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

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



**BÁO CÁO MÔN HỌC**

**LẬP TRÌNH DI ĐỘNG**

**CHUYÊN NGÀNH : CÔNG NGHỆ PHẦN MỀM**

**XÂY DỰNG ỨNG DỤNG DỊCH THUẬT**

Giảng viên hướng dẫn: **Th.S Nguyễn Thanh Truyền**

Sinh viên thực hiện: **Nguyễn Hoàng Huy 0850080023**

**Nguyễn Lê Quỳnh Anh 0850080059**

**Nguyễn Thị Ngọc Hiền 0850080072**

**Châu Tú Hoa 0850080073**

**Nguyễn Lam 0850080081**

**Nguyễn Phạm Phương Quỳnh 0850080100**

**Nguyễn Trường Thịnh 0850080105**

Lớp : **08\_ĐH\_CNPM**

Khóa : **08**

***TP. Hồ Chí Minh, Tháng 8 năm 2023***

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

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



**BÁO CÁO MÔN HỌC**

**LẬP TRÌNH DI ĐỘNG**

**CHUYÊN NGÀNH : CÔNG NGHỆ PHẦN MỀM**

**XÂY DỰNG ỨNG DỤNG DỊCH THUẬT**

Giảng viên hướng dẫn: **Th.S Nguyễn Thanh Truyền**

Sinh viên thực hiện: **Nguyễn Hoàng Huy 0850080023**

**Nguyễn Lê Quỳnh Anh 0850080059**

**Nguyễn Thị Ngọc Hiền 0850080072**

**Châu Tú Hoa 0850080073**

**Nguyễn Lam 0850080081**

**Nguyễn Phạm Phương Quỳnh 0850080100**

**Nguyễn Trường Thịnh 0850080105**

Lớp : **08\_ĐH\_CNPM**

Khóa : **08**

***TP. Hồ Chí Minh, Tháng 8 năm 2023***

**MỞ ĐẦU**

"Xây dựng Ứng Dụng Dịch Thuật” là một đề tài được quan tâm và có ứng dụng rộng rãi trong thời hiện đại hiện nay. Với sự phát triển nhanh chóng của công nghệ và sự kết nối toàn cầu, nhu cầu giao tiếp và trao đổi thông tin qua các ngôn ngữ đã trở nên ngày càng phổ biến.

Trong tình hình ngày nay, khi việc kết nối và tương tác qua các nước trở nên phổ biến, ứng dụng chuyển đổi ngôn ngữ đóng vai trò quan trọng trong việc giúp con người vượt qua rào cản ngôn ngữ và thúc đẩy sự đa dạng văn hóa. Điều này có thể thúc đẩy hơn nữa qua việc giao tiếp, hợp tác và giao dịch toàn cầu.

Chúng em xây dựng Ứng Dụng Dịch Thuật này sẽ giúp đem lại lợi ích cho cộng đồng người dùng, hỗ trợ giao tiếp và trao đổi thông tin một cách linh hoạt và dễ dàng. Chúng em hy vọng rằng ứng dụng này sẽ tiếp thêm động lực cho các nghiên cứu và phát triển trong lĩnh vực ngôn ngữ và công nghệ thông tin, giúp đẩy mạnh sự đa dạng văn hóa và sự kết nối toàn cầu.

Hy vọng báo cáo này sẽ mang lại những thông tin hữu ích và cung cấp cái nhìn tổng quan về Ứng Dụng Dịch Thuật trong thực tế. Chúng em cũng hy vọng rằng báo cáo này sẽ tiếp thêm động lực và cung cấp giá trị cho các nghiên cứu và phát triển liên quan đến ngôn ngữ và công nghệ thông tin.

**LỜI CẢM ƠN**

Chúng em xin gửi lời cảm ơn chân thành đến Khoa Hệ Thống Thông Tin và Viễn Thám Trường Đại Học Tài Nguyên và Môi Trường TPHCM đã và đang tạo điều kiện thuận lợi cho chúng em học tập và hoàn thành đề tài báo cáo này. Đặc biệt, chúng em xin bày tỏ lòng biết ơn sâu sắc đến thầy ThS. Nguyễn Thanh truyền đã truyền đạt kiến thức và hướng dẫn chúng em trong quá trình nghiên cứu bài đồ án này.

Chúng em đã cố gắng vận dụng những kiến thức đã học được trong những buổi học qua để hoàn thành bài nghiên cứu này. Và do kiến thức của hạn chế, cũng như không có nhiều kinh nghiệm thực tiễn nên khó tránh khỏi những thiếu sót trong quá trình nghiên cứu và trình bày. Rất kính mong sự góp ý của quý thầy để bài báo cáo đồ án của chúng em được hoàn thiện và chỉnh chu hơn.

Một lần nữa, chúng em xin trân trọng cảm ơn sự quan tâm giúp đỡ của các thầy đã giúp đỡ trong quá trình thực hiện bài báo cáo nghiên cứu này.

Chúng em xin trân trọng cảm ơn!

**NHẬN XÉT**

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

|  |
| --- |
| **Điểm** |

MỤC LỤC

**[CHƯƠNG 1: TỔNG QUAN 1](#_Toc30660)**

**[1.1. Lý do chọn đề tài 1](#_Toc10469)**

**[1.2. Phạm vi nghiên cứu 1](#_Toc2486)**

**[1.3. Tính khoa học 2](#_Toc23837)**

**[1.4. Tính ứng dụng và thực tiễn 2](#_Toc18951)**

**[1.5. Tính hiệu quả 3](#_Toc13270)**

**[CHƯƠNG 2: CƠ SỞ LÝ LUẬN 4](#_Toc29424)**

**[2.1. Cơ sở lý thuyết 4](#_Toc21886)**

**[2.1.1. Vấn đề đặt ra 4](#_Toc19952)**

**[2.1.2. Mô tả ứng dụng 4](#_Toc17721)**

**[2.2. Cơ sở kỹ thuật 5](#_Toc15029)**

**[2.2.1. Giới thiệu về Android Studio 5](#_Toc4265)**

**[2.2.1.1. Khái niệm 5](#_Toc21057)**

**[2.2.1.2. Đặc điểm 6](#_Toc20242)**

**[2.2.1.3. Vai trò 6](#_Toc12291)**

**[2.2.2. Giới thiệu về Flutter 6](#_Toc29299)**

**[2.2.2.1. Khái niệm 6](#_Toc30984)**

**[2.2.2.2. Đặc điểm 7](#_Toc17033)**

**[2.2.2.3. Vai trò 7](#_Toc14708)**

**[2.2.3. Giới thiệu về Dart 8](#_Toc22750)**

**[2.2.3.1. Khái niệm 8](#_Toc13010)**

**[2.2.3.2. Đặc điểm 9](#_Toc28235)**

**[2.2.3.3. Vai trò 9](#_Toc18961)**

**[2.2.4. Giới thiệu về SDK 9](#_Toc3483)**

**[2.2.4.1. Khái niệm 9](#_Toc31260)**

**[2.2.4.2. Đặc điểm 10](#_Toc7214)**

**[2.2.4.3. Vai trò 11](#_Toc6250)**

**[2.3. Nội dung lý thuyết 11](#_Toc28877)**

**[2.3.1. Tìm hiểu nội dung Container 11](#_Toc25354)**

**[2.3.2. Tìm hiểu nội dung Textfeild 16](#_Toc12820)**

**[2.3.2. Tìm hiểu nội dung Button 18](#_Toc19431)**

**[2.3.4. Tìm hiểu nội dung Center 41](#_Toc16656)**

**[2.3.5. Tìm hiểu nội dung Media Query 43](#_Toc21340)**

**[2.3.6. Tìm hiểu nội dung ListView 45](#_Toc23057)**

**[2.3.7. Tìm hiểu nội dung TableData 49](#_Toc5209)**

**[2.3.8. Tìm hiểu nội dung GirdView 50](#_Toc9402)**

**[CHƯƠNG 3: CÀI ĐẶT VÀ THỬ NGHIỆM 59](#_Toc30365)**

**[3.1. Giao diện 59](#_Toc30160)**

**[3.2. Danh sách các ngôn ngữ trong ứng dụng 60](#_Toc24532)**

**[3.3. Dịch thuật 61](#_Toc22625)**

**[3.4 Dịch thuật bằng giọng nói 71](#_Toc17053)**

**[CHƯƠNG 4: TỔNG KẾT 72](#_Toc24882)**

**[4.1. Kết quả đạt được 72](#_Toc8897)**

**[4.2. Hạn chế 72](#_Toc9885)**

**[4.3. Hướng phát triển 72](#_Toc8226)**

**[4.4. Kết luận 72](#_Toc18489)**

**[TÀI LIỆU THAM KHẢO 73](#_Toc26159)**

# CHƯƠNG 1: TỔNG QUAN

* 1. **Lý do chọn đề tài**

Lý do chọn đề tài “Xây Dựng Ứng Dụng Dịch Thuật” là do nhận thấy nhu cầu giao tiếp và trao đổi thông tin qua các ngôn ngữ khác nhau đang ngày càng phổ biến và cần thiết trong thời hiện đại. Việc sử dụng ứng dụng này giúp giảm bớt rào cản ngôn ngữ và hỗ trợ con người trong việc truyền tải ý nghĩa chính xác và hiệu quả trong môi trường quốc tế.

Cùng với sự phát triển nhanh chóng của công nghệ và Internet, việc giao tiếp của con người có thể vượt qua ranh giới địa lý cách biệt. Điều này đặt ra yêu cầu về việc chuyển đổi sang ngôn ngữ khác nhau một cách nhanh chóng và tiện lợi là nhu cầu cấp thiết là vô cùng quan trọng.

Nhằm để người dùng có thể tương tác và giao tiếp một cách trôi chảy và tự nhiên. Mục tiêu của Ứng Dụng Dịch Thuật sẽ là một công cụ hiệu quả và tin cậy cho người dùng, giúp họ vượt qua rào cản ngôn ngữ, tạo cơ hội tốt hơn trong việc mở rộng mạng lưới xã hội và kinh doanh, đồng thời tạo ra giá trị và thuận lợi cho cộng đồng. Đây là một đề tài hứa hẹn đem lại lợi ích thực tiễn và có ảnh hưởng tích cực đối với người dùng trong thời hiện đại của sự toàn cầu hóa.

* 1. **Phạm vi nghiên cứu**

Xây dựng ứng dụng di động: Sử dụng Android Studio và Flutter để xây dựng ứng dụng có khả năng hoạt động trên hệ điều hành Android. Việc sử dụng Flutter cho phép ứng dụng có tính di động cao và hiệu năng tốt trên nhiều thiết bị di động khác nhau.

Giao diện người dùng thân thiện: Thiết kế giao diện đơn giản, dễ sử dụng và thân thiện với người dùng để họ có thể dễ dàng thao tác và chuyển đổi tên một cách nhanh chóng.

Tích hợp Internet: Ứng dụng sẽ kết nối với Internet để truy cập cũng như sử dụng các dịch vụ liên quan đến ngôn ngữ.

Thử nghiệm và đánh giá: Tiến hành các thử nghiệm và đánh giá để đảm bảo tính đúng đắn và hiệu suất của ứng dụng. Điều này đòi hỏi thử nghiệm với nhiều trường hợp dữ liệu và tình huống để đảm bảo khả năng hoạt động tốt trong mọi trường hợp.

* 1. **Tính khoa học**

Đề tài “Xây Dựng Ứng Dụng Dịch Thuật” bằng Android Studio với Flutte mang tính khoa học cao và đáng quan tâm trong lĩnh vực xử lý ngôn ngữ tự nhiên và ứng dụng di động.

Tối ưu hiệu năng ứng dụng: Việc sử dụng Flutter trong xây dựng ứng dụng giúp đạt được hiệu năng cao và đáng tin cậy trên nhiều thiết bị di động khác nhau. Chúng ta sẽ tập trung nghiên cứu và tối ưu mã nguồn, cấu trúc dữ liệu và thuật toán để đảm bảo ứng dụng hoạt động mượt mà và nhanh chóng trong mọi trường hợp sử dụng.

Mở rộng và ứng dụng thực tế: Đề tài này không chỉ giới hạn ở việc xây dựng một ứng dụng cơ bản, mà còn tập trung vào khả năng mở rộng và ứng dụng thực tế. Chúng ta sẽ xem xét các khía cạnh liên quan đến việc ứng dụng này có thể được sử dụng trong thực tế và các lĩnh vực ứng dụng rộng rãi, từ cải thiện trải nghiệm người dùng đối với các ứng dụng hiện có đến hỗ trợ các doanh nghiệp định hình thương hiệu quốc tế.

* 1. **Tính ứng dụng và thực tiễn**

Tính ứng dụng và tiện ích: Ứng dụng này cung cấp một công cụ hữu ích cho người dùng, giúp họ dễ dàng thay đổi tên cá nhân, địa danh hoặc thương hiệu từ Tiếng Việt sang ngôn ngữ khác một cách chính xác và hiệu quả. Với giao diện thân thiện và dễ sử dụng, người dùng có thể thực hiện việc chuyển đổi ngôn ngữ một cách trực quan và thuận tiện.

Khả năng giải quyết rào cản ngôn ngữ: Ứng dụng này giải quyết một trong những rào cản quan trọng khi giao tiếp và trao đổi thông tin trên môi trường quốc tế, đó là ngôn ngữ. Việc dịch thuật từ Tiếng Việt sang ngôn ngữ khác giúp người dùng vượt qua rào cản ngôn ngữ và tạo điều kiện thuận lợi cho giao tiếp và kết nối với mọi người trên khắp thế giới.

Tích hợp Internet và di động: Ứng dụng kết nối với Internet để truy cập và cập nhật bộ từ điển, đồng thời hỗ trợ di động với tính di động cao và hiệu năng tốt trên nhiều thiết bị di động khác nhau.

Ứng dụng thực tế và tiềm năng đa dạng: Ứng dụng này có tính ứng dụng rộng rãi và thực tiễn, từ việc hỗ trợ cá nhân trong việc chuyển đổi tên cá nhân, địa danh cho đến việc hỗ trợ doanh nghiệp trong việc thương mại quốc tế và quảng bá thương hiệu. Đây là một giải pháp đáng giá cho mọi người trong thời đại của sự toàn cầu hóa và kết nối toàn cầu.

Tiềm năng ứng dụng trong nhiều lĩnh vực: Khả năng mở rộng và phát triển của ứng dụng này cho phép ứng dụng trong nhiều lĩnh vực khác nhau, từ giáo dục, du lịch, thương mại đến truyền thông và trao đổi văn hóa.

* 1. **Tính hiệu quả**
* Tính nhanh chóng và chính xác: Ứng dụng được thiết kế để dịch thuật một cách nhanh chóng và chính xác, giúp tiết kiệm thời gian và tăng hiệu suất trong quá trình giao tiếp và trao đổi thông tin.
* Hiệu suất cao và di động: Ứng dụng được xây dựng bằng Flutter, giúp tăng cường hiệu suất và tính di động trên nhiều thiết bị di động khác nhau, đồng thời giảm thiểu việc tiêu tốn tài nguyên của thiết bị.
* Giao diện người dùng thân thiện: Thiết kế giao diện đơn giản, dễ sử dụng và thân thiện giúp người dùng dễ dàng thao tác và chuyển đổi tên một cách tự nhiên và trực quan.
* Tích hợp Internet và bộ từ điển đa dạng: Ứng dụng kết nối với Internet để truy cập và cập nhật bộ từ điển đa dạng, giúp đề xuất các từ tương đương chính xác trong Tiếng Anh và các ngôn ngữ khác dựa trên dữ liệu thực tế và phong phú.

# CHƯƠNG 2: CƠ SỞ LÝ LUẬN

## 2.1. Cơ sở lý thuyết

### 2.1.1. Vấn đề đặt ra

Trong thời đại của sự toàn cầu hóa và liên kết toàn cầu, việc giao tiếp và trao đổi thông tin trở nên ngày càng quan trọng. Tuy nhiên, mỗi quốc gia và vùng lãnh thổ đều có ngôn ngữ và chữ viết riêng, tạo ra rào cản trong việc truyền tải thông tin giữa các người dùng có ngôn ngữ khác nhau. Đặc biệt, việc dịch thuật có thể gặp nhiều khó khăn và thách thức.

**Một số vấn đề đặt ra của đề tài này bao gồm:**

* *Khó khăn trong việc chuyển đổi tên đa dạng:* Tên người, địa danh và thương hiệu thường đa dạng và phong phú, đòi hỏi một bộ từ điển lớn và đáng tin cậy để đảm bảo tính chính xác trong quá trình chuyển đổi.
* *Cần thiết phải có giải pháp hiệu quả và nhanh chóng:* Trong nhiều trường hợp, người dùng cần thực hiện chuyển đổi tên một cách nhanh chóng và hiệu quả, đặc biệt khi cần sử dụng trong các hoạt động giao tiếp, kinh doanh hoặc du lịch.
* *Bảo mật thông tin người dùng:* Trong quá trình chuyển đổi tên, đảm bảo bảo mật thông tin người dùng là một yếu tố quan trọng để tránh việc rò rỉ thông tin cá nhân và dữ liệu nhạy cảm.
* *Hiệu suất và tính di động:* Ứng dụng cần đáp ứng yêu cầu về hiệu suất và tính di động, đảm bảo hoạt động mượt mà và đáng tin cậy trên nhiều thiết bị di động khác nhau.
* *Khả năng mở rộng và phát triển:* Đề tài cần tập trung vào việc xây dựng một ứng dụng có khả năng mở rộng và phát triển trong tương lai, để cung cấp các tính năng và bổ sung từ điển mới theo nhu cầu của người dùng.

