**TRƯỜNG ĐẠI HỌC TÀI NGUYÊN VÀ MÔI TRƯỜNG TP.HCM**

**KHOA: HỆ THỐNG THÔNG TIN VÀ VIỄN THÁM**

****

**BÁO CÁO ĐỒ ÁN MÔN HỌC:**

**CÔNG NGHỆ LẬP TRÌNH ĐA NỀN TẢNG CHO ỨNG DỤNG DI ĐỘNG**

**LẬP TRÌNH ỨNG DỤNG “TÊN TIẾNG ANH CỦA BẠN LÀ GÌ ?”**

|  |  |
| --- | --- |
| Giảng viên hướng dẫn: | **ThS.Nguyễn Thanh Truyền** |
| Nhóm thực hiện: | **01** |
| Lớp: | **08\_ĐH\_TTMT** |
| Khoá: | **2019 - 2023** |

***TP.Hồ Chí Minh, tháng 08 năm 2023***

**TRƯỜNG ĐẠI HỌC TÀI NGUYÊN VÀ MÔI TRƯỜNG TP.HCM**

**KHOA: HỆ THỐNG THÔNG TIN VÀ VIỄN THÁM**

****

**BÁO CÁO ĐỒ ÁN MÔN HỌC:**

**CÔNG NGHỆ LẬP TRÌNH ĐA NỀN TẢNG CHO ỨNG DỤNG DI ĐỘNG**

**LẬP TRÌNH ỨNG DỤNG “TÊN TIẾNG ANH CỦA BẠN LÀ GÌ ?”**

|  |  |
| --- | --- |
| Giảng viên hướng dẫn: | **ThS.Nguyễn Thanh Truyền** |
| Nhóm thực hiện: | **01** |
| Lớp: | **08\_ĐH\_TTMT** |
| Khoá: | **2019 - 2023** |

***TP.Hồ Chí Minh, tháng 08 năm 2023***

**LỜI MỞ ĐẦU**

* **Lý do chọn đề tài**

Chủ đề Xây dựng giao diện UI ứng dụng: “Tên tiếng Anh của bạn là gì ?” kết hợp hai công nghệ phát triển mạnh mẽ là Flutter và Dart để tạo ứng dụng thú vị và linh hoạt.

Lý do đầu tiên để chọn chủ đề này là Flutter là một framework hiện đại và mạnh mẽ để phát triển ứng dụng di động. Flutter cho phép phát triển ứng dụng đồng thời trên nền tảng Android và iOS bằng một mã nguồn duy nhất. Nó cung cấp nhiều tiện ích linh hoạt tích hợp giúp dễ dàng tạo giao diện người dùng đẹp và trực quan. Flutter cho phép các nhà phát triển tạo ra trải nghiệm người dùng tuyệt vời với các tính năng và tiện ích hiện đại. Lý do thứ hai là Dart, ngôn ngữ lập trình chính được sử dụng trong Flutter. Dart là một ngôn ngữ linh hoạt, dễ đọc và hiệu quả. Nó hỗ trợ các chức năng hướng đối tượng và có thể được biên dịch nhanh chóng thành mã máy. Dart cũng hỗ trợ tải lại nóng trong quá trình phát triển, cho phép các nhà phát triển xem kết quả ngay lập tức khi họ thực hiện các thay đổi đối với mã nguồn của mình.

Lý do thứ ba là sử dụng Android Studio là môi trường lý tưởng để phát triển ứng dụng Flutter. Android Studio là một IDE mạnh mẽ được phát triển bởi Google dành riêng cho việc phát triển ứng dụng Android. Nó cung cấp các công cụ mạnh mẽ để lập trình và gỡ lỗi các ứng dụng. Kết hợp Flutter và Android Studio tạo ra môi trường phát triển hoàn hảo để tạo các ứng dụng chất lượng cao.

Tóm lại, chủ đề Xây dựng giao diện UI ứng dụng: “Tên tiếng Anh của bạn là gì ?” là một chủ đề thú vị và sáng tạo. Không chỉ là một ứng dụng vui nhộn mà còn có một chút tính năng giáo dục giúp người dùng hiểu rõ hơn về ý nghĩa và nguồn gốc tên tiếng Anh của mình. Giao diện đơn giản, thân thiện với người dùng và hấp dẫn cho phép bạn tương tác và học hỏi một cách dễ dàng và thú vị.

* **Mục tiêu nghiên cứu**
* Mục tiêu chung: Xây dựng giao diện UI ứng dụng: “Tên tiếng Anh của bạn

là gì ?”.

* Mục tiêu cụ thể:
* Lý thuyết: tìm hiểu về các widget: Container, Textfeild, Button, Center,

Media Query, ListView, TableData, Grid

* Thực hành: sử dụng widget, event của button xây dựng giao diện UI “Tên

tiếng Anh của bạn là gì ?” và xử lý code logic chuyển tên tiếng việt sang

tiếng anh.

* **Cấu trúc đề tài**
* Chương 1. Cơ sở lý luận.
* Chương 2. Phương pháp thực hiện.
* Chương 3. Kết quả đạt được.
* Chương 4. Kết luận và hướng phát triển.

**LỜI CẢM ƠN**

Trước hết em xin trân trọng cảm ơn thầy ThS.Nguyễn Thanh Truyền đã tận tình giúp đỡ, định hướng cách tư duy và cách làm việc khoa học. Đó là những đóng góp hết sức quý báu không chỉ trong quá trình thực hiện đồ án môn học mà còn là hành trang tiếp bước cho nhóm em trong quá trình học tập sau này.

Nhóm em xin chân thành cảm ơn bạn bè trong lớp đã tạo điều kiện tốt nhất để nhóm em có thể học hỏi và giúp đỡ nhóm em trong quá trình thực hiện đồ án môn học.

Trong qua trình thực hiện đồ án môn học, nhóm em đã cố gắng và nỗ lực thực hiện đồ án một cách hoàn chỉnh nhất, tuy nhiên vẫn không thể tránh khỏi những thiếu sót. Vì vậy, nhóm em rất mong nhận được sự đóng góp, chỉ bảo và giúp đỡ của thầy và các bạn để nhóm em nâng cao kiến thức về chuyên môn cũng như khả năng xây dựng đề tài cho đồ án môn học.

**LỊCH LÀM VIỆC**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tuần | Nội dung  công việc được giao | Tự nhận xét về mức độ hoàn thành | Nhận xét của  CB hướng dẫn | Chữ ký của  CB hướng dẫn |
| **1**  Từ ngày  đến ngày |  |  |  |  |
| **2**  Từ ngày  đến ngày |  |  |  |  |
| **3**  Từ ngày  đến ngày |  |  |  |  |
| **4**  Từ ngày  đến ngày |  |  |  |  |
| **5**  Từ ngày  đến ngày |  |  |  |  |
| **6**  Từ ngày  đến ngày |  |  |  |  |
| **7**  Từ ngày  đến ngày |  |  |  |  |
| **8**  Từ ngày  đến ngày |  |  |  |  |

TP. Hồ Chí Minh, tháng 8 năm 2023

**TÓM TẮT**

Đề tài: “Xây dựng giao diện UI ứng dụng: Tên tiếng Anh của bạn là gì ?” được thực hiện từ 13/07/2023 đến 10/08/2023.

Mục tiêu của đề tài là xây dựng thành công giao diện UI trên Android studio bằng Flutter và Dart.

Để đạt được những kết quả cụ thể như sau: xây dựng giao diện người dùng với các giao diện: Giao diện chính của ứng dụng, Giao diện dịch tên từ Tiếng Anh sang Tiếng Việt, Giao diện dịch bằng giọng nói, Giao diện dịch bằng máy ảnh.

**MỤC LỤC**

[**CHƯƠNG 1: MỞ ĐẦU** 1](#_Toc142328183)

[**1.1.Tổng quan về ngôn ngữ Dart** 1](#_Toc142328184)

[**1.1.1.Kiểu dữ liệu** 1](#_Toc142328185)

[**1.1.2.Các biến và các hàm** 4](#_Toc142328186)

[**1.1.3.Toán tử (Operators)** 5](#_Toc142328187)

[**1.1.4.Vòng lặp** 8](#_Toc142328188)

[**1.2.Tổng quan về Flutter** 8](#_Toc142328189)

[**1.3.Các tính năng của Flutter** 10](#_Toc142328190)

[**1.4.Tổng quan kiến trúc của Flutter** 12](#_Toc142328191)

[**1.4.1.Kiến trúc của Flutter** 12](#_Toc142328192)

[**1.4.2.Flutter Engine** 12](#_Toc142328193)

[**1.4.3.Thư viện nền tảng (Foundation Library)** 12](#_Toc142328194)

[**1.4.4.Nền tảng (Widget)** 12](#_Toc142328195)

[**1.4.5.Thiết kế các widget cụ thể** 13](#_Toc142328196)

[**1.5.Widget** 14](#_Toc142328197)

[**1.5.1.Khái niệm** 14](#_Toc142328198)

[**1.5.2.Phân loại widget** 16](#_Toc142328199)

[**1.6.Các widget quan trọng** 17](#_Toc142328200)

[**1.6.1.Widger tạo ứng dụng/thiết lập trang** 17](#_Toc142328201)

[**1.6.2.Widget dùng để bố trí** 18](#_Toc142328202)

[**1.6.4.Widget dùng để chứa nội dung** 19](#_Toc142328203)

[**1.6.5.Widget nhận đầu vào người dùng** 19](#_Toc142328204)

[**1.7.Bố cục (layout) giao diện trong flutter** 20](#_Toc142328205)

[**1.7.1.Bố cục** 20](#_Toc142328206)

[**1.7.2.Bố cục một widget** 22](#_Toc142328207)

[**1.7.3.Các loại widget bố cục** 22](#_Toc142328208)

[**1.8.Tổng quan về Androi Studio** 23](#_Toc142328209)

[**1.8.1.Cấu trúc dự án** 23](#_Toc142328210)

[**1.8.2.Hệ thống xây dựng Gradle** 25](#_Toc142328211)

[**1.8.3.Biến thể bản dựng** 25](#_Toc142328212)

[**1.8.4.Hỗ trợ nhiều APK** 26](#_Toc142328213)

[**1.8.5.Rút gọn tài nguyên** 26](#_Toc142328214)

[**1.8.6.Quản lý phụ thuộc** 26](#_Toc142328215)

[**CHƯƠNG 2: PHƯƠNG PHÁP THỰC HIỆN** 27](#_Toc142328216)

[**2.1.Sơ đồ phương pháp thực hiện** 27](#_Toc142328217)

[**2.2.Thiết kế giao diện** 28](#_Toc142328218)

[**2.2.1.Giao diện “Portable Translator”** 28](#_Toc142328219)

[**2.2.2.Giao diện Text Translator** 29](#_Toc142328220)

[**2.2.3.Giao diện Audio Translator** 30](#_Toc142328221)

[**2.2.4.Giao diện Image Text Translator** 31](#_Toc142328222)

[**2.3.Xác định bố cục widget cho ứng dụng** 32](#_Toc142328223)

[**2.3.1.Widget cho giao diện Portable Translator** 32](#_Toc142328224)

[**2.3.2.Widget cho giao diện Text Translator** 33](#_Toc142328225)

[**2.3.3.Widget cho giao diện Audio Translator** 34](#_Toc142328226)

[**2.3.4.Widget cho giao diện Image Text Translator** 35](#_Toc142328227)

[**CHƯƠNG 3: KẾT QUẢ ĐẠT ĐƯỢC** 36](#_Toc142328228)

[**3.1.Giao diện “Portable Translator”** 36](#_Toc142328229)

[**3.2.Giao diện Text Translator** 38](#_Toc142328230)

[**3.3.Giao diện Audio Translator** 40](#_Toc142328231)

[**3.4.Giao diện Image Text Translator** 42](#_Toc142328232)

[**CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN** 43](#_Toc142328233)

[**4.1.Kết luận** 43](#_Toc142328234)

[**4.2.Hướng phát triển** 43](#_Toc142328235)

[**TÀI LIỆU THAM KHẢO** 44](#_Toc142328236)

**MỤC LỤC BẢNG, HÌNH ẢNH**

[Bảng 1. 1.Kiểu dữ liệu ngôn ngữ Dart 1](#_Toc142255435)

[Bảng 1. 2.Toán tử số học 6](#_Toc142255436)

[Bảng 1. 3.Toán tử logic 7](#_Toc142255437)

