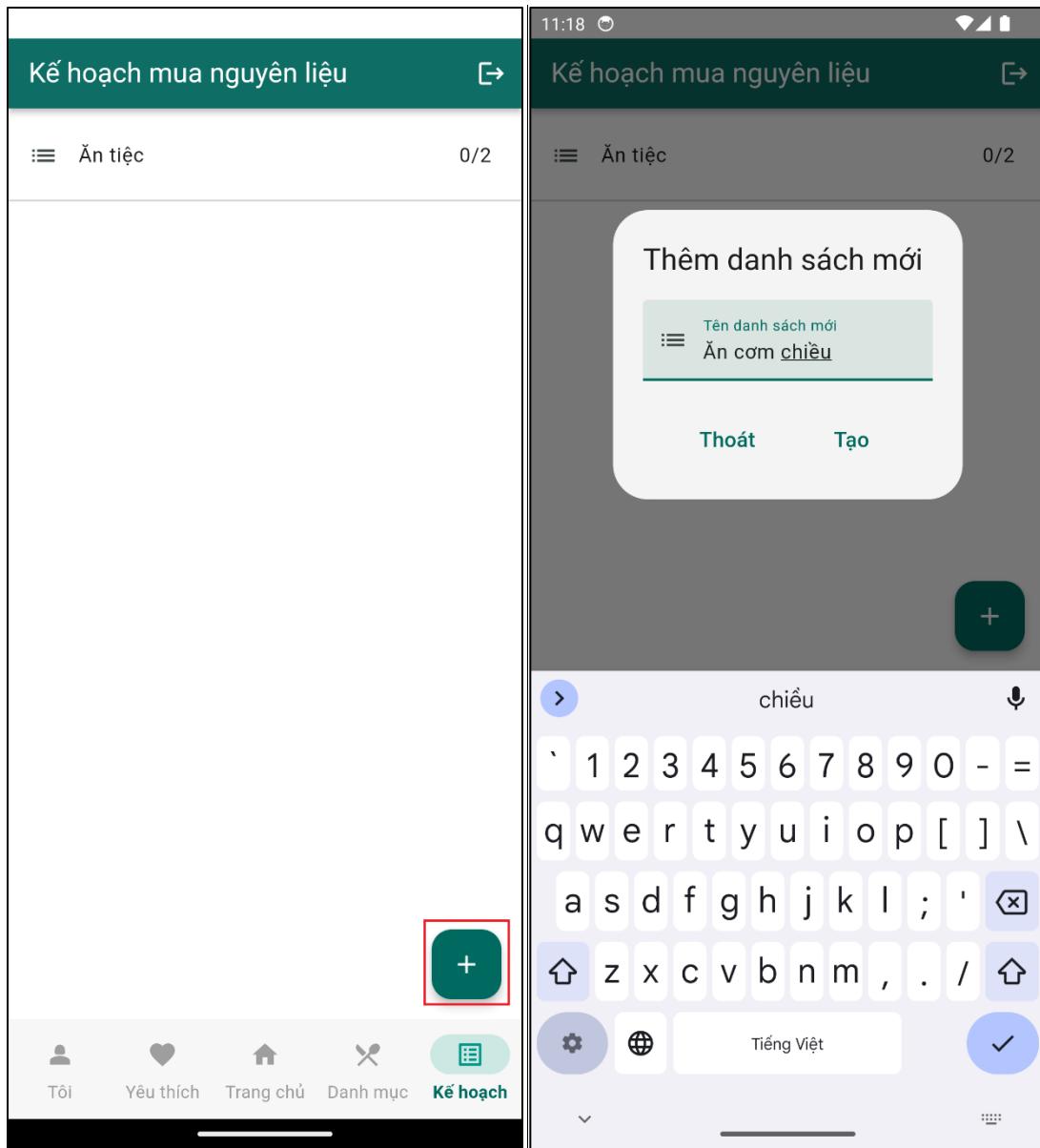


- **Chi tiết cài đặt giao diện chi tiết công thức nấu ăn:**
  - Các widget được sử dụng gồm có Scaffold, AppBar, Text, SingleChildScrollView, EdgeInsets.all, Card, ConstrainedBox, BoxConstraints, Column, Flexible, Card, ClipRRect, Image.network, BorderRadius.circular, SizedBox, Padding, Row, Expanded, ValueListenableBuilder, IconButton, Container, Icon, Text, EdgeInsets.only
  - Các thư viện được sử dụng gồm material và provider.
  - Vai trò của từng thư viện:
    - **Thư viện flutter/material:** thư viện chính của flutter, cung cấp các widget và công cụ để xây dựng giao diện người dùng theo Material Design.