### 2.1.2. Mô tả ứng dụng

Dưới đây là mô tả về tính năng và trải nghiệm của ứng dụng:

* ***Giao diện thân thiện và dễ sử dụng:***

+ Ứng dụng có giao diện người dùng thân thiện và dễ sử dụng, giúp người dùng thao tác một cách trực quan và tự nhiên.

+ Các phần tử giao diện được thiết kế rõ ràng, dễ nhìn và dễ nhận biết, giúp người dùng tìm hiểu và sử dụng ứng dụng một cách thuận tiện.

* ***Tính nhanh chóng và hiệu quả:*** Ứng dụng có tốc độ dịch thuật nhanh chóng và hiệu quả trong quá trình chuyển đổi ngôn ngữ.
* ***Tích hợp Internet và tính di động:*** Ứng dụng kết nối với Internet để truy cập và cập nhật bộ từ điển đa dạng, đồng thời hỗ trợ di động với tính di động cao và hiệu năng tốt trên nhiều thiết bị di động khác nhau.
* ***Dịch thuật bằng giọng nói:*** Ứng dụng có thể nhận dạng giọng nói khi người dùng chọn micro và nói, văn bản sẽ tự động điền vào phần nhập văn bản. Sau đó người dùng cần chọn ngôn ngữ cần dịch và bấm dịch thì đoạn văn bản bên dưới sẽ hiện ra phần văn bản sau khi đã dịch.

## 2.2. Cơ sở kỹ thuật

### 2.2.1. Giới thiệu về Android Studio

#### 2.2.1.1. Khái niệm

Android Studio là môi trường phát triển tích hợp (IDE) chính thức được phát triển bởi Google, được sử dụng để phát triển ứng dụng di động chạy trên hệ điều hành Android. Nó được xây dựng trên nền tảng IntelliJ IDEA của JetBrains và cung cấp một loạt công cụ mạnh mẽ để phát triển, thử nghiệm, và triển khai các ứng dụng Android.

Android Studio cung cấp môi trường phát triển tích hợp đồ họa cho lập trình viên Android, giúp họ dễ dàng tạo, thiết kế và xây dựng ứng dụng Android. Nó hỗ trợ nhiều ngôn ngữ lập trình như Java, Kotlin và C++, cho phép lập trình viên lựa chọn ngôn ngữ ưa thích để phát triển ứng dụng.



Hình 2. 1 Androi Studio

#### 2.2.1.2. Đặc điểm

* *Công cụ thiết kế giao diện đồ họa (Layout Editor):* Cho phép lập trình viên thiết kế giao diện người dùng bằng cách kéo và thả các thành phần UI một cách trực quan, giúp tiết kiệm thời gian và tăng hiệu suất làm việc.
* *Trình giả lập Android (Android Emulator):* Android Studio đi kèm với một trình giả lập Android tích hợp, cho phép lập trình viên kiểm tra và chạy ứng dụng trên nhiều thiết bị ảo với nhiều phiên bản hệ điều hành Android khác nhau.
* *Quản lý phiên bản (Version Control):* Hỗ trợ tích hợp các hệ thống quản lý phiên bản như Git, giúp lập trình viên quản lý mã nguồn của ứng dụng một cách hiệu quả.
* *Hỗ trợ ngôn ngữ đa dạng:* Android Studio hỗ trợ nhiều ngôn ngữ lập trình như Java, Kotlin và C++, cho phép lập trình viên lựa chọn ngôn ngữ phù hợp với nhu cầu và sở thích cá nhân.

#### 2.2.1.3. Vai trò

Trong đề tài "Xây Dựng Ứng Dụng Dịch Thuật” thuật bằng Android Studio với Flutter, Android Studio đóng vai trò là môi trường phát triển chính để xây dựng ứng dụng di động như mong muốn. Nó cung cấp các công cụ và tính năng mạnh mẽ cho lập trình viên, giúp họ thiết kế giao diện người dùng, xử lý logic, và triển khai ứng dụng trên nền tảng Android.

Android Studio sẽ hỗ trợ việc phát triển ứng dụng dịch thuật bằng cách cung cấp trình giả lập Android cho việc thử nghiệm ứng dụng trên các thiết bị ảo, trình biên dịch và kiểm tra mã nguồn, cũng như tích hợp các công cụ hỗ trợ phân tích và sửa lỗi.

### 2.2.2. Giới thiệu về Flutter

#### 2.2.2.1. Khái niệm

Flutter là một framework mã nguồn mở được phát triển bởi Google, dùng để xây dựng giao diện người dùng đẹp và đa nền tảng cho các ứng dụng di động, web và máy tính bảng. Nó ra mắt lần đầu vào năm 2017 và đã nhanh chóng trở thành một công cụ phổ biến cho các nhà phát triển ứng dụng di động. Flutter cho phép bạn tạo các ứng dụng chất lượng cao với hiệu suất tốt và giao diện người dùng linh hoạt.

Flutter được xây dựng dựa trên ngôn ngữ lập trình Dart, cũng là ngôn ngữ mà Google đã phát triển. Điểm mạnh của Flutter nằm ở việc nó sử dụng mô hình lập trình khác so với các framework truyền thống. Thay vì sử dụng cách tiếp cận dựa trên cây như các framework khác, Flutter sử dụng mô hình lập trình có trạng thái (stateful) và vô trạng thái (stateless) để quản lý giao diện người dùng.



Hình 2. 2 Flutter

#### 2.2.2.2. Đặc điểm

* *Hot Reload:* Flutter hỗ trợ hot reload, cho phép bạn thay đổi mã nguồn và xem kết quả ngay lập tức trên trình giả lập hoặc thiết bị di động mà không cần khởi động lại ứng dụng. Điều này giúp giảm thời gian phát triển và tăng năng suất.
* *Giao diện người dùng đẹp và đồng nhất:* Flutter cho phép bạn tạo giao diện người dùng đẹp và đồng nhất trên các nền tảng khác nhau mà không cần viết lại mã nguồn nhiều lần. Giao diện được xây dựng bằng các widget độc lập và có thể tái sử dụng.
* *Hiệu suất cao:* Flutter sử dụng động cơ chuyển đổi (AOT) và động cơ nạp trước (JIT) để tối ưu hóa hiệu suất của ứng dụng. Điều này giúp các ứng dụng Flutter chạy nhanh và mượt mà.
* *Hỗ trợ đa nền tảng:* Flutter hỗ trợ nhiều nền tảng bao gồm Android, iOS, web và máy tính bảng. Điều này giúp bạn phát triển một lần và triển khai ứng dụng của mình trên nhiều nền tảng một cách dễ dàng.

#### 2.2.2.3. Vai trò

Flutter đóng vai trò quan trọng trong đề tài "Xây dựng ứng dụng dịch thuật" Nó là công cụ chính để xây dựng ứng dụng di động, cung cấp các tính năng mạnh mẽ và linh hoạt để tạo giao diện người dùng đẹp và hiệu quả.

Với Flutter, bạn có thể xây dựng giao diện người dùng phong phú, thân thiện và có khả năng tương tác cao. Flutter cũng cho phép bạn triển khai ứng dụng lên nhiều nền tảng một cách dễ dàng, giúp đảm bảo rằng ứng dụng của bạn có thể hoạt động trên nhiều thiết bị và hệ điều hành khác nhau.

Đồng thời, khả năng hot reload của Flutter sẽ hỗ trợ lập trình viên trong việc thử nghiệm và cập nhật ứng dụng nhanh chóng, giúp giảm thời gian phát triển và tối ưu hóa quá trình làm việc. Tích hợp Android Studio với Flutter cũng cung cấp môi trường phát triển tích hợp đồ họa mạnh mẽ, giúp bạn tối ưu hóa quá trình phát triển ứng dụng và giảm thiểu lỗi trong quá trình xây dựng.

### 2.2.3. ****Giới thiệu về Dart****

#### 2.2.3.1. Khái niệm

Dart là một ngôn ngữ lập trình mã nguồn mở được phát triển bởi Google. Nó được giới thiệu lần đầu vào năm 2011 và được sử dụng rộng rãi trong việc phát triển ứng dụng di động và web. Dart được thiết kế để đơn giản, linh hoạt và dễ dàng để học, và nó là ngôn ngữ chính của Flutter - framework phát triển ứng dụng di động đa nền tảng của Google.

Khái niệm:

Dart là một ngôn ngữ lập trình hướng đối tượng, hỗ trợ cả tính năng lập trình cấu trúc và lập trình hàm. Nó sử dụng cú pháp tương tự như nhiều ngôn ngữ lập trình phổ biến khác như C++, Java và JavaScript, giúp lập trình viên có thể dễ dàng chuyển đổi và học hỏi.

Dart cũng hỗ trợ việc biên dịch trực tiếp (JIT) và biên dịch trước (AOT), giúp tối ưu hóa hiệu suất ứng dụng. Nó hỗ trợ mạnh mẽ việc kiểm tra kiểu tĩnh, giúp ngăn chặn lỗi tại thời điểm biên dịch và cải thiện tính tin cậy của ứng dụng.



Hình 2. 3 Dart

#### 2.2.3.2. Đặc điểm

* ***Dễ học và sử dụng:*** Dart có cú pháp rõ ràng và dễ hiểu, giúp lập trình viên mới dễ dàng tiếp cận và học hỏi.
* ***Hỗ trợ lập trình hướng đối tượng:*** Dart hỗ trợ lập trình hướng đối tượng với tính năng như class, object, inheritance, interface, v.v.
* Kiểu dữ liệu mạnh mẽ: Dart hỗ trợ kiểu dữ liệu mạnh mẽ và kiểm tra kiểu tĩnh, giúp ngăn chặn lỗi liên quan đến kiểu dữ liệu và cải thiện tính tin cậy của mã nguồn.
* ***Tích hợp bộ nhớ rác (Garbage Collection):*** Dart có bộ nhớ rác tích hợp tự động thu thập các đối tượng không còn sử dụng, giúp giảm thời gian lập trình và tránh lỗi xử lý bộ nhớ.
* ***Hỗ trợ biên dịch trực tiếp và biên dịch trước:*** Dart hỗ trợ biên dịch trực tiếp (JIT) trong quá trình phát triển, giúp giảm thời gian phát triển và cải thiện trải nghiệm thử nghiệm người dùng. Biên dịch trước (AOT) sẽ cải thiện hiệu suất của ứng dụng khi triển khai.

#### 2.2.3.3. Vai trò

Dart đóng vai trò là ngôn ngữ chính được sử dụng trong đề tài "Xây dựng ứng dụng dịch thuật" Nó là ngôn ngữ lập trình được Flutter sử dụng để xây dựng ứng dụng di động đa nền tảng. Với tính năng dễ học, cú pháp rõ ràng và hỗ trợ lập trình hướng đối tượng, Dart giúp lập trình viên tạo ra mã nguồn dễ đọc, dễ bảo trì và dễ phát triển.

Dart cũng hỗ trợ kiểm tra kiểu tĩnh và có tích hợp bộ nhớ rác, giúp giảm thiểu lỗi và cải thiện hiệu suất ứng dụng. Với việc tích hợp JIT và AOT, Dart cung cấp môi trường lập trình mạnh mẽ cho phát triển ứng dụng di động đa nền tảng với Flutter.

### 2.2.4. ****Giới thiệu về SDK****

#### 2.2.4.1. Khái niệm

SDK (Software Development Kit) là một bộ công cụ phát triển phần mềm dùng để hỗ trợ lập trình viên trong việc phát triển ứng dụng, ứng dụng di động, trò chơi, hoặc các sản phẩm phần mềm khác. SDK chứa các thư viện, các công cụ biên dịch, trình biên dịch, trình tạo mã nguồn và tài liệu hướng dẫn. SDK giúp tạo ra môi trường lập trình hoàn chỉnh để xây dựng phần mềm dễ dàng và hiệu quả.

SDK cung cấp môi trường lập trình hoàn chỉnh cho lập trình viên, chứa các tài nguyên cần thiết để phát triển và xây dựng phần mềm. Nó bao gồm các thành phần như trình biên dịch, trình tạo mã nguồn, các thư viện, các trình giả lập, tài liệu hướng dẫn và các công cụ hỗ trợ phát triển khác.

SDK thường được phân phối miễn phí hoặc có thể có phiên bản trả phí với tính năng bổ sung. Các công ty công nghệ thường phát triển SDK để hỗ trợ các nhà phát triển bên ngoài xây dựng ứng dụng trên nền tảng của họ.



Hình 2. 4 SDK

#### 2.2.4.2. Đặc điểm

1. Thư viện và API: SDK cung cấp các thư viện và API (Application Programming Interface) để lập trình viên có thể sử dụng các chức năng có sẵn trong hệ thống hay nền tảng mà họ đang phát triển ứng dụng.
2. Trình biên dịch và trình tạo mã nguồn: SDK chứa các công cụ biên dịch mã nguồn và trình tạo mã nguồn giúp chuyển đổi mã nguồn thành mã máy và kiểm tra lỗi trước khi triển khai ứng dụng.
3. Công cụ giả lập: Một số SDK đi kèm với các công cụ giả lập để cho phép lập trình viên thử nghiệm ứng dụng trên môi trường giả lập trước khi triển khai lên thiết bị thật.
4. Tài liệu hướng dẫn và hỗ trợ: SDK đi kèm với tài liệu hướng dẫn và hỗ trợ để giúp lập trình viên hiểu rõ hơn cách sử dụng các thành phần và công cụ trong SDK.

#### 2.2.4.3. Vai trò

Trong đề tài "Xây Dựng Ứng Dụng Dịch Thuật” bằng Android Studio với Flutter, SDK đóng vai trò quan trọng. Cụ thể là:

**+ Android SDK:** Để phát triển ứng dụng di động cho hệ điều hành Android, Android SDK cung cấp các công cụ, thư viện và API cần thiết để xây dựng ứng dụng Android.

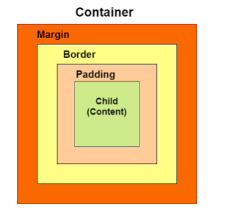
**+ Flutter SDK:** là bộ công cụ phát triển chính cho xây dựng ứng dụng di động đa nền tảng với Flutter. Nó cung cấp môi trường lập trình để tạo giao diện người dùng và xử lý logic ứng dụng với Dart.

## 2.3. Nội dung lý thuyết

### 2.3.1. Tìm hiểu nội dung Container

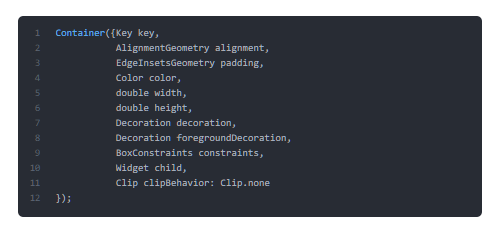
Container (Vùng chứa) trong Flutter là một widget mẹ có thể chứa nhiều widget con và quản lý chúng một cách hiệu quả thông qua chiều rộng, chiều cao, khoảng đệm, màu nền, v.v. Nó là một widget kết hợp vẽ, định vị và định cỡ thông thường của các widget con. Nó cũng là một lớp để lưu trữ một hoặc nhiều widget và định vị chúng trên màn hình theo nhu cầu của chúng ta. Nói chung, nó tương tự như một chiếc hộp để chứa nội dung. Nó cho phép nhiều thuộc tính cho người dùng để trang trí các widget con của nó, chẳng hạn như sử dụng margin , ngăn cách Container với các nội dung khác.