[Hình 1. 1.Ngôn ngữ Dart 1](#_Toc142328237)

[Hình 1. 2.Ví dụ hàm có giá trị trả về 4](#_Toc142328238)

[Hình 1. 3.Ví dụ hàm không có giá trị trả về 5](#_Toc142328239)

[Hình 1. 4.Ví dụ toán tử so sánh 8](#_Toc142328240)

[Hình 1. 5.Flutter 9](#_Toc142328241)

[Hình 1. 6.Tính năng Flutter 10](#_Toc142328242)

[Hình 1. 7.Sơ đồ Widget 13](#_Toc142328243)

[Hình 1. 8.Ví dụ Widget 14](#_Toc142328244)

[Hình 1. 9.Sơ đồ ví dụ Widget 15](#_Toc142328245)

[Hình 1. 10.Ví dụ Stateless Widget 16](#_Toc142328246)

[Hình 1. 11.Ví dụ Statedul Widget 17](#_Toc142328247)

[Hình 1. 12.Ví dụ bố cục Flutter 21](#_Toc142328248)

[Hình 1. 13.Sơ đồ bố cục Flutter 21](#_Toc142328249)

[Hình 1. 14.Cấu trúc dự án trong Android Studio 24](#_Toc142328250)

[Hình 2. 1.Sơ đồ phương pháp thực hiện 27](#_Toc142328251)

[Hình 2. 2.Giao diện Portable Translator 28](#_Toc142328252)

[Hình 2. 3.Giao diện Text Translator 29](#_Toc142328253)

[Hình 2. 4.Giao diện Audio Translator 30](#_Toc142328254)

[Hình 2. 5.Giao diện Image Text Translator 31](#_Toc142328255)

[Hình 2. 6.Widget bố cục giao diện Portable Translator 32](#_Toc142328256)

[Hình 2. 7.Widget bố cục giao diện Text Translator 33](#_Toc142328257)

[Hình 2. 8.Widget bố cục giao diện Audio Translator 34](#_Toc142328258)

[Hình 2. 9.Widget bố cục giao diện Image Text Translator 35](#_Toc142328259)

[Hình 3. 1.Kết quả giao diện Portable Translator 37](#_Toc142328260)

[Hình 3. 2.Kết quả giao diện Text Translator 39](#_Toc142328261)

[Hình 3. 3.Kết quả giao diện Audio Translator 41](#_Toc142328262)

[Hình 3. 4.Kết quả giao diện Image Text Translator 42](#_Toc142328263)

# **CHƯƠNG 1: MỞ ĐẦU**

## **1.1.Tổng quan về ngôn ngữ Dart**

Flutter sử dụng ngôn ngữ lập trình Dart, ngôn ngữ Dart ra đời từ năm 2011 và được phát triển bởi Google, nhắm đến việc tạo ra ứng dụng đa nền tảng cho ứng dụng di động, máy tính, server và web. Dart là theo hướng đối tượng, đơn giản, dễ tiếp cận và cũng khá dễ hiểu. Ngôn ngữ này có cú pháp giống ngôn ngữ C.

Dart là ngôn ngữ mã nguồn mở và miễn phí, được phát triển trên Github. Phiên bản mới nhất hiện nay là 3.0.7.



Hình 1. 1.Ngôn ngữ Dart

### **1.1.1.Kiểu dữ liệu**

Bảng 1. 1.Kiểu dữ liệu ngôn ngữ Dart

|  |  |  |
| --- | --- | --- |
| **Loại dữ liệu** | **Ví dụ** | **Mô tả** |
| int | int a = 1; | Kiểu dữ liệu đại diện cho các số nguyên. Lưu trữ giá trị từ -2^63 đến 2^63-1. |
| double | double b = 2.5; | Kiểu dữ liệu double đại diện cho các số thực. Lưu trữ giá trị từ -1.7976931348623157E +308 đến 1.7976931348623157E+308. |
| num | num c = 7;  num d = 3.14; | Kiểu dữ liệu num đại diện cho một số, có thể là số nguyên hoặc số thực. Nó là một kiểu dữ liệu trừu tượng và là lớp cha của kiểu int và double. |
| String | String name = “John”;  String message = ‘Hello, $name!’; | Kiểu dữ liệu đại diện cho một chuỗi các ký tự Unicode. Được bao bọc bởi dấu ngoặc kép hoặc dấu nháy đơn. |
| Bool | bool isRaining = true;  bool isSunny = false; | Kiểu dữ liệu đại diện cho giá trị Boolean, có thể là true hoặc false. Được sử dụng để biểu thị các trạng thái hoặc điều kiện. |
| List | List<int> numbers = [1, 2, 3, 4, 5];  List<String> names = [‘John’, ‘Alice’, ‘Mary’] | Kiểu dữ liệu đại diện cho một danh sách các đối tượng. Một tập hợp các phần tử có thứ tự và có thể chứa các giá trị của bất kỳ kiểu dữ liệu nào. |
| Set | Set<int> numbers = {1, 2, 3, 4, 5};  Set<String> names = {‘John’, ‘Alice’, ‘Mary’}; | Đại diện cho một tập hợp các đối tượng không trùng lặp. Lưu trữ các phần tử một cách không có thứ tự và không cho phép các phần tử trùng lặp. |
| Map | Map<String, int> ages = {‘John’: 30, ‘Alice’: 25, ‘Mary’: 40}; | Đại diện cho một cặp khóa/giá trị. Lưu trữ các phần tử dưới dạng một bộ khóa/giá trị, trong đó mỗi khóa là duy nhất và được sử dụng để truy xuất giá trị tương ứng. |
| Runes | Runes heart = Runes(‘\u2665’); | Đại diện cho mỗi chuỗi Unicode. Được sử dụng để lưu trữ các ký tự Unicode và được bao bọc bởi dấu ngoặc kép. |
| symbol | Symbol mySymbol = #name; | Đại diện cho một biểu tượng duy nhất được sử dụng như một tên. Sử dụng trong một số trường hợp như định dang của hàm hoặc thuộc tính. |
| Object | Object myObject = ‘Hello’; | Kiểu dữ liệu đại diện cho tất cả các đối tượng trong Dart. Là lớp cha của tất cả các lớp khác trong Dart và được sử dụng để tham chiếu đến bất kỳ đối tượng nào. |
| dynamic | dynamic myVariable = ‘Hello’; | Kiểu dữ liệu đại diện cho bất kỳ kiểu dữ nào. Có nghĩa nó cho phéo biến có thể có bất kỳ kiểu dữ liệu nào và được sử dụng trong những trường hợp mà kiểu dữ liệu không thể xác định trước. |
| var | var x = 10;  var y = ‘Hello’; | Được sử dụng để khai báo biến mà không cần xác định kiểu dữ liệu của biến đó trước. Kiểu dữ liệu của biến sẽ được suy ra từ giá trị được gán cho biến. |

### **1.1.2.Các biến và các hàm**

Các biến là không gian tên trong bộ nhớ lưu trữ các giá trị. Tên của một biến được gọi là định danh (identifiers). Là nơi chứa dữ liệu, có thể lưu trữ giá trị của bất kỳ kiểu nào.

Ví dụ: khi khai báo một biến var myAge = 20;

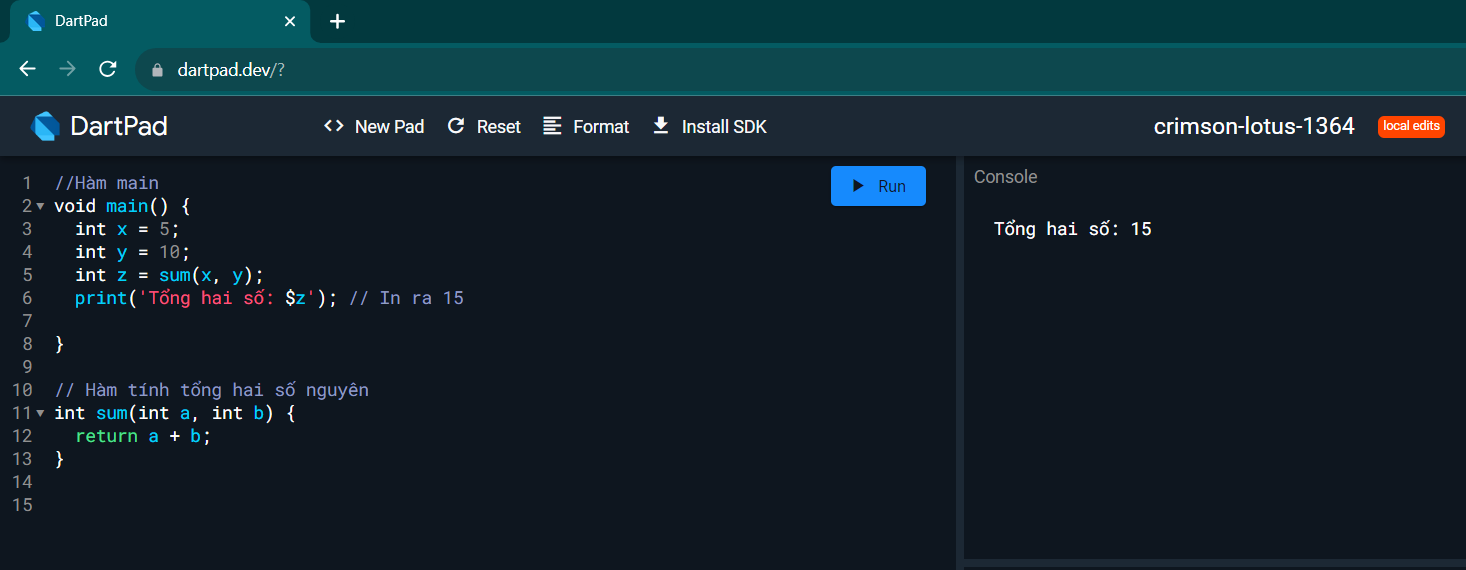
Ở ví dụ trên, ta có thể thấy myAge là một biến lưu trữ giá trị số nguyên 20. Cũng có thể thay thế cho nó là kiểu int, kiểu double. Tuy nhiên, trong Dart có một tính năng Type Inference, suy luận các loại giá trị. Cho nên, nếu ta tạo một biến với từ khóa var, Dart có thể suy ra biến đó thuộc loại số nguyên.

Bên cạnh biến, hàm là một khối mã thực thi một tác vụ cụ thể. Nó có thể được gọi từ bất kỳ đâu trong chương trình và có thể nhận các tham số đầu vào và trả về một giá trị đầu ra.

Trong Dart có hai loại hàm: hàm có giá trị trả về và hàm không có giá trị trả về

* Hàm có giá trị trả về: hàm có thể trả về một giá trị bất kỳ và kiểu dữ liệu của giá trị trả về được xác định trong khai báo hàm bằng từ khóa return

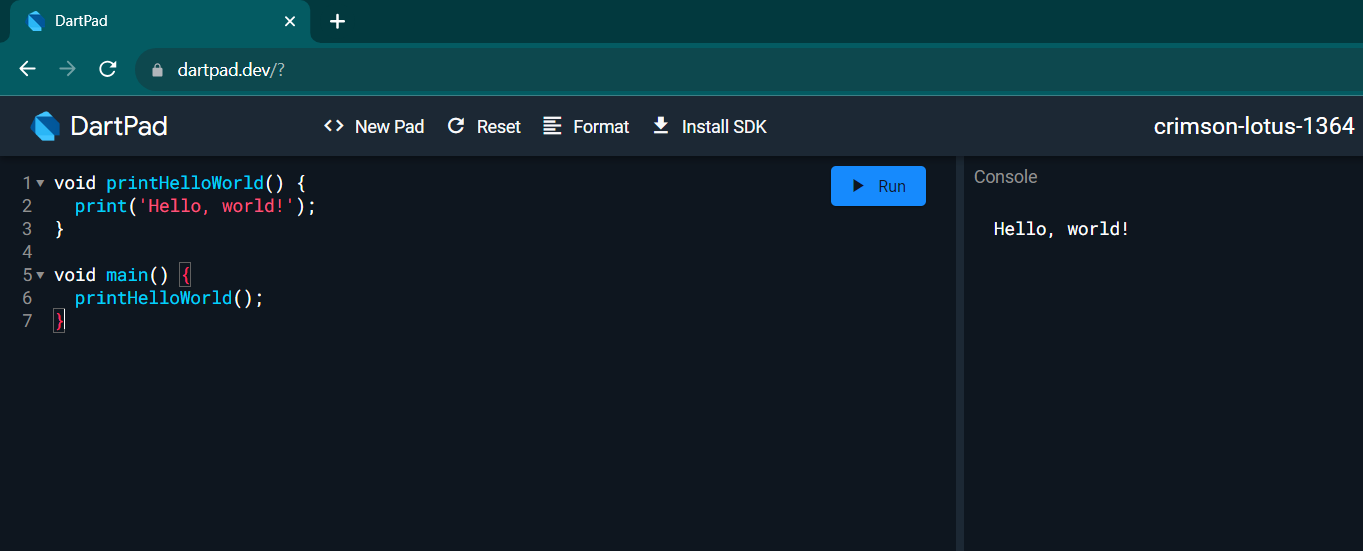
Ví dụ:



Hình 1. 2.Ví dụ hàm có giá trị trả về

* Hàm không có giá trị trả về: hàm không trả về giá trị nào và kiểu dữ liệu trả về được đặt là void

Ví dụ:



Hình 1. 3.Ví dụ hàm không có giá trị trả về

### **1.1.3.Toán tử (Operators)**

Ngôn ngữ Dart hỗ trợ tất cả các toán tử, toán tử để xác định cách thức làm việc giữa các toán hạng. Toán hạng có thể là một hằng số, biến số hoặc một lời gọi hàm. Các loại toán tử:

* Số học
* Tăng và giảm
* Logic
* So sánh

#### **1.1.3.1.Toán tử số học**

Toán tử số học là những phép cộng, trừ, nhân, chia cơ bản như toán học.