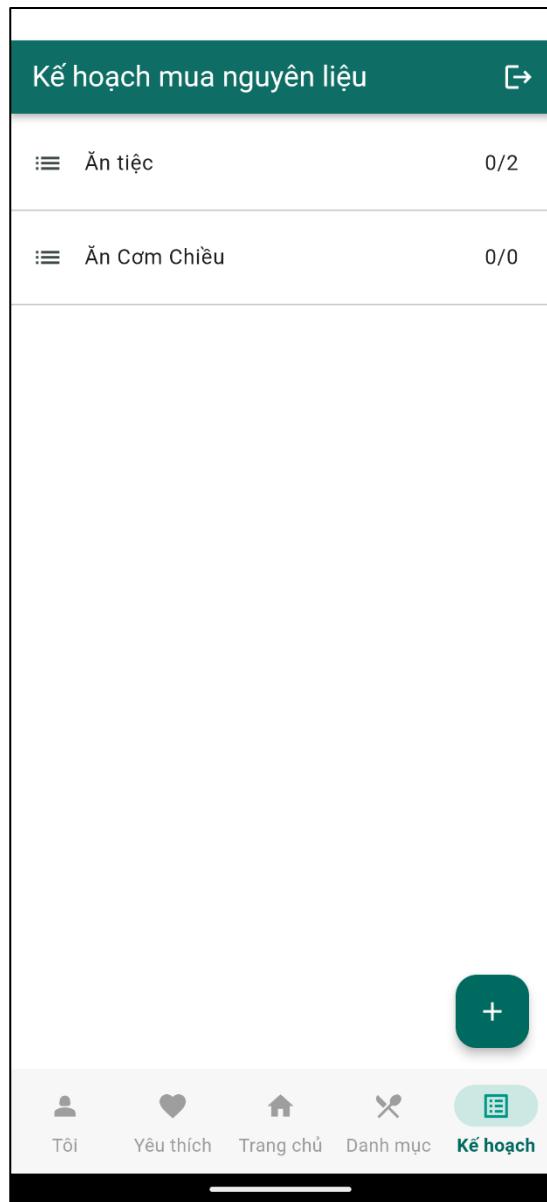
- **Thư viện provider/provider:** được sử dụng để triển khai mô hình quản lý trạng thái (state management) trong ứng dụng Flutter thông qua provider pattern. Cụ thể tại trang chi tiết công thức nấu ăn để truy xuất hàm đổi trạng thái công thức yêu thích được định nghĩa trong provider FoodRecipesManager.
- Trong trang này thì có sử dụng provider và ValueNotifier để quản lý trạng thái chia sẻ. Đối với ValueNotifier thì thuộc tính isFavorite của lớp FoodRecipe sẽ là một ValueNotifier với giá trị mặc định là false, có nghĩa là công thức chưa được thêm yêu thích. Trong trang chi tiết công thức nấu ăn có sử dụng ValueListenableBuilder để quan sát trạng thái của một foodRecipe và quyết định vẽ icon yêu thích. Nếu thuộc tính isFavorite có giá trị là true thì sẽ hiển thị icon hình trái tim được lấp đầy, còn nếu là false thì hiển thị icon trái tim rỗng. Điều đó có nghĩa là tại trang chi tiết này nếu người dùng nhấp vào và thay đổi trạng thái giao diện sẽ được cập nhật. Đồng thời lúc đó hàm toggleFavoriteFoodRecipe của provider FoodRecipesManager được gọi để thay đổi trạng thái trên firebase.
- Chức năng có thực hiện việc lưu trữ dữ liệu vào dịch vụ lưu trữ của Firebase bằng cách thay đổi giá trị của isFavorite trên firebase. Cấu trúc JSON được gửi lên có dạng:
- ```
favoriteFoodRecipe {
  UserId1 {
    Id sản phẩm 1: true
    .....
  }
  .....
}
```