Một Container cơ bản có các thuộc tính margin, border và padding xung quanh widget của nó, như thể hiện trong hình ảnh bên dưới:



Hình 2. 5 Container

Sau đây là cú pháp của phương thức khởi tạo lớp Container:



Hình 2. 6 Cú pháp container

* **Thuộc tính của widget Container**

**1. Child:** Thuộc tính này được sử dụng để lưu trữ widget con của Container. Giả sử chúng ta đã sử dụng Text làm widget con của nó, có thể được hiển thị trong ví dụ dưới đây:

1. Container(
2. child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),
3. )

**2. Color:** Thuộc tính này được sử dụng để đặt màu nền của Text. Nó cũng thay đổi màu nền của toàn bộ Container. Xem ví dụ dưới đây:

Container(

color: Colors.green,

child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),

)

**3. Height and width**: Thuộc tính này được sử dụng để thiết lập chiều cao và chiều rộng của container theo nhu cầu của chúng ta. Theo mặc định, Container luôn chiếm không gian dựa trên widget của nó. Xem đoạn code dưới đây:

Container(

width: 200.0,

height: 100.0,

color: Colors.green,

child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),

)

**4. Margin**: Thuộc tính này được sử dụng để bao quanh không gian empty xung quanh Container. Chúng ta có thể quan sát điều này bằng cách nhìn thấy khoảng trắng xung quanh thùng chứa. Giả sử chúng ta đã sử dụng EdgeInsets.all (25) để đặt lề bằng nhau theo cả bốn hướng, như thể hiện trong ví dụ dưới đây:

Container(

width: 200.0,

height: 100.0,

color: Colors.green,

margin: EdgeInsets.all(20),

child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),

)

**5. Padding:** Thuộc tính này được sử dụng để đặt khoảng cách giữa đường viền của Container (cả bốn hướng) và widget con của nó. Chúng ta có thể quan sát điều này bằng cách nhìn thấy khoảng trống giữa Container và widget con. Ở đây, chúng tôi đã sử dụng EdgeInsets.all(35) để đặt khoảng cách giữa văn bản và tất cả bốn hướng Container:

1. Container(
2. width: 200.0,
3. height: 100.0,
4. color: Colors.green,
5. padding: EdgeInsets.all(35),
6. margin: EdgeInsets.all(20),
7. child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),
8. )

**6. Alignment:** Thuộc tính này được sử dụng để thiết lập vị trí của con trong Container. Flutter cho phép người dùng căn chỉnh phần tử của nó theo nhiều cách khác nhau như giữa, dưới cùng, dưới giữa, trên cùng, bên phải, bên trái, bên trái, bên phải và nhiều hơn nữa. Trong ví dụ dưới đây, chúng ta sẽ căn chỉnh con của nó vào vị trí dưới cùng bên phải.

Container(

width: 200.0,

height: 100.0,

color: Colors.green,

padding: EdgeInsets.all(35),

margin: EdgeInsets.all(20),

alignment: Alignment.bottomRight,

child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),

)

1. **Decoration:** Thuộc tính này cho phép nhà phát triển thêm trang trí trên widget. Nó trang trí hoặc vẽ các phụ tùng phía sau widget con. Nếu chúng ta muốn trang trí hoặc vẽ trước mặt một widget con, chúng ta cần sử dụng tham số forgroundDecoration. Hình ảnh dưới đây giải thích sự khác biệt giữa chúng khi phần nền phía trước Trang trí bao phủ widget con và lớp sơn trang trí phía sau widget con.

- Thuộc tính trang trí hỗ trợ nhiều tham số, chẳng hạn như màu sắc, độ dốc, hình nền, đường viền, bóng, v.v. Điều này nhằm đảm bảo rằng chúng ta có thể sử dụng thuộc tính màu trong Container hoặc trang trí, nhưng không thể sử dụng cả hai.

- Xem đoạn code dưới đây, nơi chúng tôi đã thêm thuộc tính đường viền và bóng để trang trí hộp:



Hình 2. 7 Decoration

1. **Transform:** Thuộc tính biến đổi cho phép các nhà phát triển xoay Container. Nó có thể xoay Container theo bất kỳ hướng nào, tức là thay đổi tọa độ Container trong widget. Trong ví dụ dưới đây, chúng ta sẽ xoay Container theo trục z.

Container(

width: 200.0,

height: 100.0,

color: Colors.green,

padding: EdgeInsets.all(35),

margin: EdgeInsets.all(20),

alignment: Alignment.bottomRight,

transform: Matrix4.rotationZ(0.1),

child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),

)

1. **Constraints:** Thuộc tính này được sử dụng khi chúng ta muốn thêm các ràng buộc bổ sung cho widget con. Nó chứa các hàm tạo khác nhau, chẳng hạn như chặt chẽ, lỏng lẻo, mở rộng, v.v.

Hãy xem cách sử dụng các hàm tạo này trong ứng dụng của chúng ta:

Nếu chúng ta sử dụng thuộc tính size trong này, nó sẽ cung cấp giá trị cố định cho widget con.

Container(

color: Colors.green,

constraints: BoxConstraints.tight(Size size)

: minWidth = size.width, maxWidth = size.width,

minHeight = size.height, maxHeight = size.height;

child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),

)

### 2.3.2. Tìm hiểu nội dung Textfeild

TextField là một trong những thành phần hết sức quen thuộc cho các lập trình viên mobile cho phép thu thập thông tin từ người dùng.

Code cho TextField cơ bản đơn giản như:

TextField()

Điều này tạo ra một TextField cơ bản:

A screenshot of a computer

Description automatically generated

* **Lấy thông tin từ TextField**

1. Sử dụng phương thức onChanged và lưu trữ giá trị hiện tại trong một biến đơn giản.

String value = "";

TextField(

onChanged: (text) {

value = text;

},

)

1. Sử dụng TextEditingController.

TextEditingController controller = TextEditingController();

TextField(

controller: controller,

)

Và chúng ta có thể lắng nghe những thay đổi bằng cách sử dụng

controller.addListener(() {

// Do something here

});

Và nhận hoặc đặt giá trị bằng cách sử dụng

print(controller.text); // Print current value

controller.text = "Demo Text"; // Set new value

Tiện ích TextField cũng cung cấp các callback khác như:

onEditingComplete: () {},

onSubmitted: (value) {},

Đây là các callback được gọi trên các hành động như khi người dùng nhấp vào nút “Done” button trên bàn phím iOS.

* **Tạo kiểu văn bản bên trong TextField**

TextField(

style: TextStyle(color: Colors.red, fontWeight: FontWeight.w300),

),

* **Thay đổi con trỏ trong TextField**

TextField(

cursorColor: Colors.red,

cursorRadius: Radius.circular(16.0),

cursorWidth: 16.0, ),

* **Kiểm soát ký tự tối đa**

TextField(

maxLength: 4,

),

### 2.3.2. Tìm hiểu nội dung Button

Nút (Button) là phần tử điều khiển đồ họa **cung cấp cho người dùng kích hoạt một sự kiện** như thực hiện hành động, lựa chọn, tìm kiếm mọi thứ, v.v. Chúng có thể được đặt ở bất kỳ đâu trong giao diện người dùng của chúng tôi như hộp thoại, biểu mẫu, thẻ, thanh công cụ,…Các nút là các widget Flutter, là một phần của thư viện material design. Flutter cung cấp một số loại nút có hình dạng, kiểu dáng và tính năng khác nhau.

* Các loại nút trong Flutter:
* Nút phẳng (Flat Button)
* Nút nâng (Raised Button)
* Nút nổi (Floating Button)
* Nút thả xuống (Drop Down Button)
* Nút biểu tượng (Icon Button)
* Nút Inkwell
* Nút PopupMenu
* Nút phác thảo (Outline Button)

Chi tiết từng nút sau đây:

1. **Nút phẳng(Flat Button)**

Nó là một **nút nhãn văn bản** không có nhiều trang trí và hiển thị **mà không có bất kỳ độ cao nào** . Nút phẳng có hai thuộc tính bắt buộc là: **child và onPressed()** . Nó chủ yếu được sử dụng trong các thanh công cụ, hộp thoại hoặc nội tuyến với các nội dung khác. Theo mặc định, nút phẳng không có màu và văn bản của nó là màu đen. Tuy nhiên, chúng ta có thể sử dụng màu cho nút và văn bản bằng cách sử dụng các thuộc tính **color và textColor** , tương ứng.

**Thí dụ:**

Mở tệp **main.dart** và thay thế bằng code bên dưới.

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatefulWidget {

@override

\_MyAppState createState() => \_MyAppState();

}

class \_MyAppState extends State<MyApp> {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(

title: Text('Flutter FlatButton Example'),

),

body: Center(child: Column(children: <Widget>[

Container(

margin: EdgeInsets.all(25),

child: FlatButton(

child: Text('SignUp', style: TextStyle(fontSize: 20.0),),

onPressed: () {},

),

),

Container(

margin: EdgeInsets.all(25),

child: FlatButton(

child: Text('LogIn', style: TextStyle(fontSize: 20.0),),

color: Colors.blueAccent,

textColor: Colors.white,

onPressed: () {},

),

),

]

))

),

);

}

}

* **Kết quả đầu ra: Đây là ví dụ cho button.**

A screen shot of a phone

Description automatically generated

1. **Nút nâng (Raised Button)**

Nó là một nút, dựa trên vật liệu vật liệu và có **thân hình chữ nhật** . Nó tương tự như một nút phẳng, nhưng nó **có độ cao (elevation)** sẽ tăng lên khi nút được nhấn. Nó thêm thứ nguyên cho giao diện người dùng dọc theo trục Z. Nó có một số thuộc tính như màu văn bản, hình dạng, phần đệm, màu nút, màu của nút khi bị tắt, thời gian animation, độ cao,..

Nút này có **hai chức năng gọi lại**.

**+ onPressed ():** Nó được kích hoạt khi nhấn nút.

**+ onLongPress ():** Nó được kích hoạt khi nhấn và giữ nút.

Cần lưu ý rằng nút này ở **trạng thái bị vô hiệu hóa** nếu các lệnh gọi lại onPressed() và onLongPressed() không được chỉ định.

****Ví dụ:**** Mở tệp **main.dart** và thay thế bằng code bên dưới.

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatefulWidget {

@override

\_MyAppState createState() => \_MyAppState();

}

class \_MyAppState extends State<MyApp> {

String msg = 'Flutter RaisedButton Example';

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(

title: Text('Flutter RaisedButton Example'),

),

body: Container(

child: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Text(msg, style: TextStyle(fontSize: 30, fontStyle: FontStyle.italic),),

RaisedButton(

child: Text("Click Here", style: TextStyle(fontSize: 20),),

onPressed: \_changeText,

color: Colors.red,

textColor: Colors.yellow,

padding: EdgeInsets.all(8.0),

splashColor: Colors.grey,

)

],

),

),

),

),

);

}

\_changeText() {

setState(() {

if (msg.startsWith('F')) {

msg = 'We have learned FlutterRaised button example.';

} else {

msg = 'Flutter RaisedButton Example';

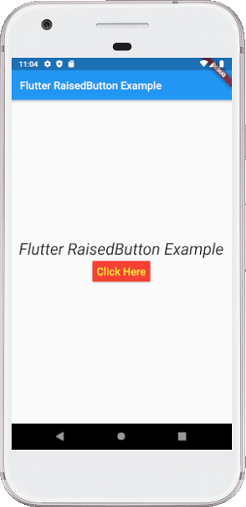
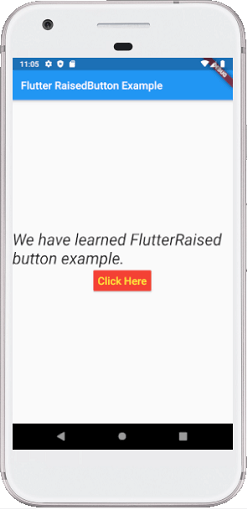
}

});

}

}

* **Kết quả đầu ra:** Khi chạy ví dụ bên trên, nó sẽ cung cấp ảnh chụp màn hình bên dưới. Nếu nhấp vào nút “***Click here***”, nó sẽ thay đổi tin nhắn văn bản. Hiển thị ảnh chụp màn hình thứ hai.

1. **Nút Hành động Nổi**

Nút FAB là một **nút biểu tượng hình tròn** kích hoạt hành động chính trong ứng dụng của chúng ta. Nó là nút được sử dụng nhiều nhất trong các ứng dụng hiện nay. Chúng ta có thể sử dụng nút này để thêm, làm mới hoặc chia sẻ nội dung. Flutter đề xuất sử dụng nhiều nhất một nút FAB trên mỗi màn hình.

Có hai loại Nút hành động nổi:

**+ *FloatingActionButton:*** Nó tạo một nút nổi hình tròn đơn giản với một widget con bên trong nó. Nó phải có một tham số chile để hiển thị một widget.

**+ *FloatingActionButton.extended:*** Nó tạo ra một nút nổi rộng cùng với một biểu tượng và nhãn bên trong nó. Thay vì một chile, nó sử dụng các nhãn và các thông số biểu tượng.

**Ví dụ:** Mở tệp **main.dart** và thay thế bằng code bên dưới.

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatefulWidget {

@override

\_MyAppState createState() => \_MyAppState();

}

class \_MyAppState extends State<MyApp> {

@override

Widget build(BuildContext context) {

return MaterialApp(home: Scaffold(

appBar: AppBar(

title: Text("FAB Button Example"),

backgroundColor: Colors.blue,

actions: <Widget>[

IconButton(icon: Icon(Icons.camera\_alt), onPressed: () => {}),

IconButton(icon: Icon(Icons.account\_circle), onPressed: () => {})

],

),

floatingActionButton: FloatingActionButton(

child: Icon(Icons.navigation),

backgroundColor: Colors.green,

foregroundColor: Colors.white,

onPressed: () => {}, ), ), ); } }

* **Kết quả đầu ra:** Chạy ứng dụng trong trình giả lập android và nó sẽ cung cấp giao diện người dùng tương tự như ảnh chụp màn hình sau. Hình ảnh thứ hai là đầu ra của nút **FAB.extended**. Code hóa của nó có thể được nhìn thấy trong phần bình luận của code ở trên.

A white rectangular frame with a white background

Description automatically generated A white rectangular object with a white background

Description automatically generated

1. **Nút DropDown**

Một nút thả xuống được sử dụng để tạo một lớp phủ đẹp mắt trên màn hình cho phép người dùng chọn bất kỳ mục nào từ nhiều tùy chọn. Flutter cho phép một cách đơn giản để triển khai hộp thả xuống hoặc nút thả xuống. Nút này hiển thị mục hiện được chọn và một mũi tên mở menu để chọn một mục từ nhiều tùy chọn.

Flutter cung cấp một **widget DropdownButton** để triển khai một danh sách thả xuống. Chúng ta có thể đặt nó ở bất kỳ đâu trong ứng dụng của mình.

**Ví dụ:** Mở tệp **main.dart** và thay thế bằng code bên dưới.

import 'package:flutter/material.dart';

void main() => runApp(MaterialApp(

home: MyApp(),

));

class MyApp extends StatefulWidget {

@override

\_MyAppState createState() => \_MyAppState();

}

class \_MyAppState extends State<MyApp> {

List<ListItem> \_dropdownItems = [

ListItem(1, "Cafedev.vn"),

ListItem(2, "Cafedev 1234"),

ListItem(3, "tutorialandexample"),

ListItem(4, "Cafedev 456")

];

List<DropdownMenuItem<ListItem>> \_dropdownMenuItems;

ListItem \_itemSelected;

void initState() {

super.initState();

\_dropdownMenuItems = buildDropDownMenuItems(\_dropdownItems);

\_itemSelected = \_dropdownMenuItems[1].value;

}

List<DropdownMenuItem<ListItem>> buildDropDownMenuItems(List listItems) {

List<DropdownMenuItem<ListItem>> items = List();

for (ListItem listItem in listItems) {

items.add(

DropdownMenuItem(

child: Text(listItem.name),

value: listItem,

),

);

}

return items;

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("DropDown Button Example"),

),

body: Column(

children: <Widget>[

Padding(

padding: const EdgeInsets.all(10.0),

child: Container(

padding: const EdgeInsets.all(5.0),

decoration: BoxDecoration(

color: Colors.greenAccent,

border: Border.all()),

child: DropdownButtonHideUnderline(

child: DropdownButton(

value: \_itemSelected,

items: \_dropdownMenuItems,

onChanged: (value) {

setState(() {

\_itemSelected = value;

});

}),

),

),

),

Text("We have selected ${\_itemSelected.name}"),

],

),

);

}

}

class ListItem {

int value;

String name;

ListItem(this.value, this.name);

}

1. **Nút biểu tượng**

IconButton là một hình ảnh được in trên widget Material. Nó là một widget hữu ích mang lại cho giao diện người dùng Flutter một cảm giác thiết kế material design. Chúng tôi cũng có thể tùy chỉnh giao diện của nút này. Nói một cách dễ hiểu, nó là một biểu tượng phản ứng khi người dùng chạm vào nó.

**Ví dụ:** Mở tệp **main.dart** và thay thế bằng code bên dưới.

import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(

title: Text("Icon Button Example"),

),

body: Center(

child: MyStatefulWidget(), ), ), ); } }

double \_speakervolume = 0.0;

class MyStatefulWidget extends StatefulWidget {

MyStatefulWidget({Key key}) : super(key: key);

@override

\_MyStatefulWidgetState createState() => \_MyStatefulWidgetState();

}

class \_MyStatefulWidgetState extends State<MyStatefulWidget> {

Widget build(BuildContext context) {

return Column(

mainAxisSize: MainAxisSize.min,

children: <Widget>[

IconButton(

icon: Icon(Icons.volume\_up),

iconSize: 50,

color: Colors.brown,

tooltip: 'Increase volume by 5',

onPressed: () {

setState(() {

\_speakervolume += 5; }); }, ),

Text('Speaker Volume: $\_speakervolume') ], ); } }

* **Kết quả đầu ra:** Chạy ứng dụng trong trình giả lập android và nó sẽ cung cấp giao diện người dùng tương tự như ảnh chụp màn hình sau. Khi nhấn nút âm lượng, nó sẽ luôn tăng thêm 5.