Giả sử biến A có giá trị là 3, biến B có giá trị là 4 thì:

Bảng 1. 2.Toán tử số học

|  |  |  |
| --- | --- | --- |
| Toán tử | Ví dụ | Miêu tả |
| + | A + B kết quả là 7 | Cộng hai toán hạng |
| - | A – B kết quả là -1 | Trừ toán hạng |
| \* | A \* B kết quả là 12 | Nhân hai toán hạng |
| / | B / A kết quả là 1 | Phép chia |
| % | B % A kết quả 1 | Phép chia lấy số dư |
| ++ | A++ kết quả là 4 | Toán tử tăng (++) tăng giá trị toán hạng thêm một đơn vị |
| -- | A - - kết quả là 2 | Toán tử giảm (- -), giảm giá trị toán hạng đi một đơn vị |

#### **1.1.3.2.Toán tử tăng và giảm**

Một toán tử tăng hay toán tử giảm được sử dụng như là một phần của biểu thức, thì sẽ có sự khác nhau quan trọng giữa dạng tiền tố và hậu tố. Nếu ta sử dụng dạng tiền tố thì toán tử tăng hoặc toán tử giảm được thực hiện trước biểu thức, và ngược lại nếu sử dụng dạng hậu tố thì toán tử tăng hoặc toán tử giảm được thực hiện sau khi biểu thức được ước lượng.

#### **1.1.3.3.Toán tử logic**

Toán tử logic được sử dụng để kiểm tra tính đúng đắn của một hoặc nhiều biểu thức. Gía trị trả về của các biểu thức này là một giá trị kiểu Boolean, true hoặc false.

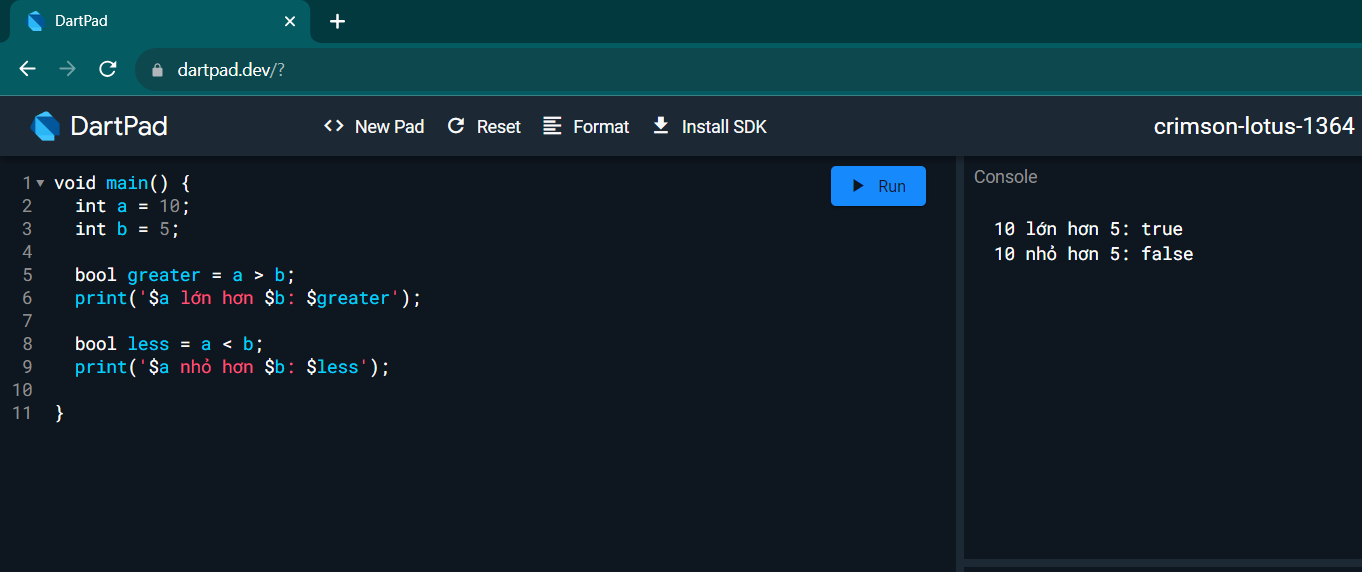
Bảng 1. 3.Toán tử logic

|  |  |  |
| --- | --- | --- |
| **Toán tử** | **Ví dụ** | **Mô tả** |
| && | bool a = true;  bool b = false;  bool c = a && b; | Toán tử && được gọi là toán tử logic AND (và) được sử dụng để kiểm tra nếu cả hai biểu thức logic đều đúng. Ngược lại nếu một trong hai biểu thức logic sai, thì kết quả là sai. Với ví dụ bên cạnh ta có thể biết c có giá trị false. |
| || | bool a = true;  bool b = false;  bool c = a || b; | Toán tử || được gọi là toán tử logic OR (hoặc) được sử dụng để kiểm tra nếu một trong hai biểu thức logic là đúng. Nếu cả hai biểu thức logic đề sai thì kết quả là sai. Với ví dụ bên cạnh ta có thể biết giá trị c là đúng |
| ! | Bool a = true;  Bool b = !a; | Toán tử ! được gọi là toán tử logic NOT (phủ định). Được sử dụng để đảo ngược giá trị của một biểu thức logic. Ví dụ bên cạnh ta có thể nói b sẽ có giá trị false. |

#### **1.1.3.4.Toán tử so sánh**

Toán tử so sánh được dùng để so sánh hai toán hạng với nhau. Kết quả trả lại của toán tử so sánh nếu đúng thì là True, còn ngược lại kết quả sẽ False.

Ví dụ:



Hình 1. 4.Ví dụ toán tử so sánh

### **1.1.4.Vòng lặp**

Các vòng lặp được sử dụng để thực thi một khố code lặp đi lặp lại cho đến khi một điều kiện được chỉ định trở thành đúng. Ngôn ngữ Dart hỗ trợ các loại câu lệnh lặp for, for…in, while, do…while.

## **1.2.Tổng quan về Flutter**

Flutter được Google giới thiệu là một người mới trong thế giới ứng dụng di động. Là một SDK mới của Google dành cho các thiết bị di độn giúp developers và designers xây dựng nhanh chóng ứng dụng dành cho các thiết bị di động (Android, IOS).

Flutter là dự án mã nguồn mở đang trong giai đoạn thử nghiệm. Flutter bao gồm Reactive framework và công nghệ hiển thị 2D (2D rendering engine) và các công cụ phát triển (development tool). Các thành phần này làm việc cùng nhau giúp ta thiết kế, xây dựng, test, debug ứng dụng. Không có gì ngạc nhiên khi Flutter giúp các nhà phát triển tạo ra các ứng dụng native đẹp mắt và giúp họ phát triển các ứng dụng đa nền tảng một cách dễ dàng.

Flutter là một SDK phát triển ứng dụng di động nguồn mở được tạo ra bởi Google. Nó được sử dụng để phát triển ứng dụng cho Android và IOS, cũng là phương thức chính để tạo ứng dụng cho Google Fuchsia.



Hình 1. 5.Flutter

Flutter gồm 2 thành phần quan trọng:

* Một SDK (Software Development Kit): một bộ sưu tập các công cụ sẽ giúp bạn phát triển các ứng dụng của ta.
* Một Framework (UI Library based on widgets): một tập hợp các thành phần giao diện người dùng (UI) có thể tái sử dụng (button, text, inputs, slider, …) giúp ta có thể cá nhân hóa tùy theo nhu cầu của riêng.

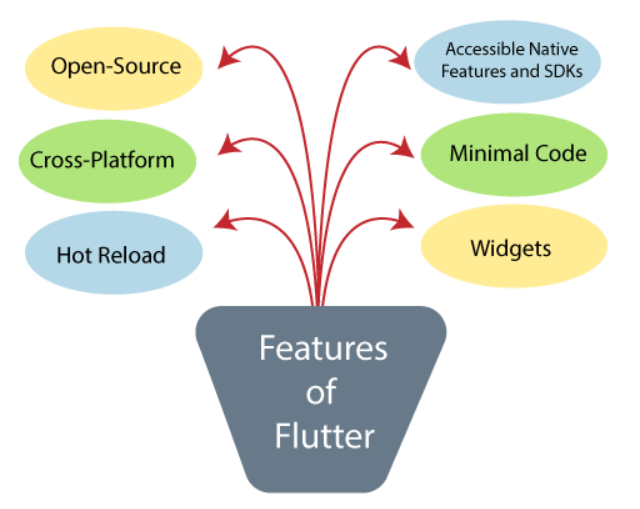
Để tạo một ứng dụng di động là một công việc rất phức tạp và đầy thử thách. Có rất nhiều framework có sẵn, cung cấp các tính năng hay để phát triển các ứng dụng di động. Để phát triển các ứng dụng dành cho thiết bị di động, Android cung cấp một framework gốc dựa trên Java và Kotlin, trong khi IOS cung cấp một framework dựa trên ngôn ngữ Objective-C/Swift.

Vì vậy, cần hai ngôn ngữ và framework khác nhau để phát triển ứng dụng cho cả hai hệ điều hành. Để khắc phục sự phức tạp này, có một số framework đã được giới thiệu hỗ trợ cả hệ điều hành cùng với các ứng dụng dành cho máy tính để bàn. Những loại framework này được được gọi là công cụ phát triển đa nên tảng.

Framework phát triển đa nền tảng có khả năng viết một code và có thể triển khai trên nhiều nền tảng khác nhau (Androi, IOS và máy tính để bàn). Nó tiết kiệm rất nhiều thời gian và nỗ lực phát triển của các nhà phát triển.

## **1.3.Các tính năng của Flutter**

Flutter cung cấp các phương pháp dễ dàng và đơn giản để bắt đầu xây dựng các ứng dụng dành cho thiết bị di động và máy tính để bàn đẹp mắt với bộ thiết kế material design và widget phong phú. Chúng ta sẽ thảo luận về các tính năng chính của nó để phát triển khung di động.