## 12. Chức năng/giao diện 12: Giao diện kế hoạch mua nguyên liệu

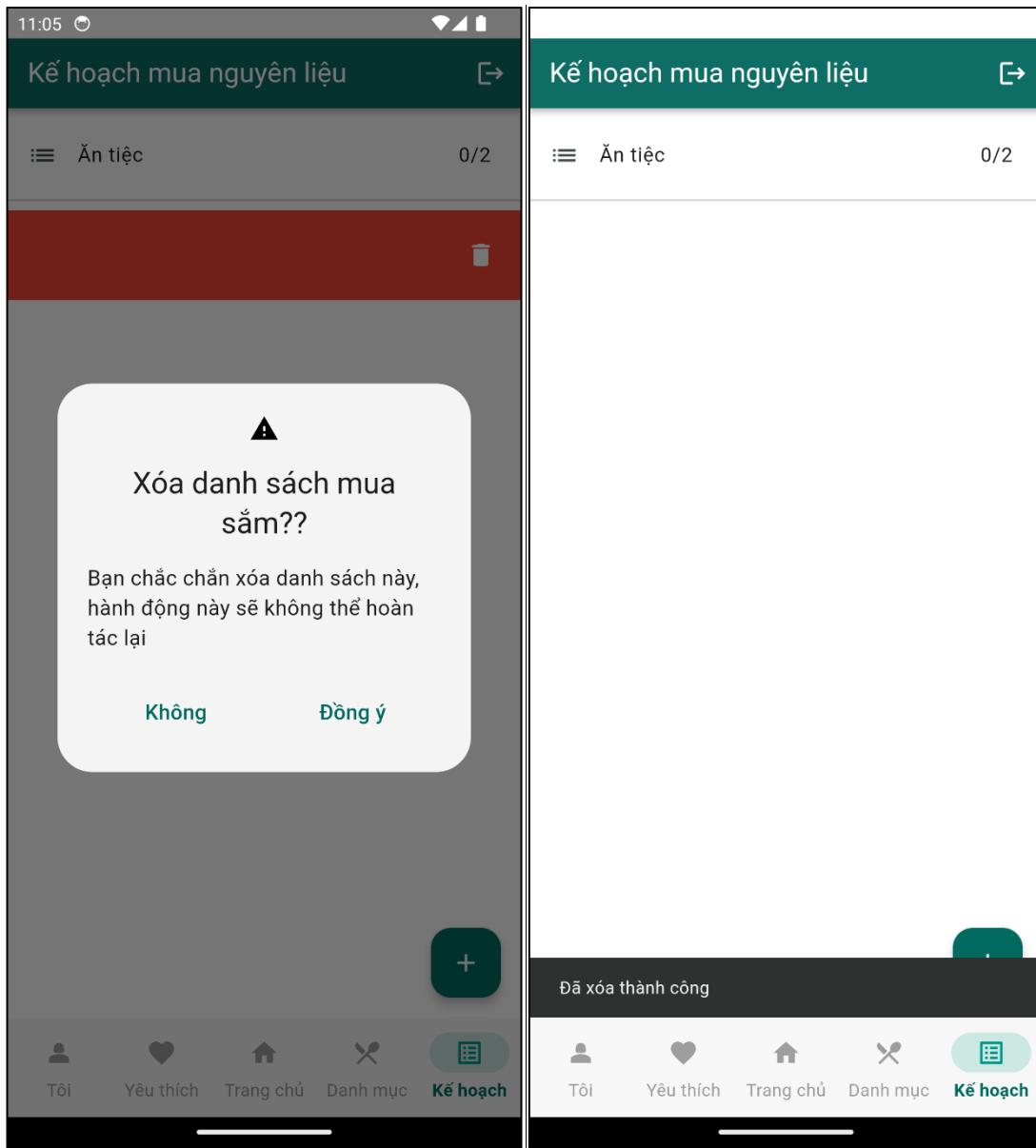
- **Miêu tả chức năng/giao diện:** Chức năng này có nhiệm vụ hỗ trợ cho người dùng ghi chép lại những nguyên liệu cần phải mua trong bữa ăn. Người dùng có thể tạo nhiều danh sách khác nhau, đồng thời cũng có thể xóa bỏ một danh sách cụ thể.
- **Ảnh chức năng/giao diện:**
  - Thêm một danh sách mới:



➤ Hiển thị danh sách các danh sách mua nguyên liệu:



- Xóa một danh sách cũ bằng cách vuốt sang trái:



- **Chi tiết cài đặt giao diện:**

- Các widget được sử dụng trong giao diện này gồm Scaffold, AppBar, Text, IconButton, Icon, Padding, EdgeInsets.only, FutureBuilder, Center, CircularProgressIndicator, RefreshIndicator, Consumer, ListView.builder, Dismissible, ScaffoldMessenger, Container, Column, InkWell, FloatingActionButton, AlertDialog, showDialog, Row, TextButton, TextFormField, InputDecoration, UnderlineInputBorder, FoodShoppingListTile. Trong đó FoodShoppingListTile gồm ListTile, Text, Icon.
- Các thư viện được sử dụng trong giao diện này gồm: material, provider, go\_router.

- Vai trò của từng thư viện:
  - **Thư viện flutter/material:** thư viện chính của flutter, cung cấp các widget và công cụ để xây dựng giao diện người dùng theo Material Design.
  - **Thư viện provider/provider:** được sử dụng để triển khai mô hình quản lý trạng thái (state management) trong ứng dụng Flutter thông qua provider pattern. Cụ thể trong trang này thì provider được dùng để nạp danh sách “danh sách kế hoạch mua nguyên liệu” của người dùng và gọi các hàm thao tác thêm và xóa
  - **Thư viện go\_router:** có vai trò cung cấp một cách tiện lợi và linh hoạt để quản lý các tuyến đường trong ứng dụng flutter. Trong ngữ cảnh của ứng dụng việc dùng go\_router để chuyển hướng đến xem danh sách các nguyên liệu cần mua của một danh sách cụ thể.
- Chức năng có sử dụng provider để quản lý trạng thái. Cụ thể đối với chức năng hiển thị danh sách các danh sách mua nguyên liệu thì provider ShoppingListManager được định nghĩa ở nút gốc để có thể truy xuất ở bất kỳ đâu. Trong lớp này cũng định nghĩa một danh sách \_items để chứa danh sách các danh sách, cùng với các phương thức như fetchShoppingList, addShoppingList, và removeShoppingList. Đối với việc hiển thị thì trong giao diện hiển thị danh sách thì có sử dụng một FutureBuilder để nạp danh sách khi lần đầu tiên vào trang và sử dụng một Consumer để quan sát sự thay đổi của danh sách \_items. Nếu có bất kỳ thay đổi gì thì sẽ cập nhật UI ngay lập tức. Đối với chức năng thêm danh sách mới thì lúc này sau khi đã đặt tên và nhấn tạo thì nó sẽ truy cập đến hàm addShoppingList để thêm một danh sách mới vào \_items và Firebase. Khi muốn xóa một danh sách thì người dùng vuốt danh sách đó sang trái, lúc này một hộp thoại xuất hiện và yêu cầu xác nhận. Nếu người dùng xác nhận đồng ý thì hàm removeShoppingList của ShoppingListManager được gọi để xóa một danh sách trong \_items và Firebase và cập nhật giao diện tương ứng.
- Chức năng này có đọc và lưu trữ dữ liệu lên dịch vụ lưu trữ của Firebase. Cấu trúc JSON của một dữ liệu được gửi lên có dạng:

```

Shopping {
  UserId1 {
    ShoppingId1 {
      creatorId (String)
      name (String)
    }
  }
}

```

}

.....