A white cell phone with a white screen

Description automatically generated

1. **Nút Inkwell**

Nút InkWell là một khái niệm thiết kế material design, được sử dụng để **phản hồi cảm ứng**. Widget này nằm trong widget Vật liệu, nơi các phản ứng mực thực sự được vẽ. Nó tạo ra giao diện người dùng ứng dụng tương tác bằng cách thêm phản hồi cử chỉ. Nó chủ yếu được sử dụng để thêm **hiệu ứng gợn sóng**.

**Ví dụ:** Mở tệp **main.dart** và thay thế bằng code bên dưới.

import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatefulWidget {

@override

\_MyAppState createState() => \_MyAppState();

}

class \_MyAppState extends State<MyApp> {

int \_volume = 0;

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(

title: Text('InkWell Button Example'),

),

body: Center(

child: new Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

InkWell(

splashColor: Colors.green,

highlightColor: Colors.blue,

child: Icon(Icons.ring\_volume, size: 50),

onTap: () {

setState(() {

\_volume += 2;

});

},

),

Text (

\_volume.toString(),

style: TextStyle(fontSize: 50) ), ], ),

),

),

);

}

}

* **Kết quả đầu ra:** Chạy ứng dụng trong trình giả lập android và nó sẽ cung cấp giao diện người dùng tương tự như ảnh chụp màn hình sau. Mỗi khi nhấn nút âm lượng chuông, nó sẽ tăng 2 âm lượng.

A white cell phone with a white screen

Description automatically generated

1. **Nút PopupMenu**

Nó là một nút hiển thị menu khi nó được nhấn và sau đó gọi phương thức onSelected, menu bị loại bỏ. Đó là vì mục từ nhiều tùy chọn được chọn. Nút này chứa một văn bản và một hình ảnh. Nó sẽ chủ yếu sử dụng với menu cài đặt để liệt kê tất cả các tùy chọn. Nó giúp tạo ra trải nghiệm người dùng tuyệt vời.

**Ví dụ:** Mở tệp **main.dart** và thay thế bằng code bên dưới.

import 'package:flutter/material.dart';

void main() { runApp(MyApp());}

class MyApp extends StatefulWidget {

@override

\_MyAppState createState() => \_MyAppState();

}

class \_MyAppState extends State<MyApp> {

Choice \_selectedOption = choices[0];

void \_select(Choice choice) {

setState(() {

\_selectedOption = choice;

});

}

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(

title: const Text('PopupMenu Button Example'),

actions: <Widget>[

PopupMenuButton<Choice>(

onSelected: \_select,

itemBuilder: (BuildContext context) {

return choices.skip(0).map((Choice choice) {

return PopupMenuItem<Choice>(

value: choice,

child: Text(choice.name),

);

}).toList();

},

),

],

),

body: Padding(

padding: const EdgeInsets.all(10.0),

child: ChoiceCard(choice: \_selectedOption),

),

),

);

}

}

class Choice {

const Choice({this.name, this.icon});

final String name;

final IconData icon;

}

const List<Choice> choices = const <Choice>[

const Choice(name: 'Wi-Fi', icon: Icons.wifi),

const Choice(name: 'Bluetooth', icon: Icons.bluetooth),

const Choice(name: 'Battery', icon: Icons.battery\_alert),

const Choice(name: 'Storage', icon: Icons.storage),

];

class ChoiceCard extends StatelessWidget {

const ChoiceCard({Key key, this.choice}) : super(key: key);

final Choice choice;

@override

Widget build(BuildContext context) {

final TextStyle textStyle = Theme.of(context).textTheme.headline;

return Card(

color: Colors.greenAccent,

child: Center(

child: Column(

mainAxisSize: MainAxisSize.min,

crossAxisAlignment: CrossAxisAlignment.center,

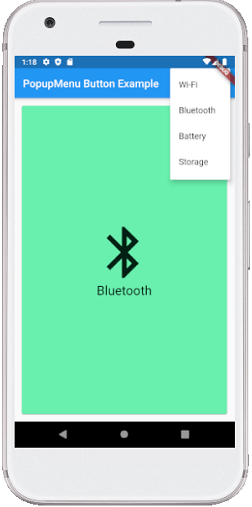
children: <Widget>[

Icon(choice.icon, size: 115.0, color: textStyle.color),

Text(choice.name, style: textStyle), ], ), ), ); } }

* **Kết quả đầu ra:** Chạy ứng dụng trong trình giả lập android và nó sẽ cung cấp giao diện người dùng tương tự như ảnh chụp màn hình sau. Khi nhấp vào ba dấu chấm hiển thị ở góc trên cùng bên trái của màn hình, nó sẽ bật lên nhiều tùy chọn. Ở đây, có thể chọn bất kỳ tùy chọn nào và nó sẽ giữ nó trong thẻ, như trong hình thứ hai.

A close-up of a phone

Description automatically generated 

1. **Nút phác thảo**

Nó tương tự như nút phẳng, nhưng nó có một đường viền mỏng hình chữ nhật tròn màu xám. Đường viền phác thảo của nó được xác định bởi thuộc tính shape.

**Ví dụ:** Mở tệp **main.dart** và thay thế bằng code bên dưới.

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatefulWidget {

@override

\_MyAppState createState() => \_MyAppState();

}

class \_MyAppState extends State<MyApp> {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(

title: Text('Outline Button Example'),

),

body: Center(child: Column(children: <Widget>[

Container(

margin: EdgeInsets.all(25),

child: OutlineButton(

child: Text("Outline Button", style: TextStyle(fontSize: 20.0),),

highlightedBorderColor: Colors.red,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(15)),

onPressed: () {},

),

),

Container(

margin: EdgeInsets.all(25),

child: FlatButton(

child: Text('Flat Button', style: TextStyle(fontSize: 20.0),),

color: Colors.blueAccent,

textColor: Colors.white,

onPressed: () {}, ), ), ], ),) , ), ); }

}

* **Kết quả đầu ra:** Chạy ứng dụng trong trình giả lập android và nó sẽ cung cấp giao diện người dùng tương tự như ảnh chụp màn hình sau.

A screen shot of a phone

Description automatically generated

1. **Nút Bar**

Flutter cung cấp sự linh hoạt để sắp xếp các nút trong một thanh hoặc một hàng. widget con ButtonBar chứa ba thuộc tính: alignment, children, and mainAxisSize.

* Căn chỉnh được sử dụng để trình bày tùy chọn căn chỉnh cho toàn bộ widget thanh nút.
* Thuộc tính Children được sử dụng để lấy số lượng nút trong một thanh.
* Thuộc tính mainAxisSize được sử dụng để cung cấp không gian ngang cho thanh nút.

**Ví dụ:** Mở tệp **main.dart** và thay thế bằng code bên dưới.

import 'package:flutter/material.dart';

void main() {

runApp(MaterialApp( home: MyApp(),));

}

class MyApp extends StatefulWidget {

@override

\_State createState() => \_State();

}

class \_State extends State<MyApp> {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Flutter ButtonBar Example'),

),

body: Padding(

padding: EdgeInsets.all(10),

child: Column(

children: <Widget>[

Padding(

padding: EdgeInsets.all(15),

child: new ButtonBar(

mainAxisSize: MainAxisSize.min,

children: <Widget>[

RaisedButton(

child: new Text('Javatpoint'),

color: Colors.lightGreen,

onPressed: () {/\*\* \*/},

),

FlatButton(

child: Text('Flutter'),

color: Colors.lightGreen,

onPressed: () {/\*\* \*/},

),

FlatButton(

child: Text('MySQL'),

color: Colors.lightGreen,

onPressed: () {/\*\* \*/},

),

],

),

),

],

)

)

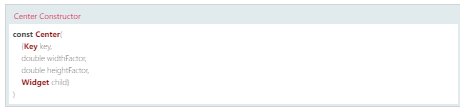
);

}

}

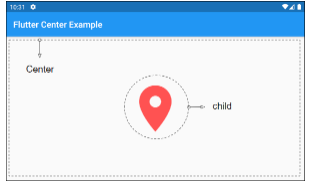
### 2.3.4. Tìm hiểu nội dung Center

Center là một widget đặt widget con duy nhất của nó tại trung tâm của nó. Center và Align khá giống nhau, chúng chỉ có duy nhất một widget con, nhưng Align cho phép bạn tùy biến vị trí của widget con bên trong nó.

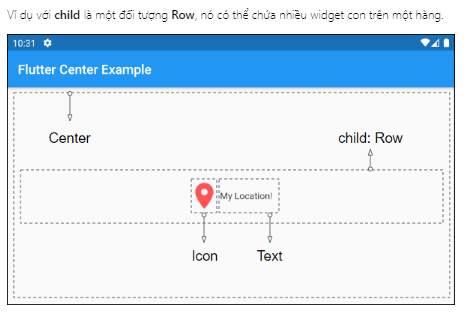


Nếu tham số widthFactor không được chỉ định thì chiều rộng của Center sẽ lớn nhất có thể, ngược lại chiều rộng của Center bằng chiều rộng của child nhân với widthFactor. Tham số heightFactor cũng có hành vi tương tự đối với chiều cao của Center. Như vậy theo mặc định kích thước của Center sẽ lớn nhất có thể.



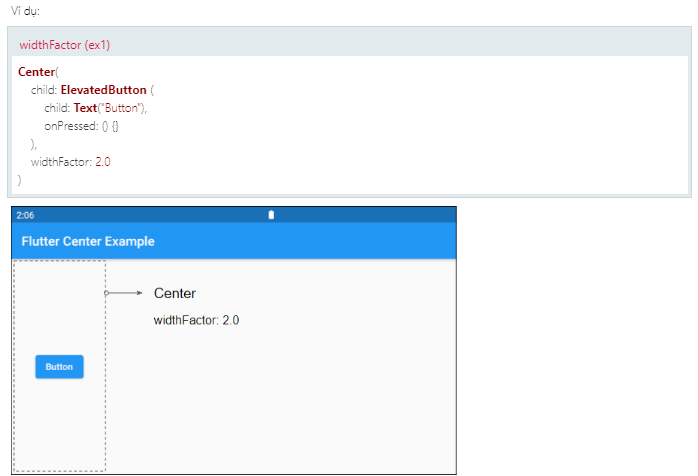


Child là widget con duy nhất của Center. Trong một số tình huống sử dụng nó có thể là đối tượng Row, Column hoặc Stack để có thể chứa được nhiều widget khác.



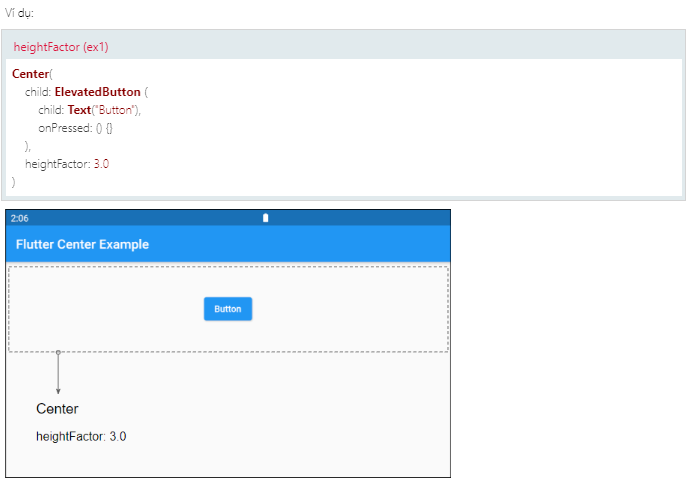
* **widthFactor**

WidthFactor là một hệ số được sử dụng để tính toán chiều rộng của Center dựa trên chiều rộng của child (Con của nó). Nếu widthFactor không null chiều rộng của Center bằng chiều rộng của child nhân với hệ số này. Nếu ***widthFactor*** không được chỉ định chiều rộng của Center sẽ lớn nhất có thể.



* **heightFactor**

HeightFactor là một hệ số được sử dụng để tính toán chiều cao của Center dựa trên chiều cao của child (Con của nó). Nếu heightFactor không null chiều cao của Center bằng chiều cao của child nhân với hệ số này. Nếu heightFactor không được chỉ định chiều cao của Center sẽ lớn nhất có thể.



### 2.3.5. Tìm hiểu nội dung Media Query

* **Khái niệm**

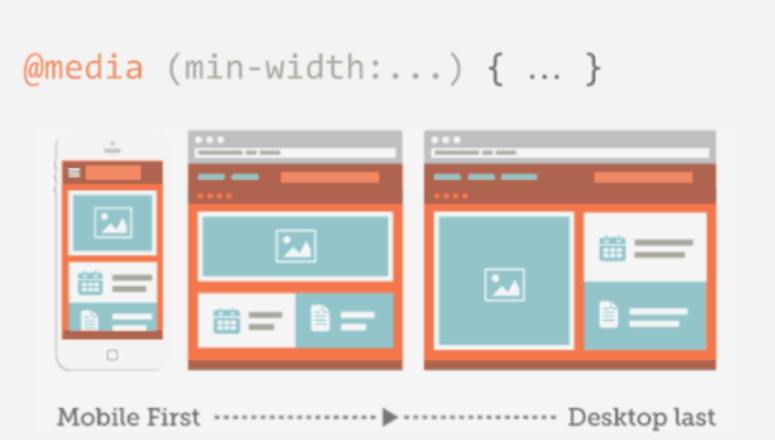
- Media Query là một cơ chế trong CSS để điều chỉnh kiểu hiển thị của ứng dụng dựa trên các đặc tính của thiết bị và môi trường hiển thị. Bằng cách sử dụng Media Query, có thể thay đổi các thuộc tính CSS như kích thước, font chữ, bố cục và các phần tử khác của ứng dụng dựa trên độ phân giải, kích thước, phương diện, tỷ lệ khung hình, độ phân giải,..Media Query đóng vai trò quan trọng trong việc xây dựng các ứng dụng đáp ứng (responsive) và tối ưu trải nghiệm người dùng trên nhiều loại thiết bị khác nhau.

* **Chức năng**

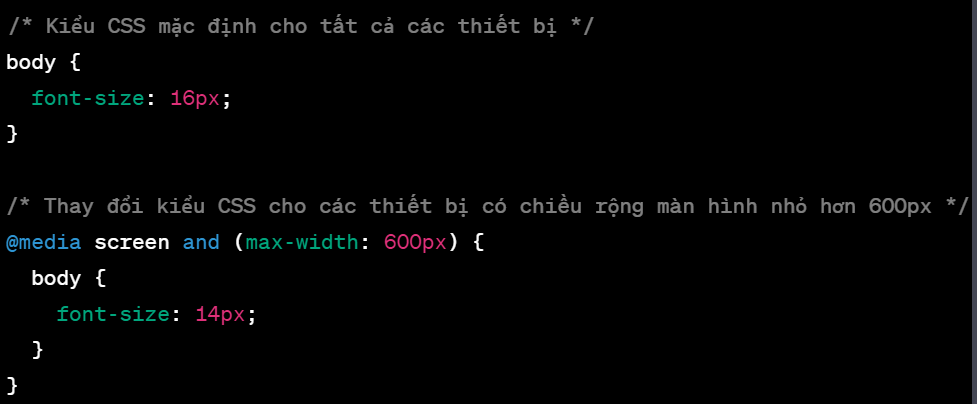
- Trong lập trình Android, việc tùy chỉnh giao diện cho các thiết bị khác nhau là cực kỳ quan trọng để đảm bảo ứng dụng của bạn hoạt động mượt mà trên mọi loại màn hình. Để làm điều này, bạn có thể sử dụng các tài nguyên thay thế và cơ chế phân giải màn hình sau:

- Sử dụng các thư mục tài nguyên như drawable-hdpi, drawable-mdpi, drawable-xhdpi,... để cung cấp hình ảnh và biểu đồ với chất lượng và kích thước phù hợp cho từng loại màn hình.

- Sử dụng các thư mục layout-large, layout-xlarge,...để cung cấp bố cục và giao diện phù hợp cho các màn hình lớn hoặc tablet.



Media query trong CSS có thể sử dụng như sau:



### 2.3.6. Tìm hiểu nội dung ListView

* **Khái niệm**

Trong Android, List View là một thành phần giao diện người dùng (UI) dùng để hiển thị một danh sách các mục có thể cuộn. Nó cho phép bạn hiển thị dữ liệu trong dạng danh sách, như văn bản, hình ảnh hoặc bất kỳ loại nội dung nào khác, và người dùng có thể cuộn qua danh sách này để xem tất cả các mục.

**Hiển thị dữ liệu trong List View :**

Hiển thị dữ liệu trong List View gồm có những phần sau :

* **Adapter:**

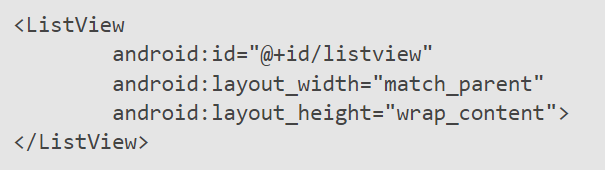
Adapter là một lớp trung gian giữa dữ liệu và List View. Nhiệm vụ của nó là chuyển đổi dữ liệu thành các mục hiển thị trên List View. Có nhiều loại Adapter khác nhau, như ArrayAdapter, BaseAdapter hoặc RecyclerView. Adapter, tùy thuộc vào loại List View bạn đang sử dụng.