Hình 1. 6.Tính năng Flutter

* **Code nguồn mở (Open-Source):** Flutter là một framework code nguồn mở và miễn phí để phát triển ứng dụng di động.
* **Đa nền tảng (Cross-platform):** tính năng cho phép flutter viết code một lần, duy trì và có thể chạy trên các nền tảng khác nhau. Nó tiết kiệm thời gian, công sức và tiền bạc của các nhà phát triển.
* **Tải lại nóng (Hot Reload):** bất cứ khi nào nhà phát triển thực hiện các thay đổi trong mã, thì những thay đổi này có thể được nhìn thấy ngay lập tức với Hot Reload. Nó có nghĩa là những thay đổi hiển thị ngay lập tức trong chính ứng dụng. Đây là một tính năng rất tiện dụng, cho phép nhà phát triển sửa các lỗi ngay lập tức.
* **Các tính năng và SDK gốc có thể truy cập (Accessible Native Features and SDKs):** tính năng này cho phép quá trình phát triển ứng dụng trở nên dễ dàng và thú vị thông qua code gốc của Flutter, tích hợp bên thứ ba và API nền tảng. Do đó, ta có thể dễ dàng truy cập SDK trên cả hai nền tảng.
* **Code tối thiểu (Minimal code):** ứng dụng Flutter được phát triển vởi ngôn ngữ lập trình Dart, sử dụng biên dịch JIT và AOT để cải thiện thời gian khởi động tổng thể, hoạt động và tăng tốc hiệu suất. JIT nâng cao hệ thống phát triển và làm mới giao diện người dùng mà không cần nổ lực thêm vào việc xây dựng hệ thống mới.
* **Widget:** Framework Flutter cung cấp các widget có khả năng phát triển các thiết kế cụ thể có thể tùy chỉnh. Quan trọng nhất, Flutter có hai bộ widget: Material Design và Cupertino widget giúp cung cấp trải nghiệm không bị trục trặc trên tất cả các nền tảng.

## **1.4.Tổng quan kiến trúc của Flutter**

### **1.4.1.Kiến trúc của Flutter**

Trong phần này, ta sẽ thảo luận về kiến trúc của Flutter framework. Kiến trúc Flutter chủ yếu bao gồm 4 thành phần:

* Động cơ rung (Flutter Engine)
* Thư viện nền tảng (Foundation Library)
* Tiện ích (Widgets)
* Thiết kế Widget cụ thể (Design Specific Widgets)

### **1.4.2.Flutter Engine**

Nó là một cổng để giúp chạy ứng dụng di động chất lượng cao và cơ bản dựa trên ngôn ngữ C++. Nó triển khai các thư viện lõi Flutter bao gồm animation và đồ họa, tệp và mạng I / O, kiến trúc plugin, hỗ trợ khả năng truy cập và thời gian chạy dart để phát triển, biên dịch và chạy các ứng dụng Flutter. Phải sử dụng thư đồ họa nguồn mở của Google, Skia để hiển thị đồ họa cấp thấp.

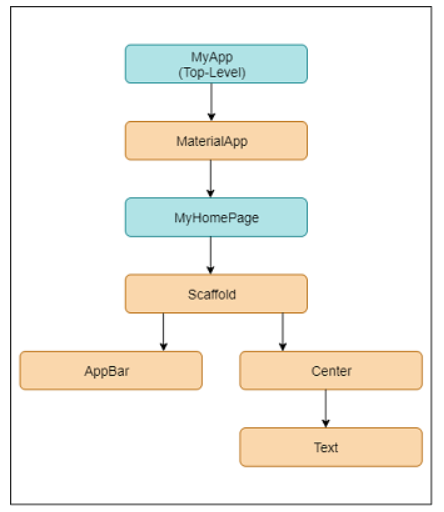
### **1.4.3.Thư viện nền tảng (Foundation Library)**

Nó chứa tất cả các gói cần thiết các khối xây dựng cơ bản để viết một ứng dụng Flutter. Các thư viện này được viết bằng ngôn ngữ Dart.

### **1.4.4.Nền tảng (Widget)**

Trong Flutter, mọi thứ đều là widget, đó là khái niệm cốt lõi của framework. Wiget trong Flutter về cơ bản là một thành phần giao diện người dùng ảnh hưởng và kiểm soát chế độ xem và giao diện của ứng dụng. Nó đại diện cho một mô tả bất biến của một phần giao diện người dùng và bao gồm đồ họa, văn bản, hình dạng và animation được tạo bằng các widget. Các widget tương tự như các thành React.

Trong Flutter, bản thân ứng dụng là một widget chứa nhiều widget con. Điều đó có nghĩa là ứng dụng là tiện ích con cấp nhất và giao diện người dùng của nó được xây dựng bằng các sử dụng một hoặc nhiều tiện ích con, bao gồm các tiện ích con phụ. Tính năng này giúp ta tạo một giao diện người dùng phức tạp rất dễ dàng.



Hình 1. 7.Sơ đồ Widget

Với sơ đồ trên, ta có thể thấy rằng tất cả các thành phần đều là các widget có chứa các widget con. Do đó, ứng dụng Flutter tự nó là một widget.

### **1.4.5.Thiết kế các widget cụ thể**

Framework Flutter có hai bộ widget phù hợp với các ngôn ngữ thiết kế cụ thể. Đây là Material Design cho ứng dụng Androi và Cupertino Style cho ứng dụng IOS.

## **1.5.Widget**

### **1.5.1.Khái niệm**

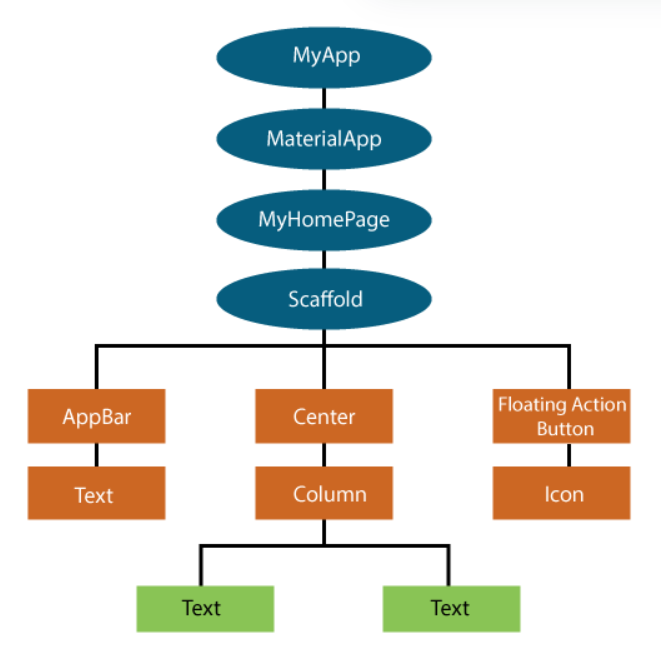
“Mọi thứ đều là widget” là khái niệm chính của Flutter. Widget trong Flutter là các thành phần xây dựng UI của ứng dụng. Mọi thứ bao gồm văn bản, hình dạng, hoạt ảnh đều được tạo thông qua widget.

Các widget trong chương trình được tổ chức dưới dạng cây, widget chứa các widget khác gọi là widget cha, widget được chứa trong widget khác gọi là widget con. Ngay cả ứng dụng Flutter cũng là một widget lớn chứa các widget khác.

Ví dụ: ứng dụng flutter đơn giản sử dụng widget material



Hình 1. 8.Ví dụ Widget



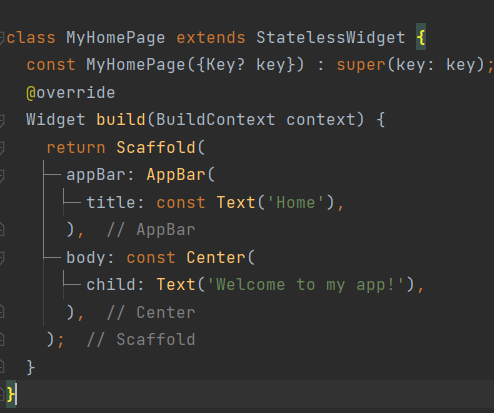
Hình 1. 9.Sơ đồ ví dụ Widget

### **1.5.2.Phân loại widget**

Widget gồm hai loại: “Stateless widget” và “Stateful widget”

* Stateless widget: loại widget chuyên dùng để chỉ hiển thị dữ liệu. Chúng không thay đổi khi ứng dụng đang chạy.

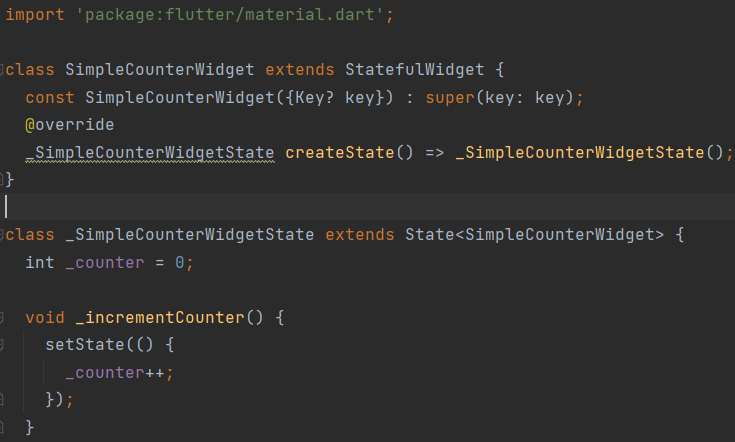
Ví dụ:



Hình 1. 10.Ví dụ Stateless Widget

* Stateful widget: loại widget này chứa dữ liệu có thể thay đổi khi đang trong vòng đời của widget. Tập hợp các dữ liệu này còn được gọi là “state”, để cập nhật dữ liệu và hiển thị dữ liệu này ở UI chúng ta cần gọi SetState. Sau khi state của stateful widget thay đổi widget sẽ được rebuild ở khung hình.

Ví dụ:



Hình 1. 11.Ví dụ Statedul Widget

Cách nhận biết 2 loại widget trên:

* Stateless widget: có phương thức build() và không có phương thức createState()
* Stateful widget: có phương thức build và có phương thức createState()

## **1.6.Các widget quan trọng**

### **1.6.1.Widger tạo ứng dụng/thiết lập trang**

MaterialApp/ CupertinoApp: widget gói nhiều widget thường dùng để xây dựng ứng dụng theo phong cách có sẵn cho Android và iOS. Widget này cung cấp các chức năng như tương tác phím vật lý, vuốt để lùi, tùy chỉnh chủ đề, … đây là widget vô cùng quan trọng để quyết định phong cách của ứng dụng. => widget kế thừa các kiến trúc theo chuẩn design marterial.

Scaffold/ CupertinoPageScaffold: widget đóng vai trò như phần nền để bố trí các widget khác theo phong cách Material (Android) hoặc Cupertino (iOS). Loại widget này cung cấp nơi để đặt các widget như NavigationBar, AppBar, BottomSheet. => widget bao chứa toàn bộ code chúng ta vào trong nó (phủ đầy giao diện của chúng ta).

* Material giống như là thẻ html có thẻ head rồi nhúng css của marterial.
* Scaffold như là thẻ body của html

### **1.6.2.Widget dùng để bố trí**

* Container: một widget rất đa dụng và thường được sử dụng trong Flutter. Widget này có thể kích thước tùy chỉnh, thêm viền, tô màu, chỉnh hình dáng, … Nó có thể chứa một widget con và căn chỉnh widget con theo nhiều cách khác nhau.
* Row/ Column: dùng để đặt nhiều widget cạnh nhau theo chiều ngang hoặc dọc. Các widget con có thể được căn chỉnh theo nhiều cách khác nhau. Để tạo kiểu Row/ Column, lập trình viên thường kết hợp sử dụng với Container.
* Flexible/ Expanded: giúp tùy chỉnh kích thước của widget trong Row hoặc Column.
* Stack: dùng để đặt nhiều widget chồng lên nhau, khi dùng Stack có thể dùng widget Positioned để hỗ trợ đặt widget ở một vị trí dễ dàng hơn.
* Card: một Container có sẵn các thiết lập tạo kiểu như đổ bóng, màu nền, bo tròn góc
* Grid: Widget này tạo ra một lưới hiển thị các widget con theo các hàng và cột. Thích hợp cho việc hiển thị danh sách các mục tiêu có cùng kiểu dữ liệu.
* **1.6.3.Widget cuộn**
* GridView: dùng để chứa tập hợp nhiều widget con và được sắp xếp theo dạng lưới. Thường dùng để chứa dữ liệu dạng chuỗi và có số lượng lớn.
* ListView: dùng để chứa tập hợp nhiều widget con, giống như Row và Column nhưng có thể cuộn. Thường dùng để chứa dữ liệu dạng chuỗi và có số lượng lớn.
* ListTile: một container đã được tạo kiểu sẵn thường dùng để tạo các phần tử con trong danh sách. Mỗi ListTile hỗ trợ tạo widget với phần đầu, tiêu đề, các nút hành động. Widget tiện dụng để tạo UI cho nhiều loại ứng dụng trên thị trường.