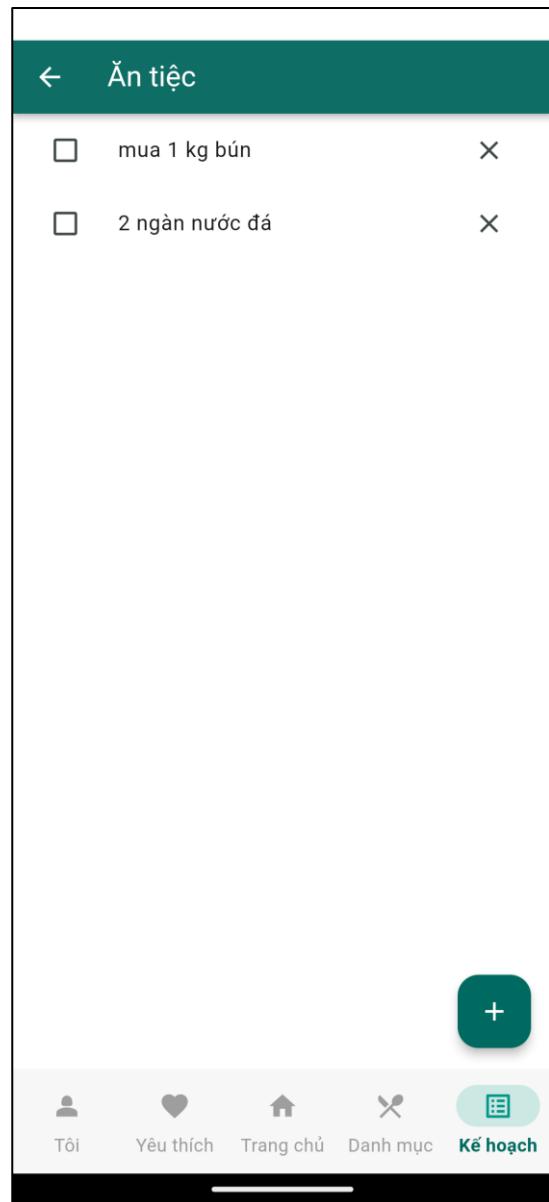
}

.....