Trong Android, Adapter thường được sử dụng để liên kết dữ liệu với UI

1. BaseAdapter : Là lớp adpater cơ sở cho các Adapter khác
2. ArrayAdapter: Một ListAdapter có thể quản lí một ListView chứa danh sách các phần tử có kiểu bất kì
3. Custom ArrayAdapter: Thường được sử dụng để hiển thị một danh sách tùy chỉnh
4. SimpleAdapter: Nó là một Adapter đơn giản và dễ hiểu để ánh xạ dữ liệu vào những View được định nghĩa trong một tập tin XML
5. Custom SimpleAdapter: Nó được sử dụng để hiển thị một danh sách được tùy chỉnh để truy cập các mục con của List hoặc Grid.

* **Layout item:** Đây là layout XML mô tả cách mỗi mục trong danh sách sẽ được hiển thị. Bạn có thể tùy chỉnh layout item để hiển thị thông tin phức tạp hơn cho mỗi mục.
* **Data source:** Đây là dữ liệu bạn muốn hiển thị trong List View, thường là một danh sách các đối tượng hoặc các mục cơ bản như chuỗi văn bản.
* **Scrolling:** List View có thể cuộn để hiển thị nhiều mục hơn so với kích thước màn hình. Điều này cho phép người dùng dễ dàng duyệt qua danh sách và xem tất cả các mục.
* **ListView API:** List View cung cấp một số phương thức và thuộc tính để bạn có thể thao tác và tùy chỉnh cách List View hoạt động và hiển thị.

**Cú pháp khai báo List View:**



**Các thuộc tính của List View**

|  |  |
| --- | --- |
| **Thuộc tính** | **Mô tả** |
| **android:id** | id là duy nhất để nhận diện Layout |
| **android:divider** | Nó có thể vẽ hoặc tô màu giữa các item trong danh sách |
| **android:dividerHeight** | Nó xác định chiều cao của dải phân cách. Điều này có thể là **px,dp,sp,in** hoặc **mm.** |
| **android:entries** | Chỉ định tham chiếu đến một tài nguyên mảng sẻ điền vào **ListView.** |
| **android:footerDividersEnable** | Khi được đặt thành **false, ListView** sẽ không vẽ dải phân cách trước mỗi chế độ xem chân trang. Giá trị mặc định là **true.** |
| **android:headerDividersEnable** | Khi được đặt thành **false, ListView** sẽ không vẽ dãi phân cách sau mỗi chế độ xem tiêu đề. Giá trị mặc định là **true.** |

* **Ưu điểm và nhược điểm :**

**+ Ưu điểm :**

* *Hiển thị danh sách dữ liệu*: List View cho phép bạn hiển thị danh sách các mục dưới dạng danh sách đơn giản. Điều này thường được sử dụng để hiển thị danh sách các văn bản, hình ảnh, hoặc bất kỳ loại nội dung nào khác trong ứng dụng của bạn.
* *Dễ dàng tùy chỉnh*: Bạn có thể tùy chỉnh giao diện của mỗi mục trong danh sách bằng cách sử dụng layout item. Điều này cho phép bạn hiển thị thông tin phức tạp hơn cho mỗi mục và tạo giao diện đẹp hơn cho danh sách.
* *Cuộn qua danh sách*: List View cho phép người dùng cuộn qua danh sách nếu danh sách chứa nhiều mục hơn so với kích thước màn hình. Điều này giúp tiết kiệm không gian màn hình và cho phép người dùng xem tất cả các mục trong danh sách.
* *Dễ dàng xử lý sự kiện*: Bạn có thể đăng ký một Item Click Listener để xử lý sự kiện khi người dùng chọn một mục trong danh sách. Điều này cho phép bạn thực hiện một hành động cụ thể khi người dùng tương tác với danh sách.
* *Đơn giản và dễ dàng sử dụng*: List View là một thành phần cơ bản và dễ dàng sử dụng. Bạn chỉ cần tạo một Adapter để cung cấp dữ liệu cho List View và gán nó vào List View của bạn. Không cần nhiều mã phức tạp để hiển thị danh sách cơ bản.

**+ Nhược điểm :**

* *Hiệu suất khi có nhiều mục*: Khi danh sách chứa một số lượng lớn các mục, List View có thể gặp phải vấn đề hiệu suất. Điều này có thể dẫn đến hiện tượng trễ khi cuộn qua danh sách hoặc khi cập nhật dữ liệu trong danh sách.
* *Phần tử trong danh sách không tái sử dụng*: List View không tái sử dụng các phần tử trong danh sách khi cuộn qua. Do đó, khi danh sách lớn, sẽ tốn nhiều bộ nhớ để duy trì tất cả các phần tử trong danh sách, dẫn đến lãng phí tài nguyên.
* *Hỗ trợ ít tính năng*: List View cung cấp một số tính năng cơ bản, nhưng không thể thực hiện được một số tính năng phức tạp hơn như hiển thị các mục có nhiều cột, các hiệu ứng động phức tạp, hay tái sắp xếp tự động danh sách.
* *Khó tùy chỉnh giao diện*: Mặc dù bạn có thể tùy chỉnh giao diện của mỗi mục trong danh sách bằng cách sử dụng layout item, nhưng việc tùy chỉnh giao diện phức tạp hơn có thể làm cho mã nguồn phức tạp hơn.
* *Không hỗ trợ hiển thị lưới*: List View mặc định chỉ hỗ trợ hiển thị các mục dưới dạng danh sách dọc. Nếu bạn muốn hiển thị các mục theo dạng lưới, bạn phải tự thực hiện hoặc sử dụng một thành phần khác như Grid View hoặc RecyclerView để thực hiện tính năng này.

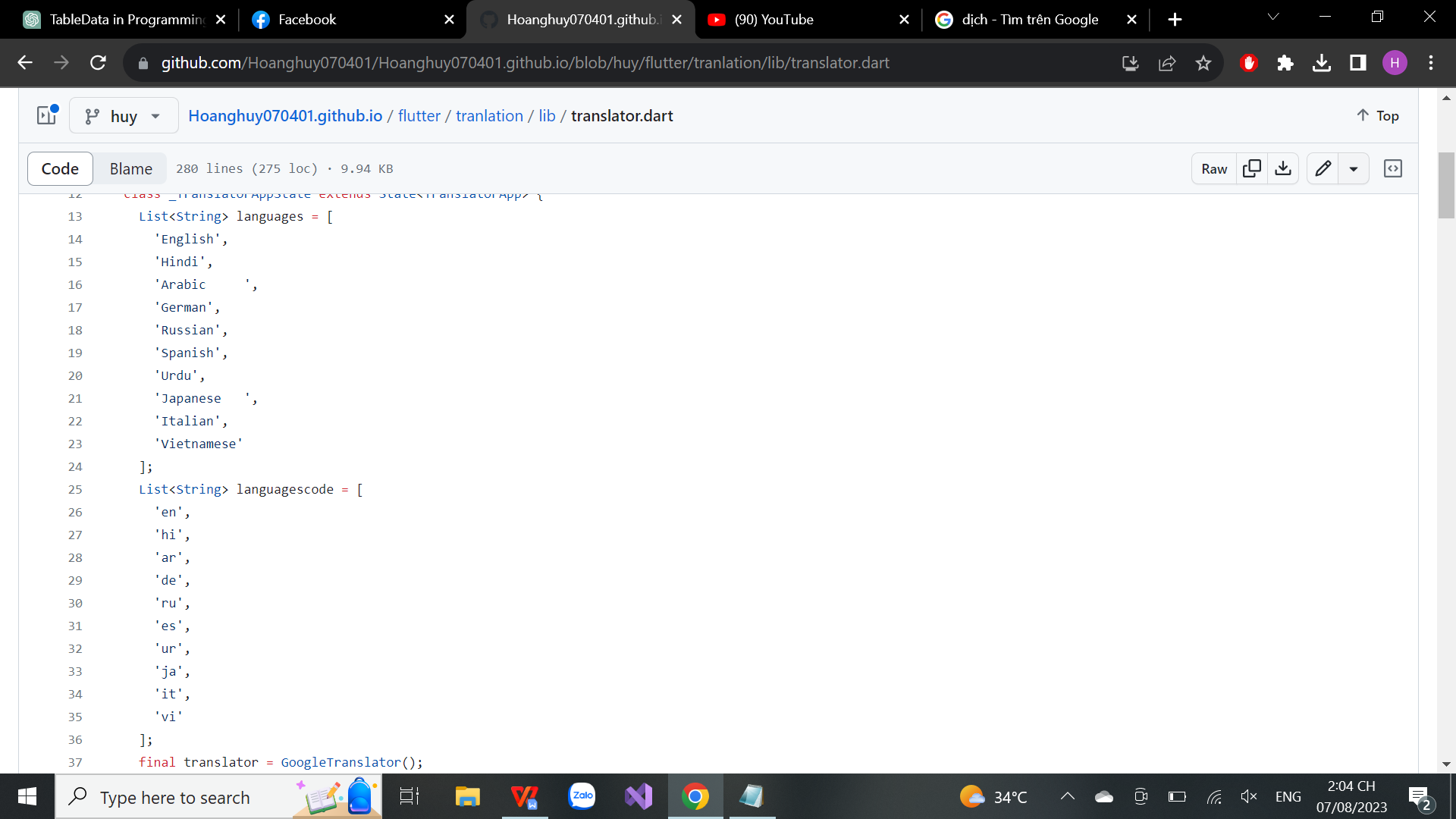
### 2.3.7. Tìm hiểu nội dung TableData

Trong lập trình, "table data" là một thuật ngữ để chỉ dữ liệu được tổ chức dưới dạng bảng hoặc lưới, gồm các hàng và cột. Các dữ liệu được sắp xếp vào các ô của bảng, giúp tổ chức và hiển thị thông tin một cách có cấu trúc và dễ hiểu.

Khi lập trình một phần mềm Android phiên dịch tiếng Anh (hoặc bất kỳ ngôn ngữ nào khác), việc sử dụng table data không phải là một yêu cầu bắt buộc, nhưng nó có thể hữu ích để tổ chức và quản lý các dữ liệu văn bản liên quan đến việc dịch và đa ngôn ngữ.

Trong Flutter, bạn có thể sử dụng table data thông qua các cấu trúc dữ liệu như List và Map để tổ chức và quản lý dữ liệu.

Ví dụ sử dụng TableData để biểu diễn danh sách các ngôn ngữ:



### 2.3.8. Tìm hiểu nội dung GirdView

Chế độ xem dạng lưới là một thành phần điều khiển đồ họa được sử dụng để hiển thị các mục ở dạng bảng. Trong phần này, chúng ta sẽ tìm hiểu cách hiển thị các mục trong chế độ xem lưới trong ứng dụng Flutter.

GridView là một tiện ích trong Flutter hiển thị các mục trong mảng 2 chiều (hàng và cột hai chiều). Như tên gợi ý, nó sẽ được sử dụng khi chúng ta muốn hiển thị các mục trong Lưới. Chúng ta có thể chọn mục mong muốn từ danh sách lưới bằng cách nhấn vào chúng. Tiện ích này có thể chứa văn bản, hình ảnh, biểu tượng, v.v... để hiển thị theo bố cục dạng lưới tùy theo yêu cầu của người dùng. Nó cũng được gọi là mảng vật dụng 2 chiều có thể cuộn được. Vì nó có thể cuộn được nên chúng ta chỉ có thể chỉ định hướng mà nó cuộn.

Chế độ xem lưới có thể được triển khai theo nhiều cách khác nhau, được đưa ra dưới đây:

1. [count()](https://www.javatpoint.com/flutter-gridview" \l "count)
2. [builder()](https://www.javatpoint.com/flutter-gridview" \l "builder)
3. custom()
4. [extent()](https://www.javatpoint.com/flutter-gridview" \l "extent)

* **GridView.count ()**

Đây là bố cục lưới được sử dụng thường xuyên nhất trong [Flutter](https://www.javatpoint.com/flutter-layouts) vì ở đây, chúng ta đã biết kích thước của lưới. Nó cho phép các nhà phát triển chỉ định số lượng hàng và cột cố định. GriedView.count () chứa các thuộc tính sau:

* CrossAxisCount: Nó được sử dụng để chỉ định số lượng cột trong chế độ xem dạng lưới.
* CrossAxisSpacing: Nó được sử dụng để chỉ định số pixel giữa mỗi tiện ích con được liệt kê trong trục chéo.
* MainAxisSpacing: Nó được sử dụng để chỉ định số pixel giữa mỗi tiện ích con được liệt kê trong trục chính.
* Padding(EdgeInsetsGeometry): Nó được sử dụng để chỉ định không gian xung quanh toàn bộ danh sách các vật dụng.
* ScrollDirection: Nó được sử dụng để chỉ định hướng mà các mục trên GridView cuộn. Theo mặc định, nó cuộn theo hướng dọc.
* Reverse: Nếu nó đúng, nó sẽ đảo ngược danh sách theo hướng ngược lại dọc theo trục chính.
* Physics: Nó được sử dụng để xác định cách hoạt động của danh sách khi người dùng đến cuối hoặc đầu tiện ích trong khi cuộn.
* ShrinkWrap: Nếu nó sai, danh sách có thể cuộn sẽ chiếm nhiều không gian hơn để cuộn theo hướng cuộn. Nó không tốt vì nó gây lãng phí bộ nhớ và giảm hiệu suất của ứng dụng. Do đó, chúng tôi sẽ bọc các tiện ích con của mình bằng cách sử dụng shrinkWrap bằng cách đặt nó thành true để tránh rò rỉ bộ nhớ trong khi cuộn.

**Ví dụ**

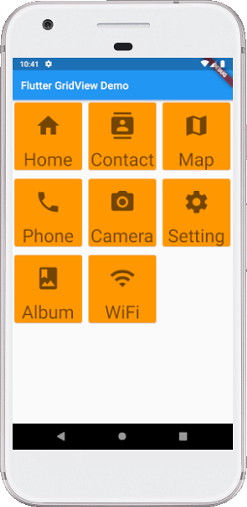
Hãy để chúng tôi hiểu nó bằng một ví dụ để xem cách tạo GridView trong [Flutter](https://www.javatpoint.com/flutter) để tạo danh sách lưới. Đầu tiên, tạo một dự án mới trong IDE mà bạn đang sử dụng. Mở dự án, điều hướng đến thư mục lib và thay thế mã bên dưới bằng tệp main.dart.

1. **import** 'package:flutter/material.dart';
3. **void** main() {runApp(MyApp());}
5. **class** MyApp **extends** StatelessWidget {
6. @override
7. Widget build(BuildContext context) {
8. **return** MaterialApp(
9. home: Scaffold(appBar: AppBar(
10. title: Text("Flutter GridView Demo"),
11. ),
12. body: GridView.count(
13. crossAxisCount: 3,
14. crossAxisSpacing: 4.0,
15. mainAxisSpacing: 8.0,
16. children: List.generate(choices.length, (index) {
17. **return** Center(
18. child: SelectCard(choice: choices[index]),
19. );
20. }
21. )
22. )
23. )
24. );
25. }
26. }
28. **class** Choice {
29. **const** Choice({**this**.title, **this**.icon});
30. **final** String title;
31. **final** IconData icon;
32. }
34. **const** List<Choice> choices = **const** <Choice>[
35. **const** Choice(title: 'Home', icon: Icons.home),
36. **const** Choice(title: 'Contact', icon: Icons.contacts),
37. **const** Choice(title: 'Map', icon: Icons.map),
38. **const** Choice(title: 'Phone', icon: Icons.phone),
39. **const** Choice(title: 'Camera', icon: Icons.camera\_alt),
40. **const** Choice(title: 'Setting', icon: Icons.settings),
41. **const** Choice(title: 'Album', icon: Icons.photo\_album),
42. **const** Choice(title: 'WiFi', icon: Icons.wifi),
43. ];
45. **class** SelectCard **extends** StatelessWidget {
46. **const** SelectCard({Key key, **this**.choice}) : **super**(key: key);
47. **final** Choice choice;
49. @override
50. Widget build(BuildContext context) {
51. **final** TextStyle textStyle = Theme.of(context).textTheme.display1;
52. **return** Card(
53. color: Colors.orange,
54. child: Center(child: Column(
55. crossAxisAlignment: CrossAxisAlignment.center,
56. children: <Widget>[
57. Expanded(child: Icon(choice.icon, size:50.0, color: textStyle.color)),
58. Text(choice.title, style: textStyle),
59. ]
60. ),
61. )
62. );
63. }
64. }

Trong đoạn mã trên, chúng tôi đã tạo một lớp cho các mục mà chúng tôi muốn hiển thị trong lưới và sau đó thêm một số dữ liệu. Tiếp theo, để xây dựng tiện ích thì sẽ hiển thị trong GridView.