### **1.6.4.Widget dùng để chứa nội dung**

* Container: với mục đích widget cho phép chúng ta tạo ra một hình dạng, vị trí và kiểu trang trí tùy chỉnh cho nội dung con bên trong.
* Center: widget này cho phép bạn căn giữa widget trong chính nó.
* ListView: widget này cho phép chúng ta hiển thị một danh sách cuộn có thể chứa nhiều widget con. ListView cung cấp các tùy chọn như ListView.builder, ListView.seperated và ListView.custom để tạo danh sách theo nhu cầu cụ thể.
* Grid: Widget này tạo ra một lưới hiển thị các widget con theo các hàng và cột. Thích hợp cho việc hiển thị danh sách các mục tiêu có cùng kiểu dữ liệu.
* TableData: Widget này giúp chúng ta hiển thị dữ liệu dưới dạng bảng có các cột và hàng.
* Text: widget dùng để chứa văn bản, văn bản có thể được chỉnh kiểu, chỉnh cách hiển thị, … Đây là một trong những widget không thể thiếu với Flutter.
* Image: widget dùng để chứa và hiển thị ảnh, hỗ trợ từ nhiều nguồn (trong bộ nhớ, từ internet, …).
* Icon: dùng để hiển thị biểu tượng, hỗ trợ nhiều biểu tượng đa dạng theo phong cách Material, Cupertino. Có thể thêm nhiều biểu tượng từ bên ngoài một cách dễ dàng.

### **1.6.5.Widget nhận đầu vào người dùng**

* TextField: tạo ra một trường để người dùng có thể nhập văn bản, hỗ trợ nhiều loại bàn phím.
* GestureDetector: có thể kết hợp với hầu hết mọi loại widget để xử lý các hành động của người dùng như bấm, nhấp đôi, vuốt, kéo, … trên các widget này.
* Button: Widget này thì thường bao gồm một image hoặc text hoặc có thể bao gồm cả hai. Khi người dùng chạm vào nó, nó sẽ kích hoạt sự kiện nhấp chuột và thực hiện hành động thích hợp. Trong Flutter có nhiều loại nút khác nhau. Các loại nút được sử dụng cho mục đích khác nhau. Một số loại button cơ bản:
* Flat Button: một button không có màu nền và có màu chữ tùy chỉnh. Được sử dụng để thực hiện các hành động phụ hoặc hành động thứ cấp trong ứng dụng của chúng ta, như xóa hoặc hủy bỏ.
* Raised Button: một button với màu nền và màu chữ tùy chỉnh. Được sử dụng để thực hiện các hành động chính trong ứng dụng của chúng ta, chẳng hạn như đăng nhập, đăng ký
* Floating Button: một button dạng hình tròn nổi thường được đặt ở góc dưới bên phải của màn hình. Nó được sử dụng để thực hiện các hành động quan trọng trong ứng dụng của chúng ta, như tạo mới hoặc gửi tin nhắn.
* Drop Down Button: một button dạng dropdown, cho phép người dùng chọn giá trị từ một danh sách các giá trị. Được sử dụng để lựa chọn các tùy chọn hoặc danh sách trong ứng dụng của chúng ta
* Icon Button: một button dạng icon, có thể sử dụng để thực hiện các hành động nhanh chóng trong ứng dụng của chúng ta, chẳng hạn như chia sẻ hoặc tải xuống.
* PopupMenu Button:
* OutlineButton: Đây là một button với đường viền và không có màu nền. Nó được sử dụng để thực hiện các hành động phụ hoặc hành động thứ cấp trong ứng dụng của bạn.
* RawMaterial Button: Đây là một button với đường viền và màu nền. Nó được sử dụng để thực hiện các hành động phức tạp hoặc tùy chỉnh trong ứng dụng của bạn.

## **1.7.Bố cục (layout) giao diện trong flutter**

### **1.7.1.Bố cục**

Khái niệm chính của cơ chế bố trí là widget. Hình ảnh, biểu tượng, văn bản hay bố cục của ứng dụng đều là widget. Một số thứ chúng ta không thấy trên giao diện ứng dụng người dùng như các hàng, cột, lưới cũng là widget.

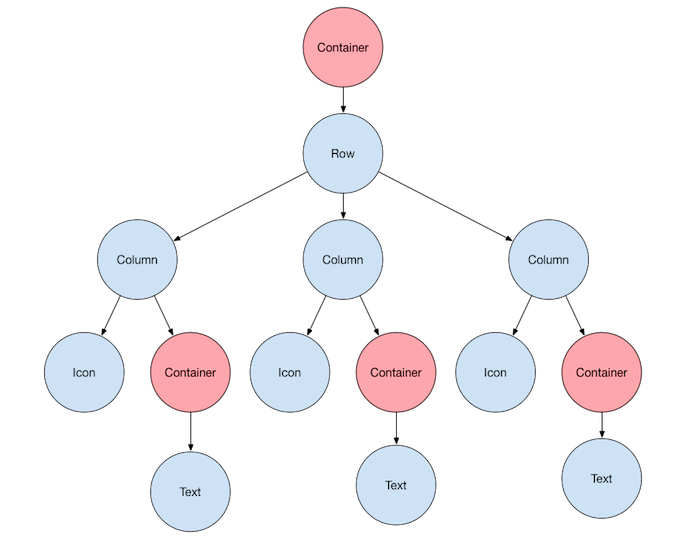
Flutter cho phép ta tạo bố cục bằng cách soạn nhiều widget để xây dựng các widget phức tạp hơn.

Ví dụ: Hình ảnh dưới đấy hiển thị ba biểu biểu tượng với nhãn bên dưới của mỗi biểu tượng.



Hình 1. 12.Ví dụ bố cục Flutter

Tiếp theo, ta có thể thấy bố cục trực quan của hình ảnh trên thông qua hình vẽ dưới đây hiển thị một hàng gồm ba cột và các cột này đều chứa một biểu tượng và trong mỗi container có nhãn



Hình 1. 13.Sơ đồ bố cục Flutter

Trong hình trên, container là một widget cho phép tùy chỉnh widget con. Sử dụng khi muốn thêm đường viền, đệm, lề, màu nền và nhiều thứ khác.

Mỗi widget văn bản được đặt trong một container để thêm lề. Toàn bộ hàng cũng được đặt trong một vùng chứa để thêm lề và phần đệm xung quanh hàng.

Các phần còn lại của giao diện người dùng được kiểm soát bởi các thuộc tính như màu sắc, kiểu văn bản.

### **1.7.2.Bố cục một widget**

Các bước bố trí widget:

Bước 1: Chọn bố cục widget.

Bước 2: Tạo một widget hiển thị.

Bước 3: Thêm widget hiển thị vào widget layout.

Bước 4: Thêm widget bố cục vào trang muốn hiển thị.

### **1.7.3.Các loại widget bố cục**

Phân loại widget bố cục thành hai loại:

* Widget đơn.
* Widget đa

#### **1.7.2.1.Widget đơn**

* Widget bố cục con duy nhất là một loại widget, có thể chỉ có một widget bên trong widget bố cục mẹ. Các widget này cũng có thể chứa chức năng bố cục đặc biệt.
* Flutter cung cấp cho người dùng nhiều widget con để làm giao diện người dùng của ứng dụng trở nên hấp dẫn.
* Đặc biệt, khi người dùng sử dụng các widget này một cách thích hợp, có thể tiết kiệm thời gian của ta và làm cho code ứng dụng dễ đọc hơn

Ví dụ: Text để hiển thị văn bản, Image để hiển thị hình ảnh.

#### **1.7.2.2.Đa widget**

* Đa widget là một loại widget chứa nhiều hơn một widget con bên trong và cách bố trí của các widget này là duy nhất.

Ví dụ: Column để xếp các widget theo chiều dọc, AppBar để hiển thị thanh tiêu đề và các nút điều hướng.

## **1.8.Tổng quan về Androi Studio**

Androi Studio là môi trường phát triển tích hợp (IDE) chính thức để phát triển ứng dụng Android. Nhờ có công cụ cho nhà phát triển và trình soạn thảo code mạnh mẽ của IntelliJ IDEA, Androi Studio cung cấp thêm nhiều tính năng giúp chúng ta nâng cao năng suất khi xây dựng ứng dụng Android, chẳng hạn như:

* Một hệ thống xây dựng linh hoạt dựa trên Gradle.
* Một trình mô phỏng nhanh và nhiều tính năng.
* Một môi trường hợp nhất nơi bạn có thể phát triển cho mọi thiết bị Android.
* Tính năng Live Edit (chỉnh sửa trực tiếp) để cập nhật các thành phần kết hợp trong trình mô phỏng và thiết bị thực theo thời gian.
* Mẫu code và quá trình tích hợp GitHub để giúp chúng ta xây dựng các tính năng ứng dụng phổ biến như nhập code mẫu.
* Đa dạng khung và công cụ thử nghiệm.
* Công cụ tìm lỗi code để nắm bắt hiệu suất, khả năng hữu dụng, khả năng tương thích với phiên bản và các vấn đề khác.
* Hỗ trợ C++ và NDK.
* Tích hợp sẵn tính năng hỗ trợ Google Cloud Platform, giúp chúng ta dễ dàng tích hợp Google Cloud Messaging và App Engine.

### **1.8.1.Cấu trúc dự án**

Mỗi dự án trong Android Studio chứa một hoặc nhiều mô-đun có tệp mã nguồn và tệp tài nguyên. Có hai loại mô-đun sau:

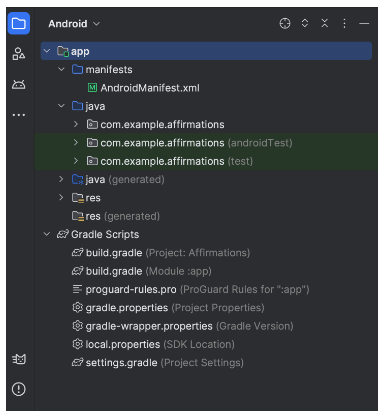
* Mô-đun ứng dụng Androi.
* Mô-đun thư viện.
* Mô-đun Google App Engine.

Theo mặc đinh, Android Studio thể hiện các tệp dự án của bạn trọng chế độ xem dự án Android. Khung hiển thị này được sắp xếp theo mô-đun để chúng ta có thể truy cập nhanh vào các tệp nguồn chính của dự án. Chúng ta có thể thấy mọi tệp bản dựng ở cấp cao nhất trong Gradle Scripts.

Trong mỗi môn-đun ứng dụng có chứa các thư mục sau:

* Manifests: chứa tệp AndroidManifest.xml.
* Java: chứa các tệp mã nguồn Java và Kotlin, bao gồm cả code kiểm thử Junit.
* Res: chứa mọi tài nguyên không phải đoạn mã, chẳng hạn như chuỗi giao diện người dùng và hình ảnh bitmap.

Cấu trúc dự án Android trên ổ đĩa khác với cách trình bày ở đây. Để xem cấu trúc tệp của dự án thực tế, hãy chọn Project thay vì Android trên trình đơn Project.



Hình 1. 14.Cấu trúc dự án trong Android Studio

### **1.8.2.Hệ thống xây dựng Gradle**

Android Studio sử dụng Gradle làm nền tảng cho hệ thống xây dựng với nhiều tính năng dành riêng cho Android do trình bổ trợ Android cho Gradle cung cấp. Hệ thống xây dựng này hoạt động như một công cụ tích hợp trên trình đơn Android Studio và độc lập với dòng lệnh. Chúng ta có thể sử dụng các tính năng của hệ thống xây dựng để làm những việc sau:

* Tùy chỉnh, cấu hình và mở rộng quy trình xây dựng.
* Tạo nhiều tệp APK cho ứng dụng với nhiều tính năng trong khi sử dụng cùng một dự án và mô-đun.
* Sử dụng lại mã và tài nguyên trên các nhóm tài nguyên (source set)

Ngoài ra, khi vận dụng tính linh hoạt của Gradle, bạn có thể làm được những việc này mà không cần sửa đổi các tệp nguồn lỗi của ứng dụng.

Tệp bản dựng Android Studio có tên build.gradle.kts nếu bạn sử dụng Kotlin hoặc có tên là build.gradle nếu bạn sử dụng Groovy. Đây là các tệp văn bản thuần túy sử dụng cú pháp Kotlin hoặc Groovy để định cấu hình bản dựng bằng các phần tử do trình hỗ trợ Android cho Gradle cung cấp. Mỗi dự án có một tệp bản dựng cao nhất cho toàn bộ dự án và các tệp bản dựng cấp mô-đun riêng cho từng mô-đun. Khi chúng ta nhập một dự án hiện có. Android Studio sẽ tự động tạo ra các tệp bản dựng cần thiết.