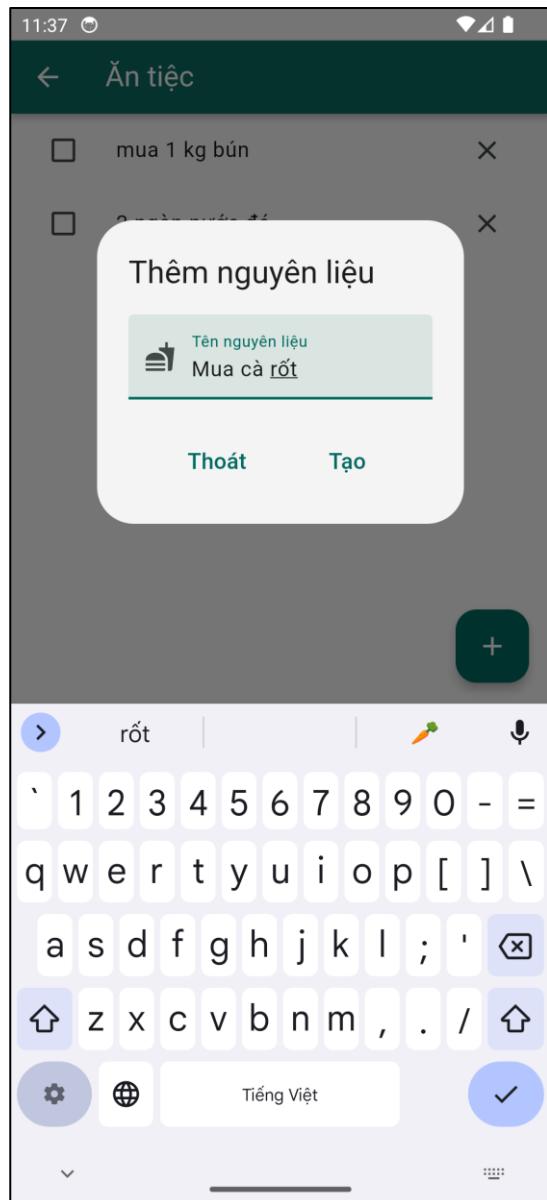
}

### **13. Chức năng/giao diện 13: Giao diện chi tiết danh sách mua nguyên liệu**

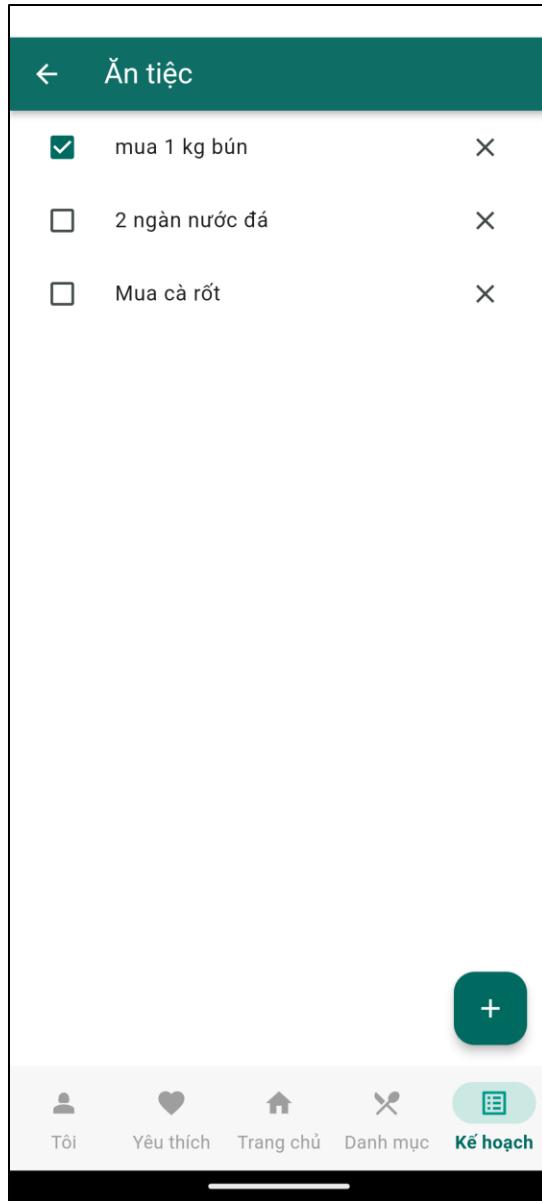
- **Miêu tả chức năng/giao diện:** Chức năng này dùng để hiển thị những nguyên liệu cần mua của một danh sách cụ thể. Người dùng có thể thêm mới nguyên liệu, có thể đánh dấu hoàn thành hoặc xóa một nguyên liệu nào đó.
- **Ảnh chức năng/giao diện:**
  - Hiển thị chi tiết danh sách:



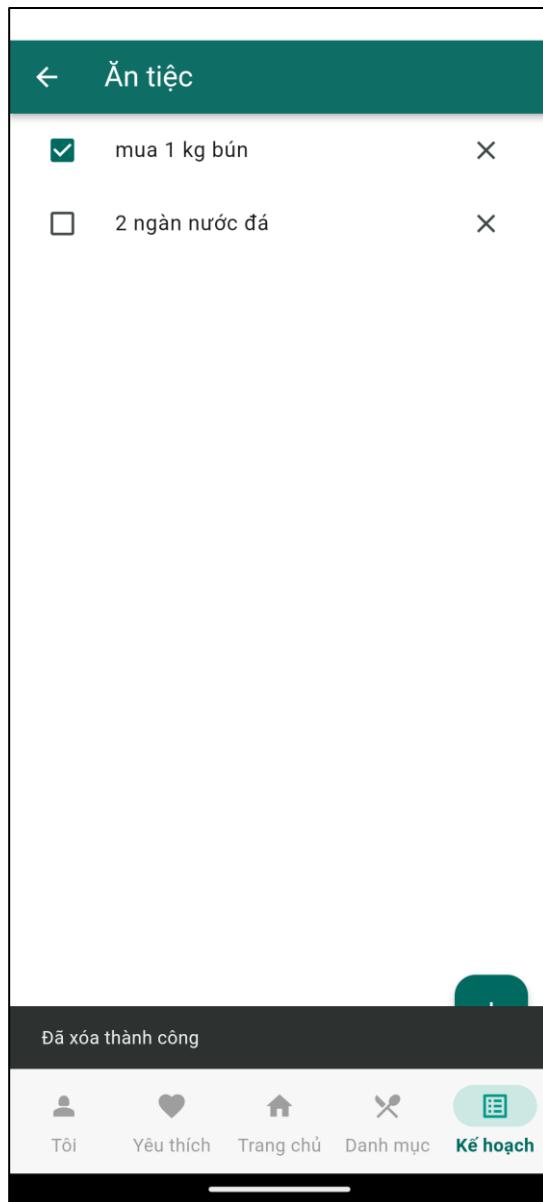
➤ Thêm nguyên liệu cho danh sách:



➤ Đánh dấu hoàn thành đối với một nguyên liệu:



➤ Xóa bỏ một nguyên liệu trong danh sách:



#### - Chi tiết cài đặt:

- Các widget được sử dụng trong giao diện này gồm: Scaffold, AppBar, Text, Center, ListView.builder, ShoppingItemListTile, FloatingActionButton, Icon, showDialog, AlertDialog, Row, TextButton, TextFormField, InputDecoration, UnderlineInputBorder. Trong đó ShoppingItemListTile gồm ListTile, Text, ValueListenableBuilder, Checkbox, IconButton, Icon, ScaffoldMessenger.
- Chức năng này sử dụng một số thư viện gồm: material và provider
- Vai trò của các thư viện:

- **Thư viện flutter/material:** thư viện chính của flutter, cung cấp các widget và công cụ để xây dựng giao diện người dùng theo Material Design.
  - **Thư viện provider/provider:** được sử dụng để triển khai mô hình quản lý trạng thái (state management) trong ứng dụng Flutter thông qua provider pattern. Cụ thể trong ứng dụng có truy xuất đến provider ShoppingListManager để nạp danh sách chi tiết của một danh sách mua nguyên liệu, thực hiện các hành động thêm, đánh dấu và xóa nguyên liệu
- Chức năng này có sử dụng provider và ValueNotifier để quản lý trạng thái chia sẻ. Đối với ValueNotifer thì mỗi một mục trong danh sách mua nguyên liệu sẽ là một ShoppingItem và trong đây có thuộc tính isCheck là ValueNotifier với giá trị ban đầu là false, có nghĩa là chưa đánh dấu. Trong giao diện thì sẽ sử dụng một ValueListenableBuilder để quan sát giá trị của isCheck để quyết định đánh dấu vào checkbox hay không. Khi người dùng tick vào checkbox thì giá trị isCheck thay đổi và widget lắng nghe sẽ tự động xây dựng lại giao diện tương ứng với hành động. Đối với provider thì lớp ShoppingListManager đã định nghĩa thêm một số hàm như addAnShoppingListItem với nhiệm vụ thêm một nguyên liệu mới vào danh sách hiện tại và Firebase và hàm removeAnShoppingItem để xóa một mục trong một danh sách và firebase. Khi người dùng thực hiện thêm hoặc xóa thì giao diện sẽ được tự động cập nhật thông qua context.watch<ShoppingListManager>().items đã được định nghĩa trong giao diện.
- Chức năng có đọc và lưu trữ dữ liệu lên dịch vụ lưu trữ của Firebase. Cấu trúc JSON của dữ liệu gửi lên có dạng như sau:
- ```

shoppingItem {
    userId1 {
        shoppingId1 {
            itemId1 {
                name (String)
                isCheck (String)
            }
            .....
        }
        .....
    }
    .....
}

```