* **Output**

Khi chạy ứng dụng trong Android Studio, có thể thấy màn hình sau trong Trình giả lập sau.



Hình 2. 8 GridView.count

* **GridView.builder ()**

Thuộc tính này được sử dụng khi chúng ta muốn hiển thị dữ liệu động hoặc theo yêu cầu. Nói cách khác, nếu người dùng muốn tạo một lưới có số lượng phần tử con lớn (vô hạn), thì họ có thể sử dụng hàm tạo GridView.builder ()với Sliver Grid Dele gate With Fixed Cross Axis Count hoặc Sliver Grid Dele gate With Max Cross Axis Extent.

Các thuộc tính phổ biến của widget này là:

**+ ItemCount :** Nó được sử dụng để xác định lượng dữ liệu sẽ được hiển thị.

**+ GridDelegate :** Nó xác định lưới hoặc bộ chia của nó. Đối số của nó không được rỗng.

**+ ItemBuilder :** Nó được sử dụng để tạo các mục sẽ được hiển thị trên chế độ xem lưới. Nó sẽ chỉ được gọi khi các chỉ số >= 0 && chỉ số < itemCount.

**Ví dụ**

Hãy để chúng tôi hiểu nó với sự giúp đỡ của một ví dụ. Mở dự án, điều hướng đến thư mục lib và thay thế mã bên dưới bằng tệp main.dart.

**import** 'package:flutter/material.dart';

**void** main() => runApp(MyApp());

**class** MyApp **extends** StatelessWidget {

  List<String> images = [

    "https://static.javatpoint.com/tutorial/flutter/images/flutter-logo.png",

    "https://static.javatpoint.com/tutorial/flutter/images/flutter-logo.png",

    "https://static.javatpoint.com/tutorial/flutter/images/flutter-logo.png",

    "https://static.javatpoint.com/tutorial/flutter/images/flutter-logo.png"

  ];

  @override

  Widget build(BuildContext context) {

**return** MaterialApp(

      home: Scaffold(

        appBar: AppBar(

          title: Text("Flutter GridView Demo"),

     backgroundColor: Colors.red,

        ),

        body: Container(

            padding: EdgeInsets.all(12.0),

            child: GridView.builder(

              itemCount: images.length,

              gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(

                  crossAxisCount: 2,

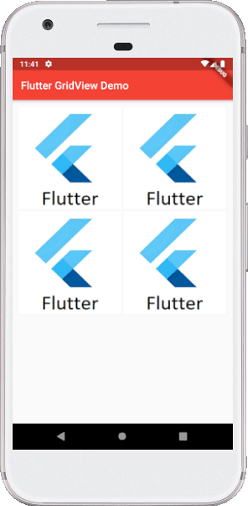
                  crossAxisSpacing: 4.0,

                  mainAxisSpacing: 4.0  ),

              itemBuilder: (BuildContext context, **int** index){

**return** Image.network(images[index]);  },  )),   ),   );   } }

* **Output:** Khi chạy ứng dụng trong Android Studio, có thể thấy màn hình sau trong Trình giả lập sau.



Hình 2. 9 GridView.builder

* **GridView.extent ()**

Thuộc tính này được sử dụng khi chúng ta muốn tạo lưới với các giá trị phạm vi tùy chỉnh. Nó có nghĩa là mỗi ô có phạm vi trục chéo tối đa.

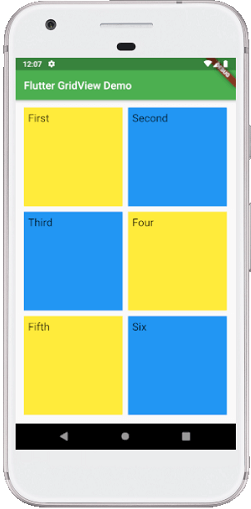
**Ví dụ**

Hãy để chúng tôi hiểu nó với sự giúp đỡ của một ví dụ. Mở dự án, điều hướng đến thư mục lib và thay thế mã bên dưới bằng tệp **main.dart**.

1. **import** 'package:flutter/material.dart';
2. **void** main() => runApp(MyApp());
3. **class** MyApp **extends** StatelessWidget {
4. // This widget is the root of your application.
5. @override
6. Widget build(BuildContext context) {
7. **return** MaterialApp(
8. home: MyGridScreen(),
9. );
10. }
11. }
12. **class** MyGridScreen **extends** StatefulWidget {
13. MyGridScreen({Key key}) : **super**(key: key);
14. @override
15. \_MyGridScreenState createState() => \_MyGridScreenState();
16. }
17. **class** \_MyGridScreenState **extends** State<MyGridScreen> {
18. @override
19. Widget build(BuildContext context) {
20. **return** Scaffold(
21. appBar: AppBar(
22. title: Text("Flutter GridView Demo"),
23. backgroundColor: Colors.green,
24. ),
25. body: Center(
26. child: GridView.extent(
27. primary: **false**,
28. padding: **const** EdgeInsets.all(16),
29. crossAxisSpacing: 10,
30. mainAxisSpacing: 10,
31. maxCrossAxisExtent: 200.0,
32. children: <Widget>[
33. Container(
34. padding: **const** EdgeInsets.all(8),
35. child: **const** Text('First', style: TextStyle(fontSize: 20)),
36. color: Colors.yellow,
37. ),
38. Container(
39. padding: **const** EdgeInsets.all(8),
40. child: **const** Text('Second', style: TextStyle(fontSize: 20)),
41. color: Colors.blue,
42. ),
43. Container(
44. padding: **const** EdgeInsets.all(8),
45. child: **const** Text('Third', style: TextStyle(fontSize: 20)),
46. color: Colors.blue,
47. ),
48. Container(
49. padding: **const** EdgeInsets.all(8),
50. child: **const** Text('Four', style: TextStyle(fontSize: 20)),
51. color: Colors.yellow,
52. ),
53. Container(
54. padding: **const** EdgeInsets.all(8),
55. child: **const** Text('Fifth', style: TextStyle(fontSize: 20)),
56. color: Colors.yellow,
57. ),
58. Container(
59. padding: **const** EdgeInsets.all(8),
60. child: **const** Text('Six', style: TextStyle(fontSize: 20)),
61. color: Colors.blue,
62. ),
63. ],
64. )),
65. );
66. }
67. }

* **Output**

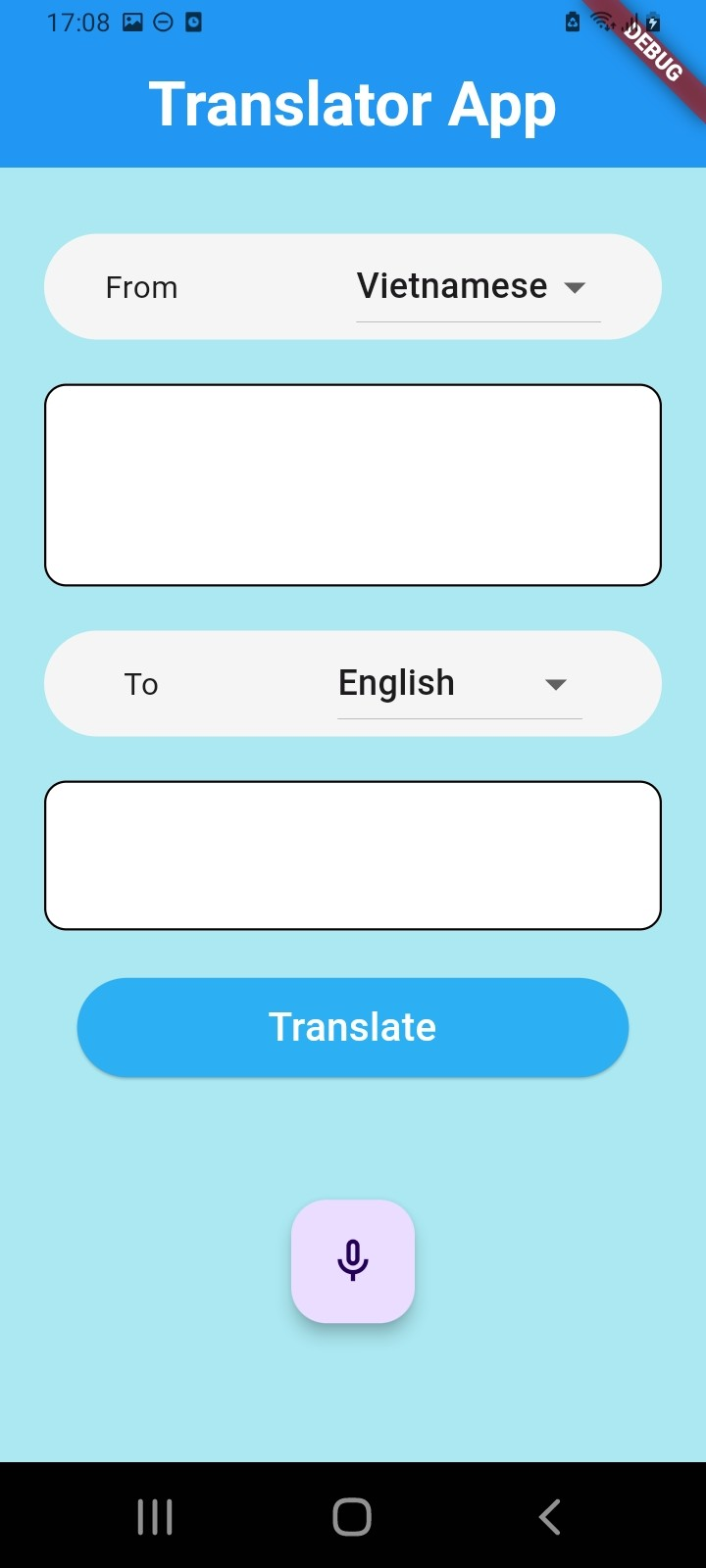
Khi chạy ứng dụng trong Android Studio, có thể thấy màn hình sau trong Trình giả lập sau.



Hình 2. 10 GridView.extent

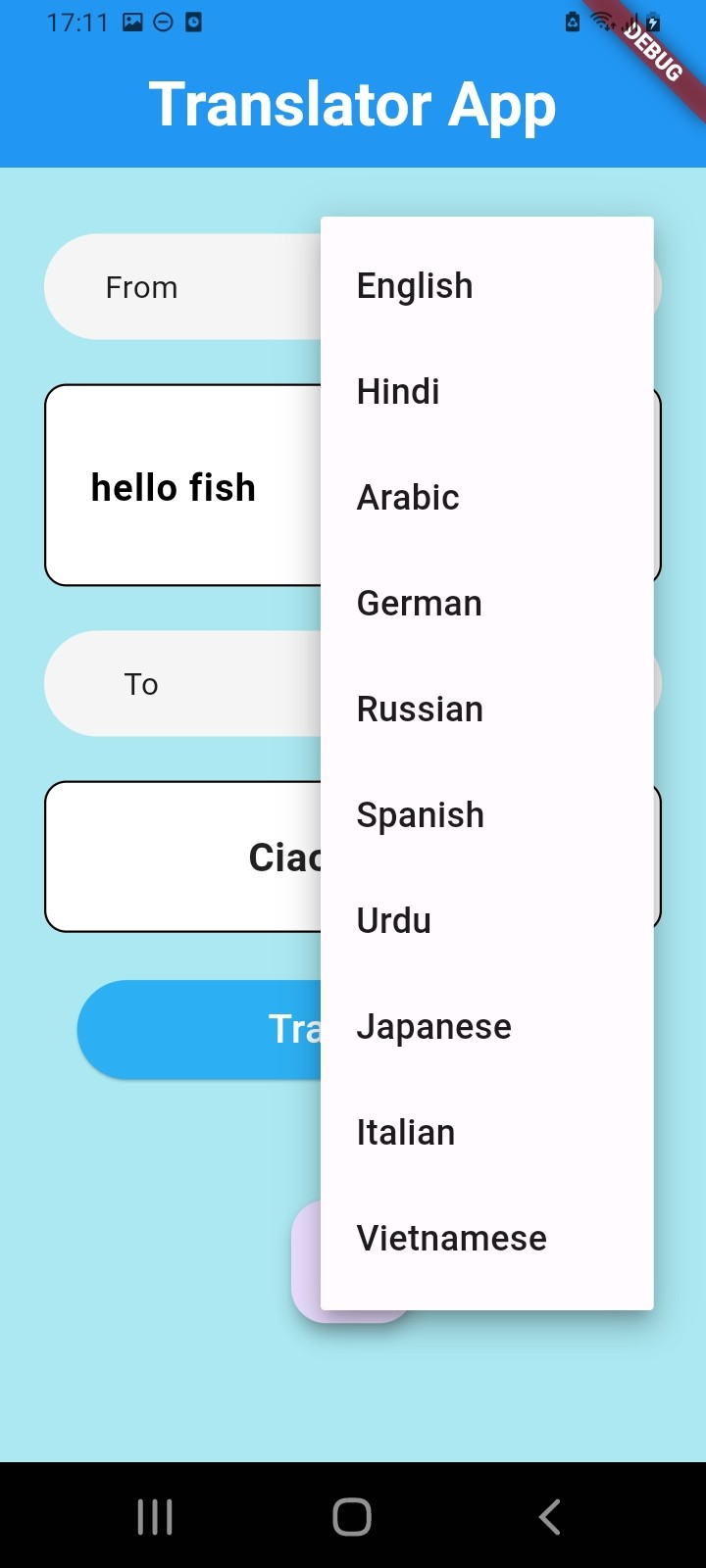
# CHƯƠNG 3: CÀI ĐẶT VÀ THỬ NGHIỆM

## 3.1. Giao diện



Hình 3. 1 Giao diện ứng dụng dịch thuật

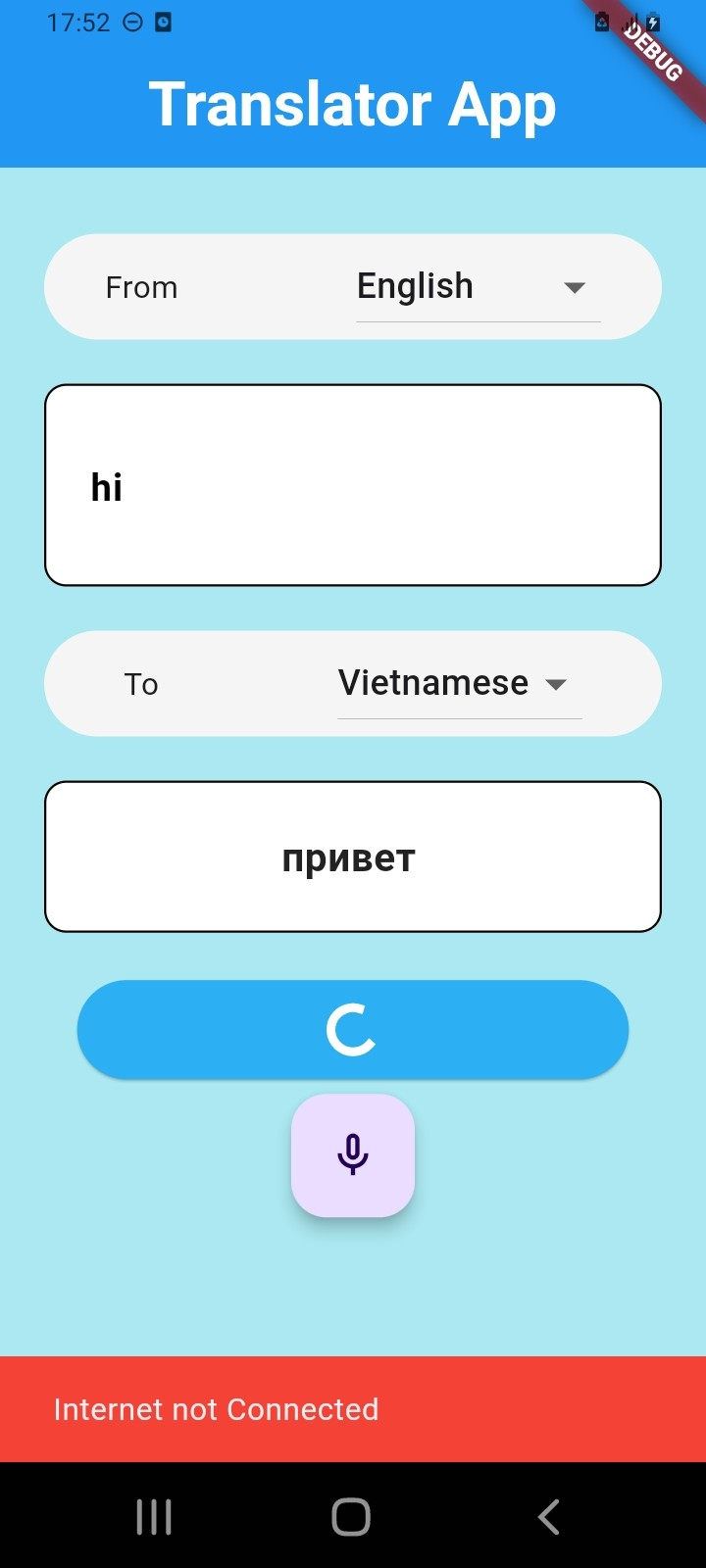
## 3.2. Danh sách các ngôn ngữ trong ứng dụng