### **1.8.3.Biến thể bản dựng**

Hệ thống xây dựng có thể giúp bạn tạo nhiều phiên bản của cùng một ứng dụng trong một dự án duy nhất. Việc này khá hữu ích nếu ứng dụng của chúng ta có cả phiên bản miễn phí và phiên bản tính phí, ngoài ra nếu bạn muốn phân phối nhiều tệp APK qua Google Play tùy theo cấu hình thiết bị.

### **1.8.4.Hỗ trợ nhiều APK**

Tính năng hỗ trợ nhiều APK cho phép chúng ta tạo nhiều APK một cách hiệu quả dựa trên mật độ màn hình hoặc ABI (giao diện nhị phân ứng dụng). Ví dụ: chúng ta có thể tạo các APK riêng biệt của một ứng dụng cho mật độ màn hình hdpi và mdpi, trong khi vẫn xem xét các APK này một biến thể duy nhất cũng như cho phép chúng dùng chung chế độ cài đặt của APK kiểm thử, javac, dx và ProGuard.

### **1.8.5.Rút gọn tài nguyên**

Tính năng rút gọn tài nguyên (resource shrinking) trong Android Studio sẽ tự động xóa các tài nguyên không dùng đến khỏi các phần phụ thuộc trong thư viện và ứng dụng đóng gói.

Ví dụ: nếu ứng dụng của chúng ta dùng dịch vụ Google Play để truy cập chức năng Google Drive, và đang không sử dụng đăng nhập Google, thì resource shrinking có thể loại bỏ các tài nguyên drawable cho các nút SignInButton.

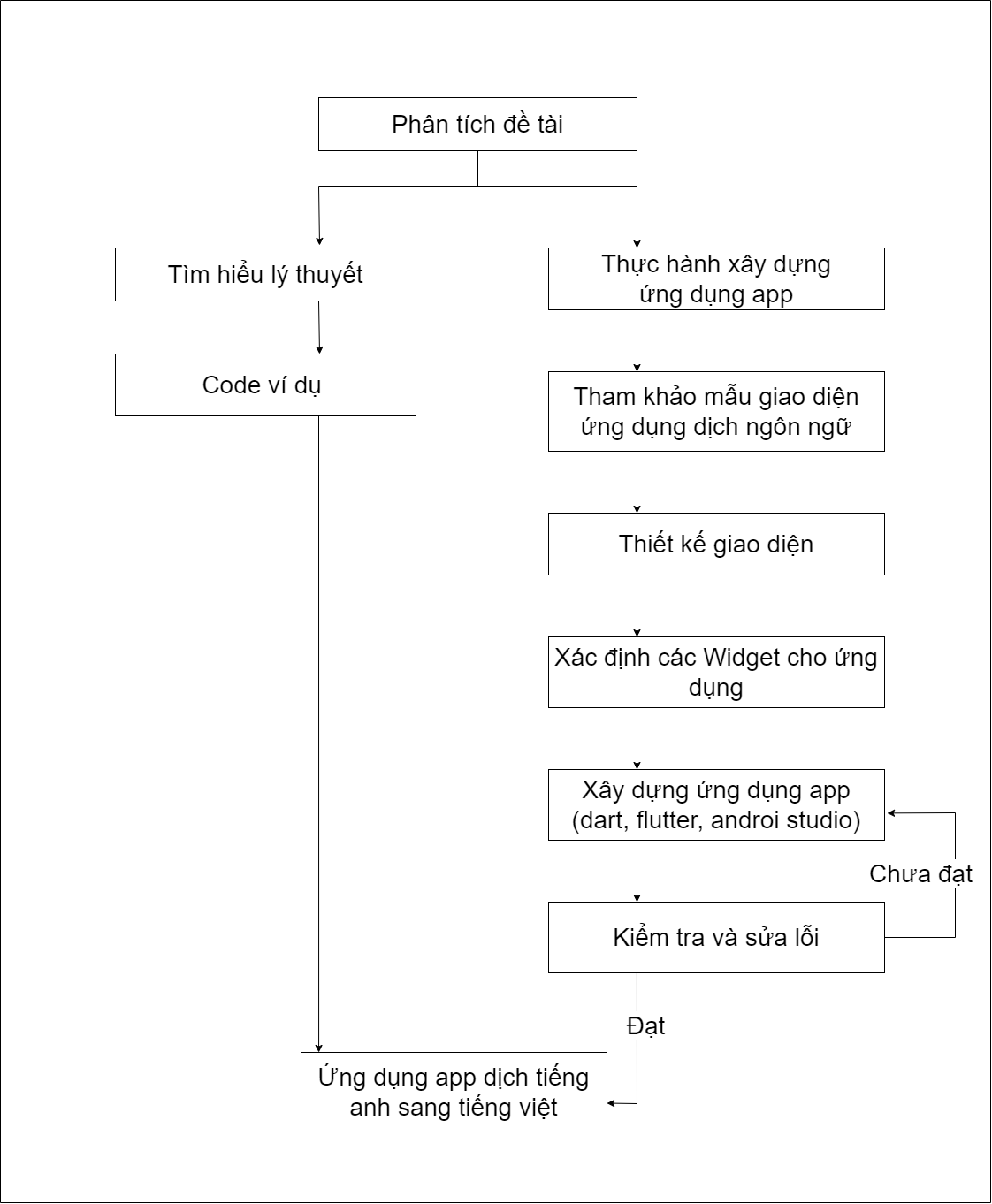
### **1.8.6.Quản lý phụ thuộc**

Trong dự án, các phụ thuộc được đặt tên và được quản lý thông qua tập lệnh của một module cụ thể. Gradle sẽ tìm kiếm các phụ thuộc này và đưa chúng vào quá trình xây dựng. Lúc này, ta có thể khai báo các phụ thuộc module, phụ thuộc nhị phân từ xa (remote binary dependencies), và phụ thuộc nhị phân cục bộ (local binary dependencies) trong tệp build.gradle.kts.

Android Studio được cấu hình mặc định để sử dụng kho lưu trữ Maven Central. Cấu hình này được bao gồm trong tệp build cấp cao nhất của dự án.

# **CHƯƠNG 2: PHƯƠNG PHÁP THỰC HIỆN**

## **2.1.Sơ đồ phương pháp thực hiện**

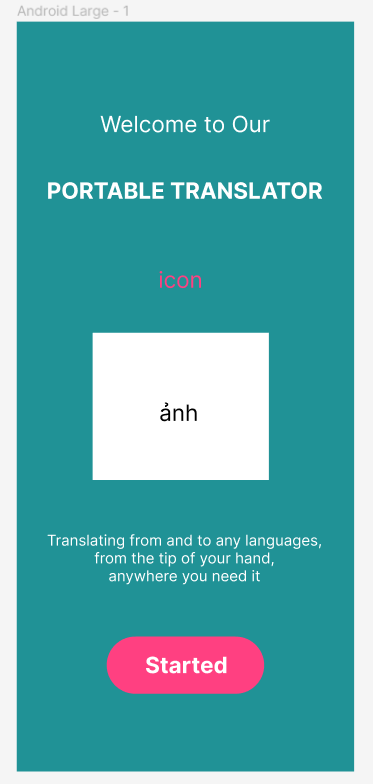
****

Hình 2. 1.Sơ đồ phương pháp thực hiện

## **2.2.Thiết kế giao diện**

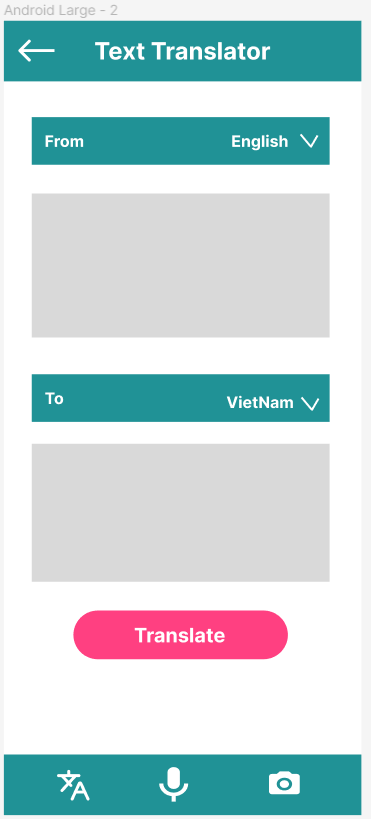
Thiết kế giao diện thực hiện trên phần mềm Figma

### **2.2.1.Giao diện “Portable Translator”**

****

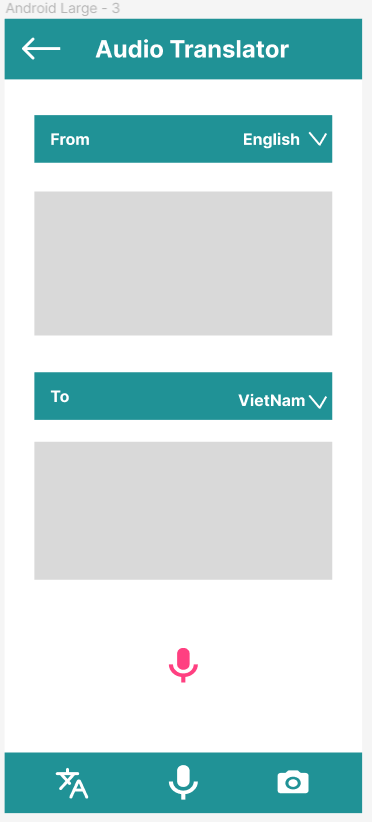
Hình 2. 2.Giao diện Portable Translator

### **2.2.2.Giao diện Text Translator**

****

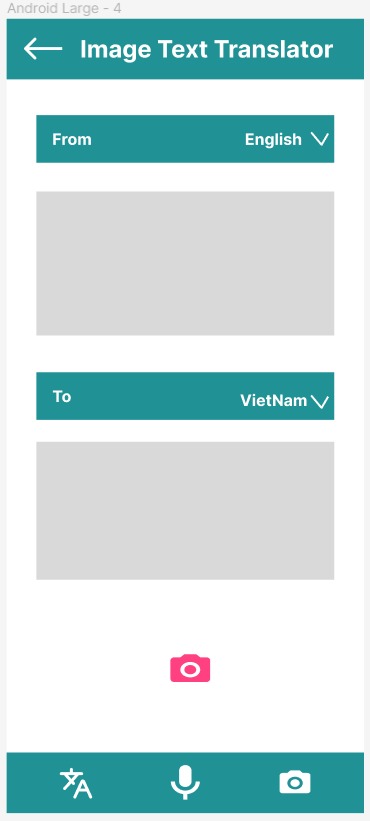
Hình 2. 3.Giao diện Text Translator

### **2.2.3.Giao diện Audio Translator**

****

Hình 2. 4.Giao diện Audio Translator

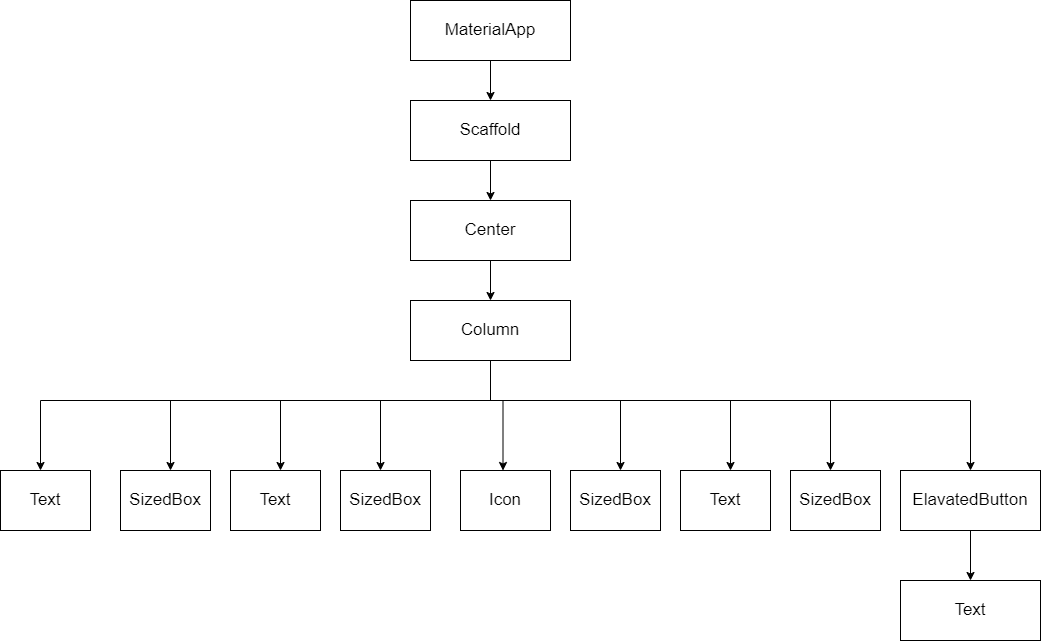
### **2.2.4.Giao diện Image Text Translator**

****

Hình 2. 5.Giao diện Image Text Translator

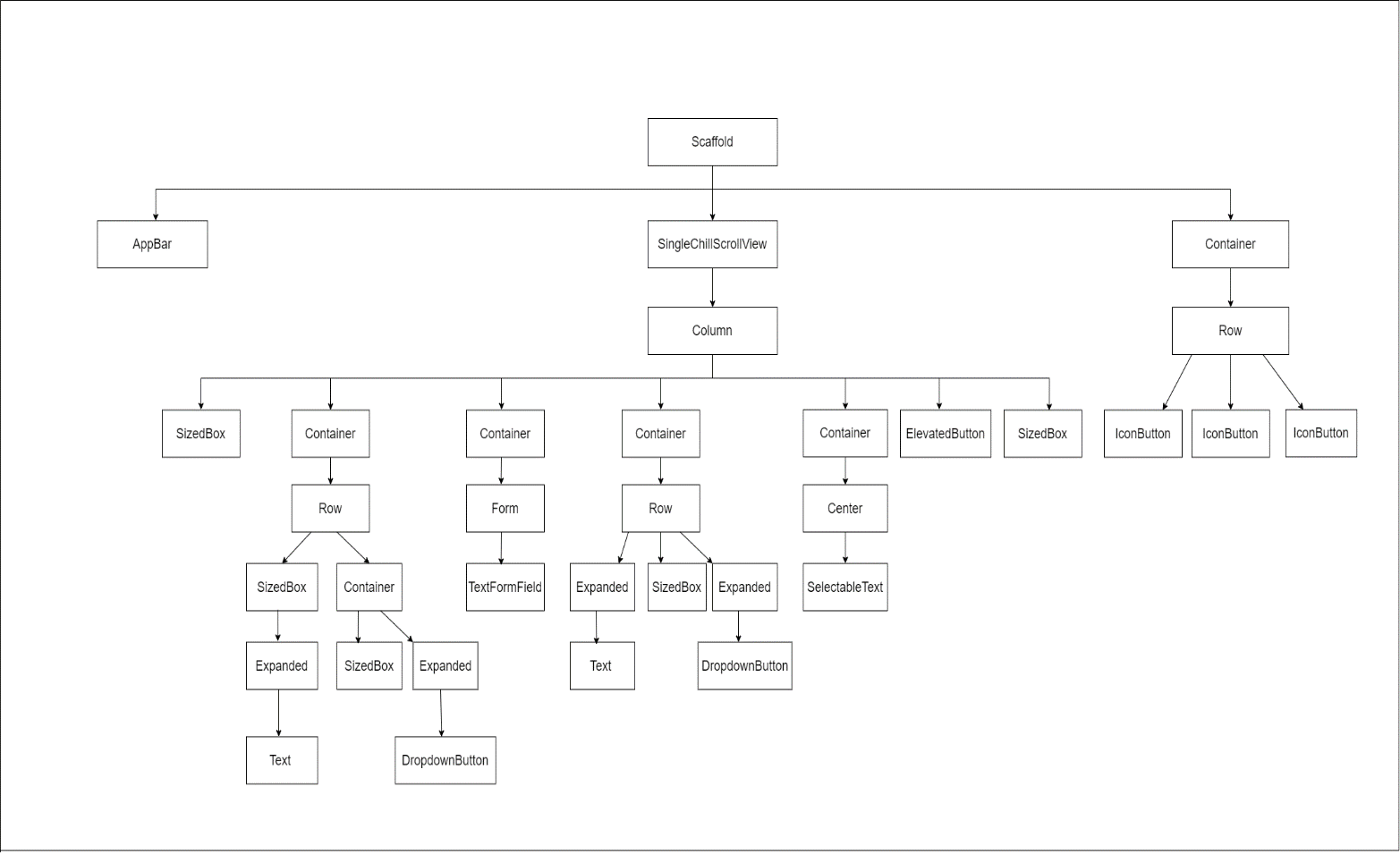
## **2.3.Xác định bố cục widget cho ứng dụng**

### **2.3.1.Widget cho giao diện Portable Translator**

****

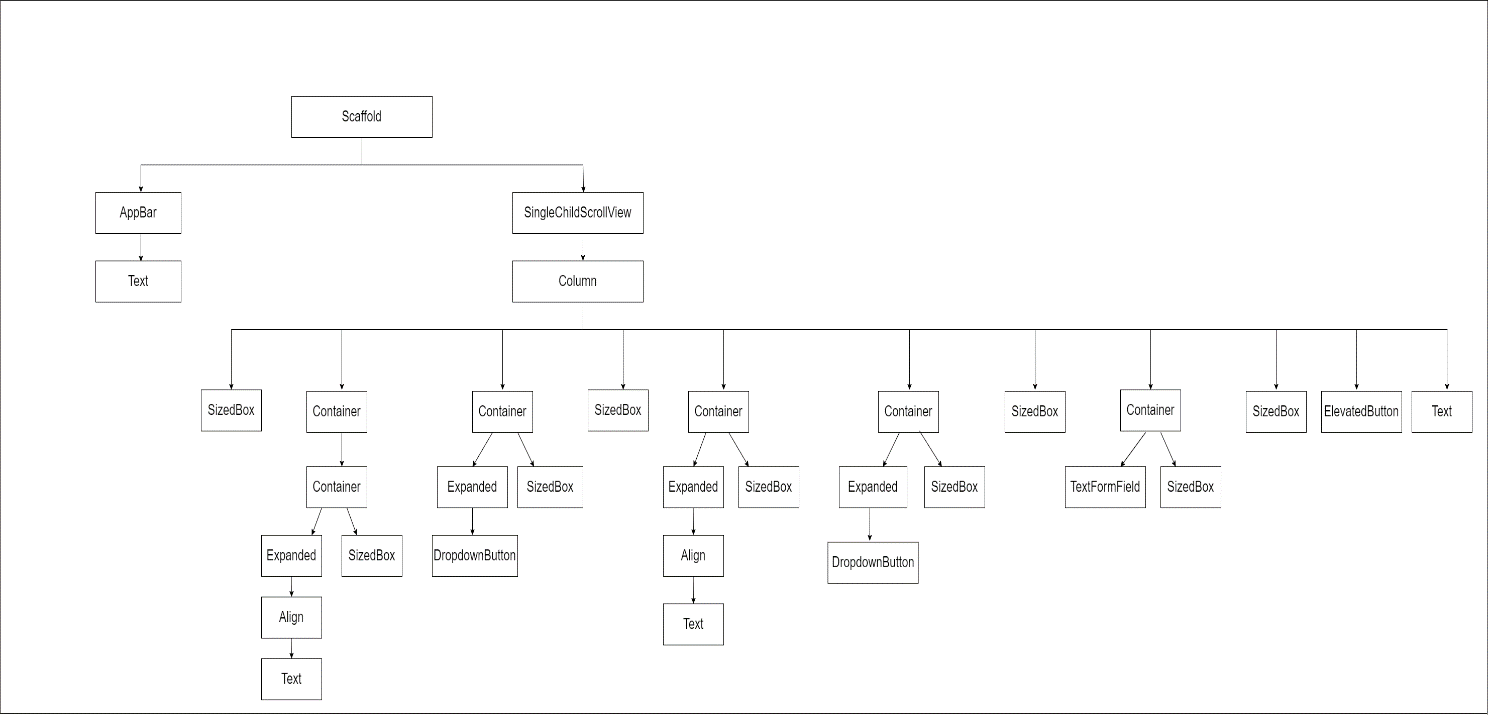
Hình 2. 6.Widget bố cục giao diện Portable Translator

### **2.3.2.Widget cho giao diện Text Translator**

****

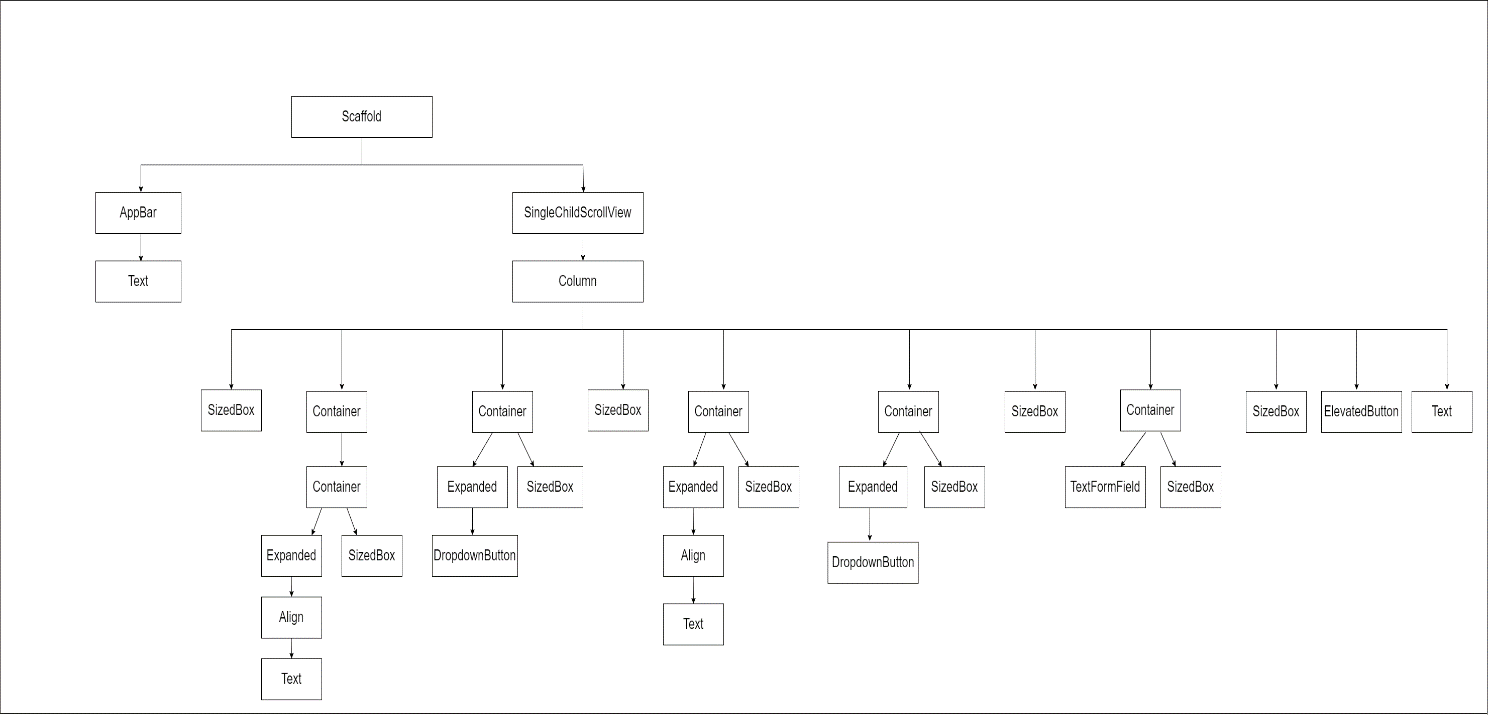
Hình 2. 7.Widget bố cục giao diện Text Translator

### **2.3.3.Widget cho giao diện Audio Translator**

****

Hình 2. 8.Widget bố cục giao diện Audio Translator

### **2.3.4.Widget cho giao diện Image Text Translator**



Hình 2. 9.Widget bố cục giao diện Image Text Translator

# **CHƯƠNG 3: KẾT QUẢ ĐẠT ĐƯỢC**

## **3.1.Giao diện “Portable Translator”**

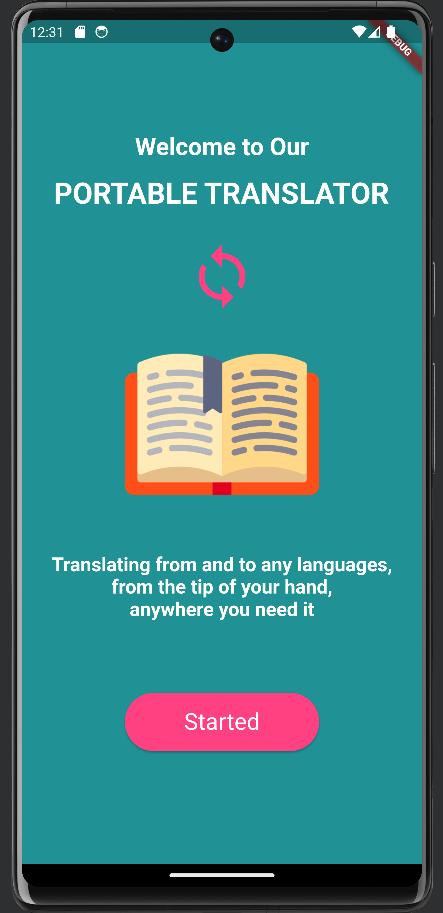
Giao diện dưới đây là trang chào mừng của ứng dụng “Portable Translator” khi người dùng khởi động ứng dụng, giao diện sẽ hiển thị một màn hình đơn giản với các nội dung sau:

Với tiêu đề hiển thị hai đoạn văn bản “Welcome to Our” và “PORTABLE TRANSLATOR” để chào mừng người sử dụng.