Hình 3. 2 Danh sách ngôn ngữ dịch thuật

## 3.3. Dịch thuật

* **Khi không có kết nối mạng**



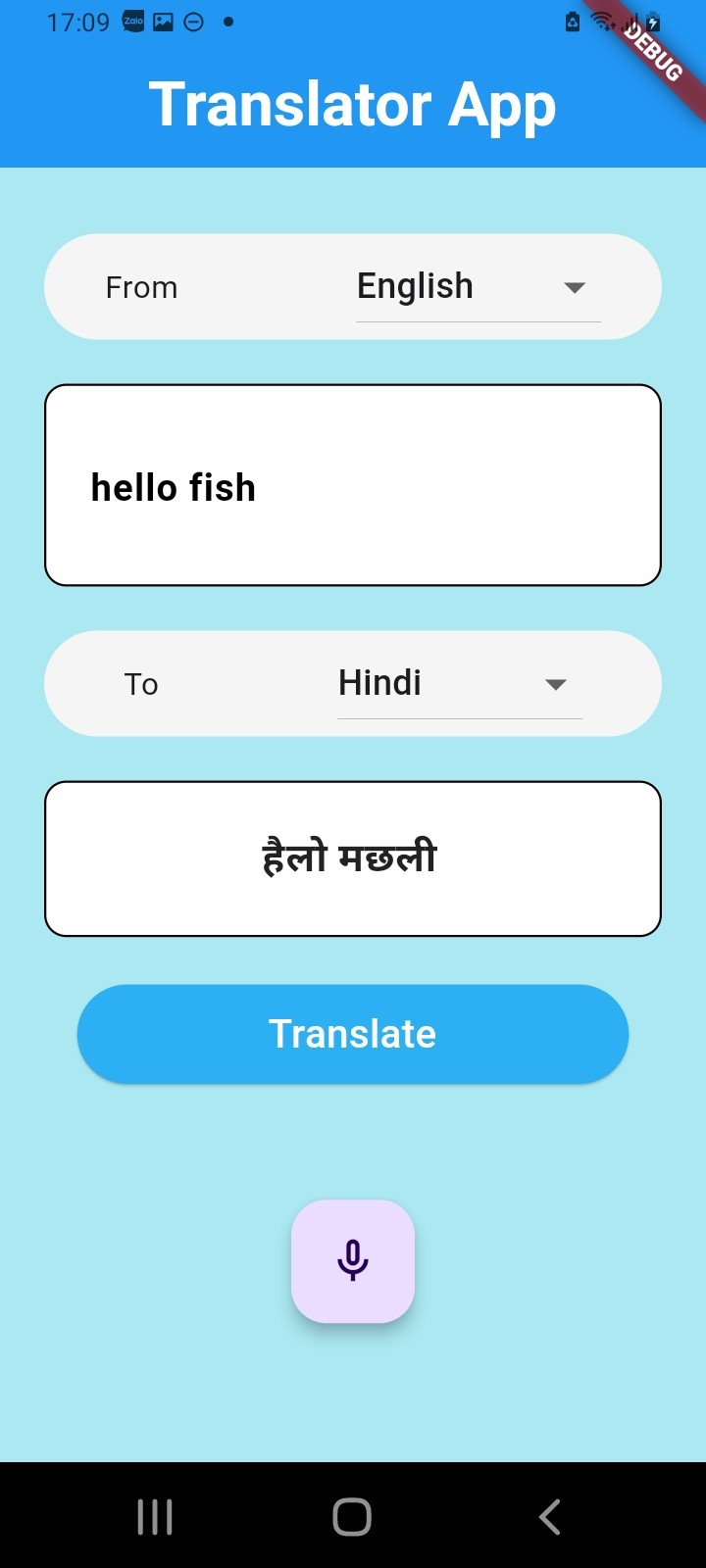
*Hình 3. 3 Thông báo khi không có mạng*

* **Tiếng Việt**



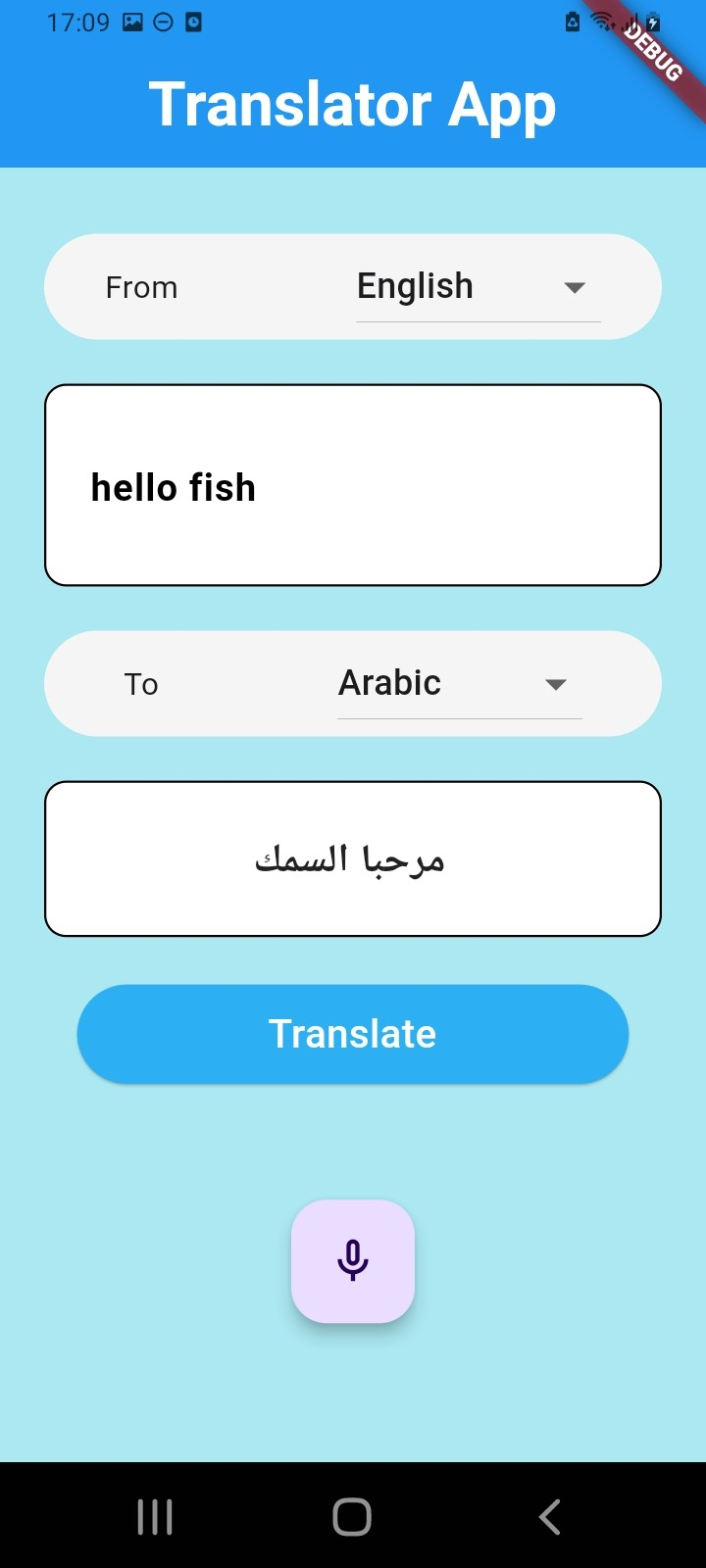
*Hình 3. 4 Dịch từ tiếng anh sang tiếng Việt*

* **Hindi**



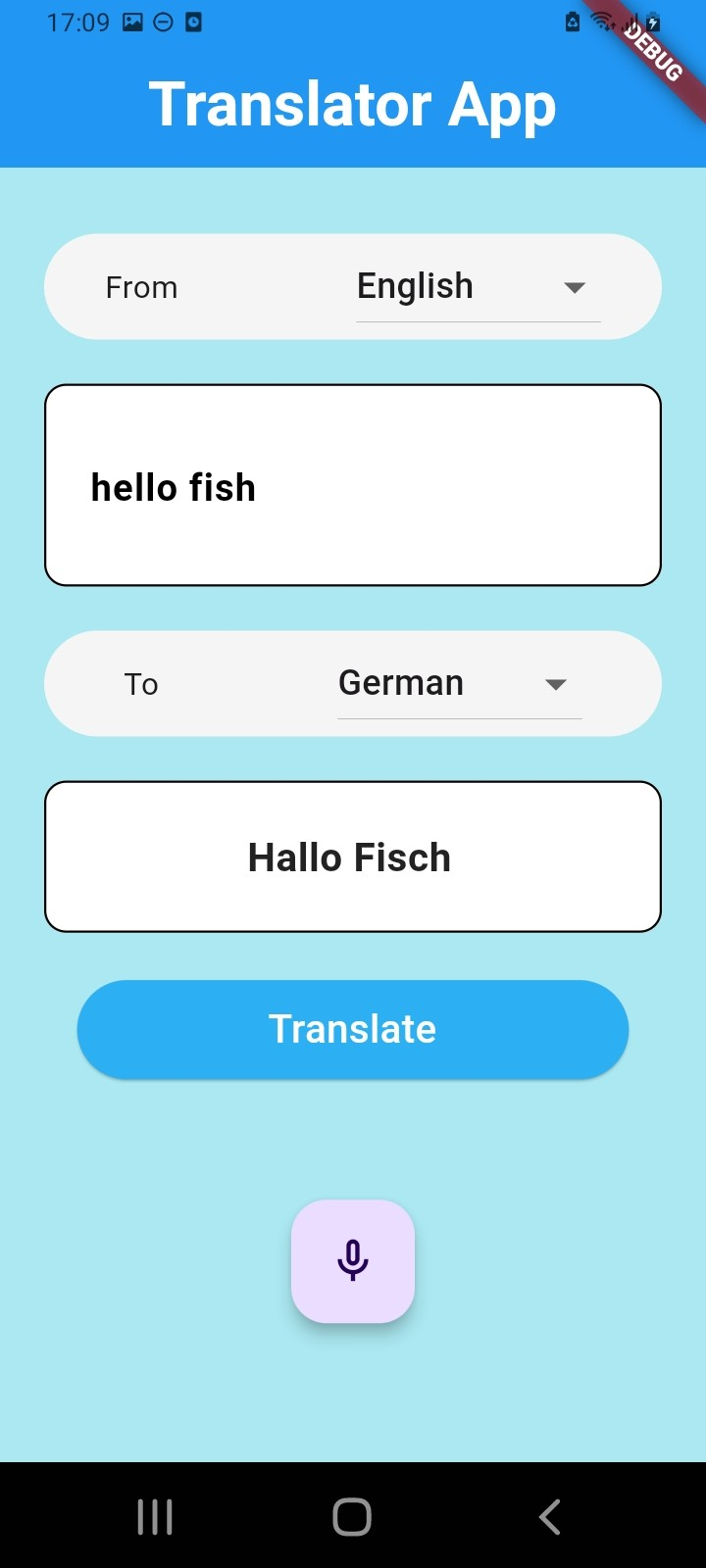
*Hình 3. 5 Dịch từ tiếng anh sang tiếng Ấn Độ*

* **Arabic**



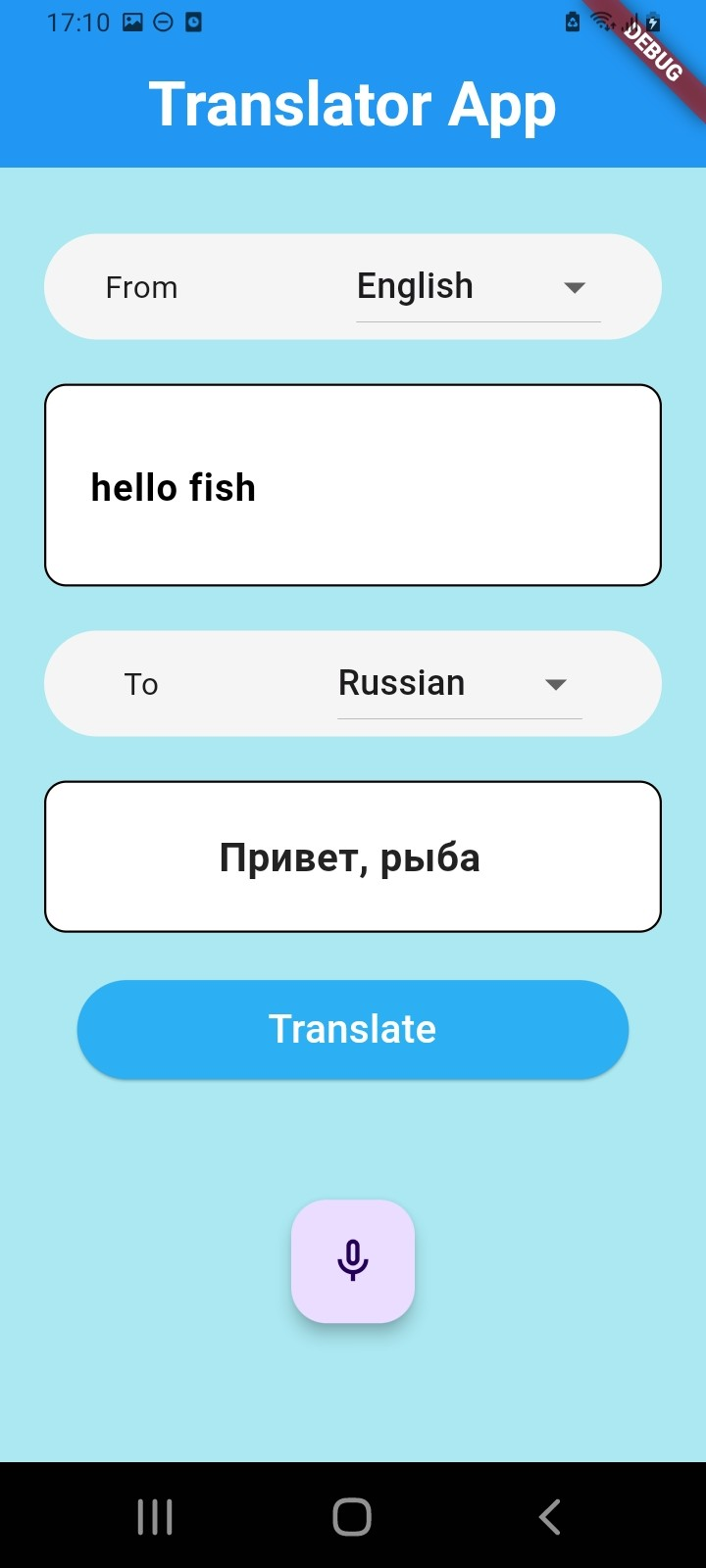
*Hình 3. 6 Dịch từ tiếng anh sang tiếng Ả Rập*

* **German**



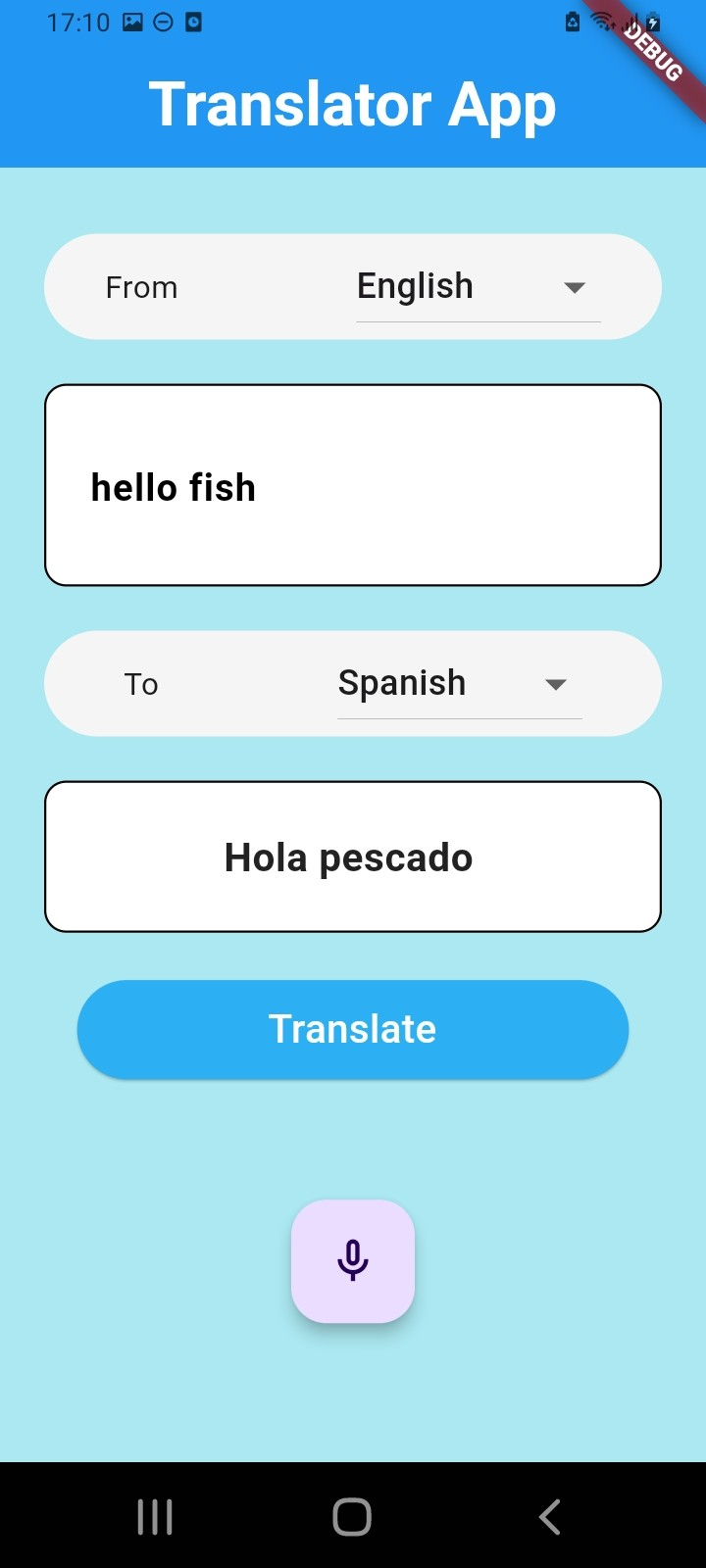
*Hình 3. 7 Dịch từ tiếng anh sang tiếng Đức*