Biểu tượng icon dịch màu hồng với ý nghĩa có thể dịch nhanh chóng và hiệu quả của ứng dụng, ngoài ra có thể dịch hai chiều, ví dụ dịch tiếng việt sang tiếng anh và dịch tiếng anh sang tiếng việt.

Tiếp theo là một đoạn text mô tả với ý nghĩa giới thiệu ứng dụng là một công cụ dịch văn bản linh hoạt, cho phép người sử dụng dịch giữa các ngôn ngữ một cách dễ dàng và tiện lợi và người dùng có thể sử dụng bất cứ nơi đâu.

Ngoài ra giao diện có nút “Started” dưới dạng ElevatedButton, mục đích để khuyến khích người dùng bắt đầu sử dụng ứng dụng. Khi người dùng nhấn vào nút này, ứng dụng sẽ lập tức chuyển đến giao diện Text Translator.

****

Hình 3. 1.Kết quả giao diện Portable Translator

## **3.2.Giao diện Text Translator**

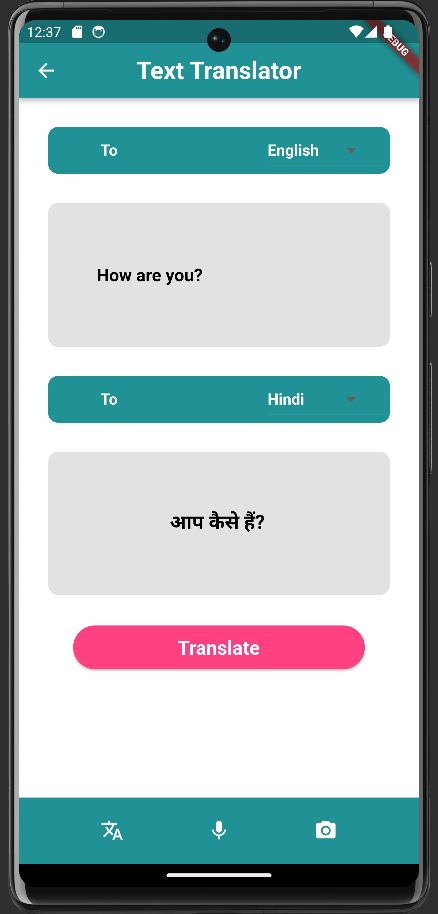
Giao diện dưới là ứng dụng dịch văn bản với những nội dung sau:

Giao diện bao gồm một AppBar có tiêu đề “Text Translator”. Phần body của ứng dụng được đặt trong SingleChildScrollView với mục đích cho phép cuộn nếu nội dung của ứng dụng quá dài.

Giao diện có hai phần chính để người dùng tương tác. Đầu tiên cho phép người dùng chọn ngôn ngữ từ đâu (From) và ngôn ngữ được dịch (To) bằng cách chọn DropdownButton. Ngôn ngữ được dùng trong ứng dụng bao gồm: Tiếng Việt, Tiếng Anh, Tiếng Hindi,Tiếng Ả Rập, Tiếng Đức, Tiếng Nga, Tiếng Tây Ban Nha, Tiếng Urdu, Tiếng Nhật và Tiếng Ý. Khi người dùng chọn ngôn ngữ, mã ngôn ngữ tương ứng sẽ được lưu trữ để sử dụng trong quá trình dịch.

Sau khi người dùng chọn ngôn ngữ và nhập văn bản, ta có thể nhấn vào nút “Translate” để thực hiện quá trình dịch. Khi người dùng nhấn nút này, ứng dụng sẽ gọi hàm translate(), lúc này nó sẽ sử dụng gói translator của flutter để dịch văn bản từ ngôn ngữ ban đầu sang ngôn ngữ đích. Kết quả dịch sẽ được hiển thị trong một Container có nền màu xám nhạt.

Ngoài ra, footer chứa ba biểu tượng, biểu tượng đầu tiên là một biểu tượng dịch văn bản, biểu tượng thứ hai là biểu tượng dịch bằng giọng nói và biểu tượng thứ ba là biểu tượng dịch bằng máy ảnh. Khi người sử dụng nhấn vào biểu tượng dịch văn bản thì sẽ xuất hiện giao diện Text Translator, nhấn vào biểu tượng giọng nói thì sẽ xuất hiện giao diện trang Micro Translator và tương tự như biểu tượng dịch bằng máy ảnh thì sẽ xuất hiện giao diện trang Camera Translator.

****

Hình 3. 2.Kết quả giao diện Text Translator

## **3.3.Giao diện Audio Translator**

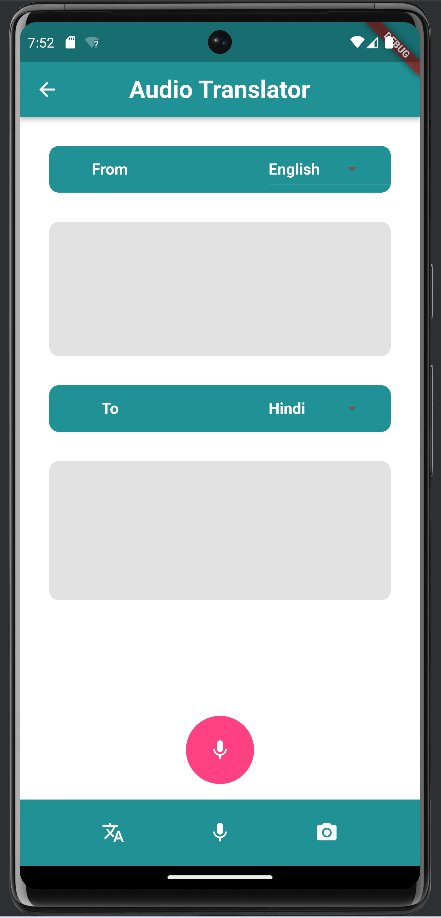
Mục đích của giao diện với chức năng dịch giọng nói sang các ngôn ngữ khác nhau. Tuy nhiên hiện tại nhóm chỉ mới xây dựng được giao diện chưa thực hiện được chức năng. Giao diện của ứng dụng bao gồm:

Widget AppBar hiển thị với tiêu đề “Audio Translator” được để ở giữa và có màu nền là màu xanh.

Phần body có nội dung sau:

Một ô chứa DropdownButton với chữ From có ý nghĩa để chọn ngôn ngữ nguồn là ngôn ngữ được dịch. Tiếp theo ô văn bản để người sử dụng nhập văn bản cần dịch. Ô chứa DropdownButton thứ hai với chữ To với mục đích để chọn ngôn ngữ đích. Ô văn bản thứ hai là ô hiển thị kết quả dịch.

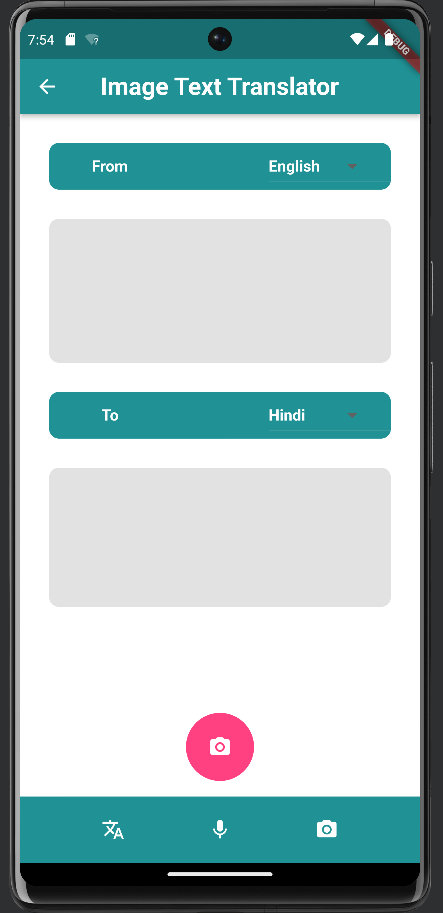
BottomNavigationBar hiển thị các biểu tượng cho phép người sử dụng có thể chuyển đổi các chức năng khác nhau của ứng dụng, bao gồm biểu tượng dịch văn bản, biểu tượng dịch bằng giọng nói và biểu tượng dịch bằng máy ảnh.

****

Hình 3. 3.Kết quả giao diện Audio Translator

## **3.4.Giao diện Image Text Translator**

Giao diện và ý nghĩa giống như Audio Translator. Nhóm em chưa thực hiện được vấn đề dịch bằng hình ảnh, mới dùng ở mức xây dựng giao diện.

****

Hình 3. 4.Kết quả giao diện Image Text Translator

# **CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

## **4.1.Kết luận**

Qua quá trình thực hiện đồ án môn học, nhóm em đã thu được một số kết quả sau:

* Về mặt kiến thức:
* Nắm được kiến thức tổng quan về ngôn ngữ Dart.
* Nắm được kiến thức tổng quan, tinh nắng, kiến trúc Flutter.
* Nắm được kiến thức tổng quan về phần mềm Android Studio.
* Nắm được khái niệm, cách sử dụng các widget trong Flutter, các bước bố trí các widget trong ứng dụng.
* Về mặt sản phẩm:
* Xây dựng hoàn thành được ứng dụng “Tên tiếng anh của bạn là gì ?”
* Thiết kế giao diện.
* Thiết kế widget cho ứng dụng.

## **4.2.Hướng phát triển**

Nhóm dự định trong tương lại sẽ hoàn thiện và phát triển ứng dụng tốt hơn với các nội dung như sau:

* Hoàn thiện giao diện lại tốt hơn.
* Sử dụng API.
* Kết nối cơ sở dữ liệu.
* Thêm chức năng dịch bằng giọng nói và máy ảnh.
* Tối ưu tốc độ cho việc dịch ngôn ngữ này sang ngôn ngữ khác.

# **TÀI LIỆU THAM KHẢO**

**Tài liệu tiếng việt**

[1] **Đại học FPT**, “Lập trình Android cơ bản”, 2014.

**Tài liệu tiếng anh**

[1] **Mark Clow**, “Learn Google Flutter Fast 65 Example Apps”, 2019.

[2] **Aakanksha Tashidar, Nisha Shah, Rushabh Gala, Trishul Giri, Pranali Chavhan**, “Application Development using Flutter”, 2020.

[3] **Eric Windmill**, “ Flutter in Action”, 2019.

**Website**

[1] Ngôn ngữ Dart

<https://dart.dev/>

[2] Flutter

<https://docs.flutter.dev/>

[3] 200lab.io

<https://200lab.io/khoa-hoc>

[4] Flutterawesome

<https://flutterawesome.com/>

[5] Flaticon

<https://www.flaticon.com/search?word=Research%20scope>

[6] Freepik

<https://www.freepik.com/>

[7] Wikipedi

<https://vi.wikipedia.org/wiki/Trang_Ch%C3%ADnh>