* **Russian**



*Hình 3.8 Dịch từ tiếng anh sang tiếng Nga*

* **Spanish**



*Hình 3. 9 Dịch từ tiếng anh sang tiếng Tây Ban Nha*

* **Urdu**



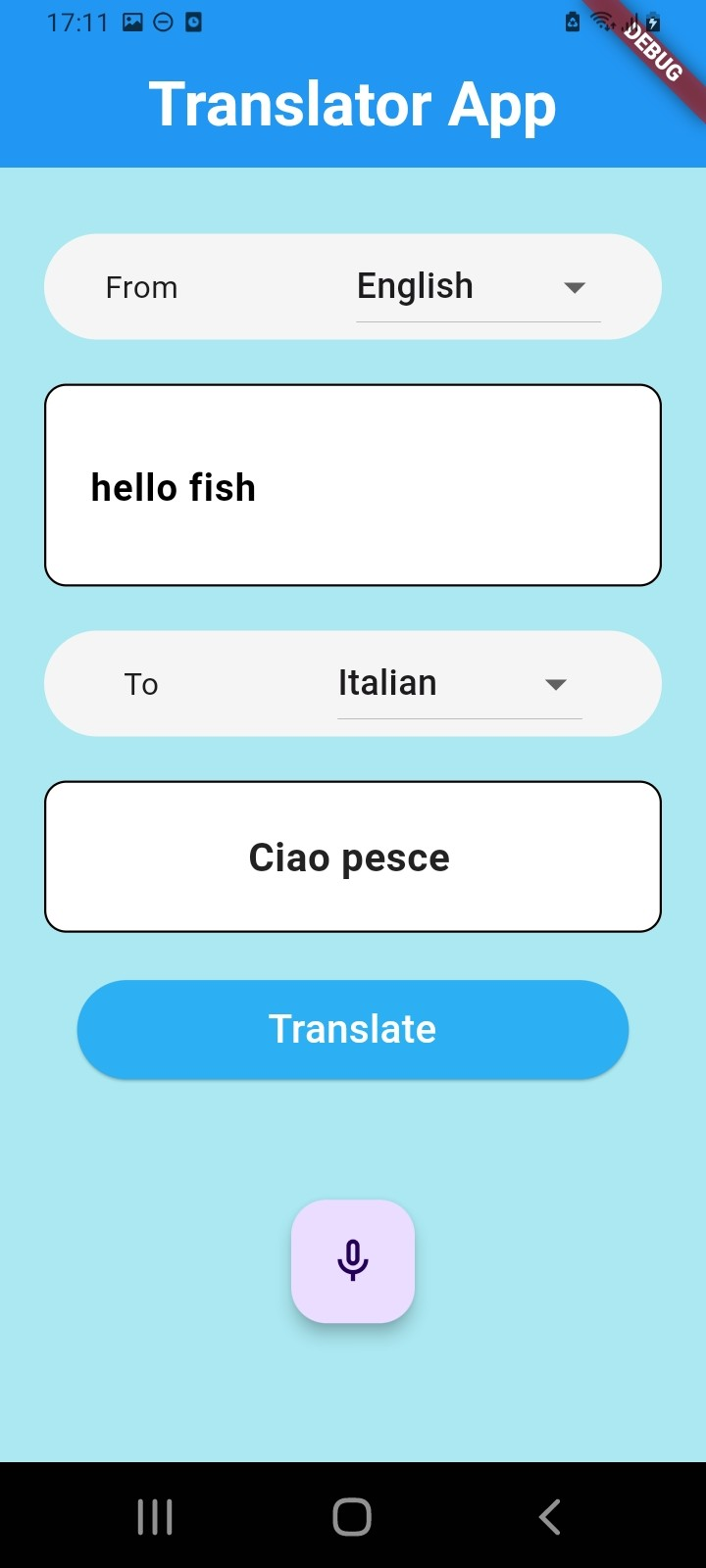
*Hình 3. 10 Dịch từ tiếng anh sang tiếng Pakistan*

* **Japanese**



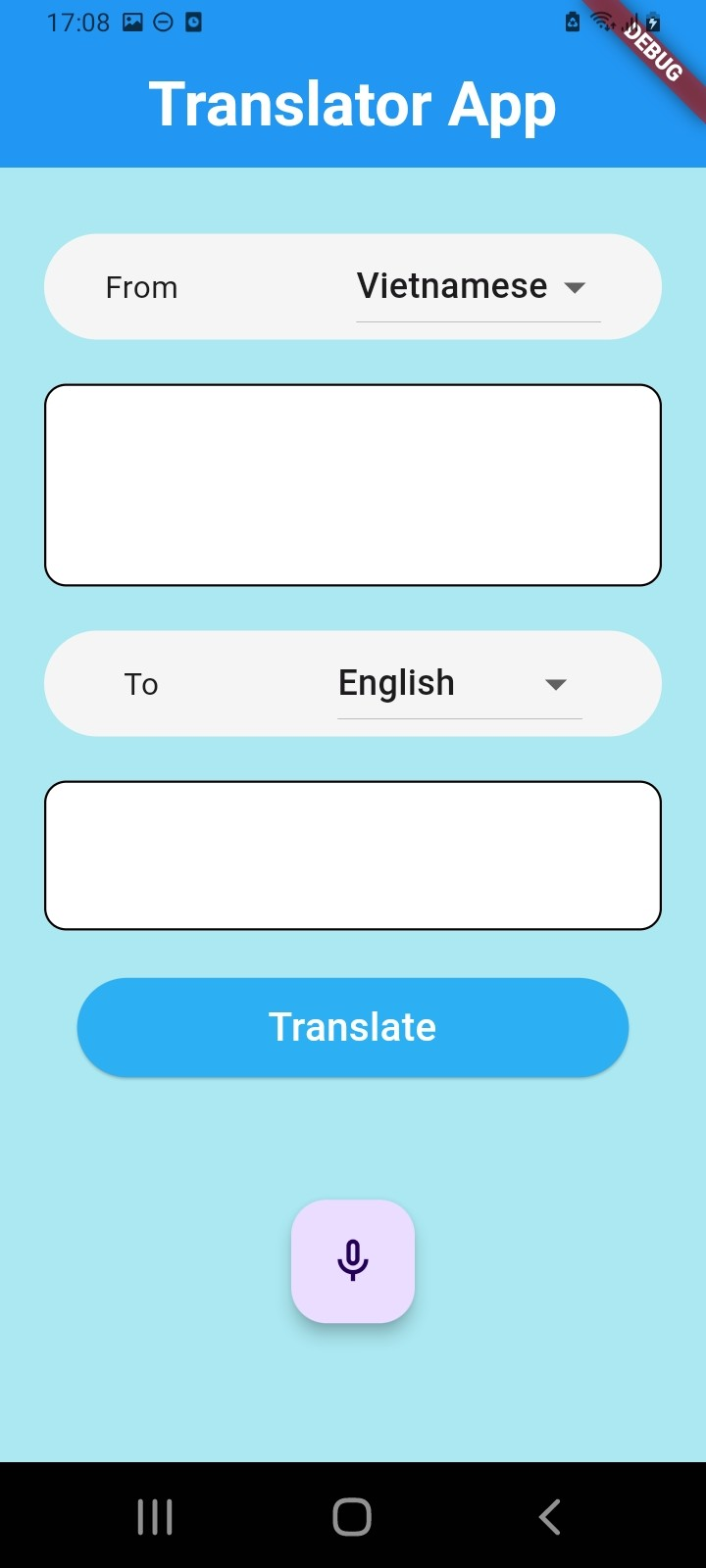
*Hình 3. 11 Dịch từ tiếng anh sang tiếng Nhật*

* **Italian**

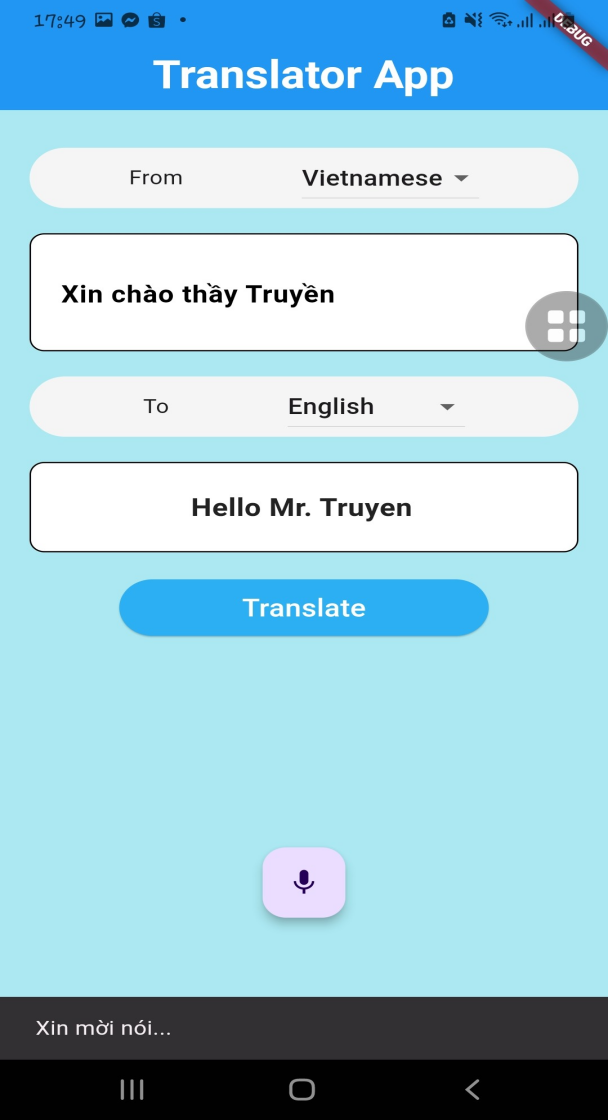
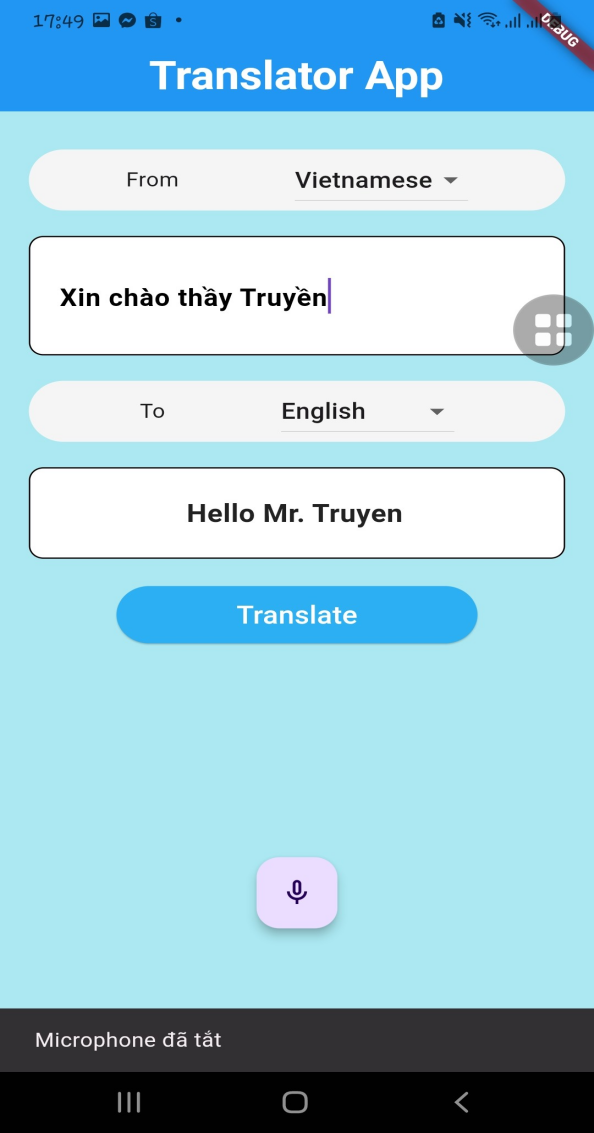


*Hình 3. 12 Dịch từ tiếng anh sang tiếng Ý*

**3.4 Dịch thuật bằng giọng nói**

Người dùng nhấp vào biểu tượng  rồi nói những gì cần dịch sang ngôn ngữ mình muốn

* **Kết quả:**

*Hình 3. 13 Bật mic để nói Hình 3. 14 Kết quả sau khi nói xong*

# CHƯƠNG 4: TỔNG KẾT

## 4.1. Kết quả đạt được

- Hoàn thành đề tài “Xây Dựng Ứng Dụng Dịch Thuật” với giao diện đơn giản, dễ dùng, dễ tiếp cận mang lại hiệu quả cao cho người sử dụng.

- Cung cấp khả năng dịch qua giọng nói.

## 4.2. Hạn chế

- Người dùng cần phải kết nối mạng để sử dụng app.

- Chưa có chức năng dịch văn bản trích xuất từ hình ảnh.

- Các ngôn ngữ đặc biệt, ngôn ngữ kỹ thuật hoặc thuật ngữ chuyên ngành có thể khó khăn hơn để dịch một cách chính xác.

## 4.3. Hướng phát triển

Phát triển thêm các chức năng như: gợi ý chính tả để tránh việc không tìm được nghĩa của từ, lấy thông tin văn bản cần dịch từ hình ảnh, hiển thị phiên âm của ngoại ngữ.

Người dùng có thể sử dụng app khi cần dịch nghĩa những từ đơn khi không có kết nối mạng.

Sử dụng trí tuệ nhân tạo và học máy để cải thiện khả năng hiểu biết và dự đoán ngữ cảnh, từ đó tạo ra các kết quả dịch thuật tốt hơn và chính xác hơn.

## 4.4. Kết luận

Với độ chính xác và thời gian xử lý nhanh chóng yêu cầu chuyển nghĩa của người dùng, “Ứng dụng dịch thuật” là một ứng dụng tiện ích giúp người dùng tra cứu nhanh chóng, năng cao khả năng ngoại ngữ của người dùng . Và trong tương lai, ứng dụng sẽ được áp dụng rộng rãi, được nhiều người biết đến, giúp ích cho chuyên ngành ngôn ngữ hay phục vụ cho việc học tập và làm việc của người dùng.

**TÀI LIỆU THAM KHẢO**

* **Tiếng Việt**

[1] Lập trình ứng dụng di động đa nền tảng với React Native - Tác giả: Lê Trọng Nhân.

[2] Lập trình ứng dụng di động đa nền tảng với Flutter - Tác giả: Lê Minh Hoàng.

[3] Phát triển ứng dụng di động đa nền tảng với Xamarin - Tác giả: Đinh Tuấn Anh.

[4] Lập trình ứng dụng di động đa nền tảng với Ionic - Tác giả: Trần Đình Quý.

[5] Xây dựng ứng dụng di động đa nền tảng bằng HTML5, CSS3 và JavaScript - Tác giả: Nguyễn Ngọc Minh.

* **Tiếng Anh**

[1] Learning React Native: Building Native Mobile Apps with JavaScript- Author: Bonnie Eisenman.

[2] Flutter in Action - Author: Eric Windmill.

[3] Xamarin.Forms Succinctly - Author: Alessandro Del Sole.

[4] Ionic in Action: Hybrid Mobile Apps with Ionic and Angular - Author: Jeremy Wilken.

[5] Programming Mobile Apps with Ionic and Angular: A Comprehensive Guide to Building Next-Generation Mobile Applications" - Authors: Chris Griffith, Todd Halfpenny.