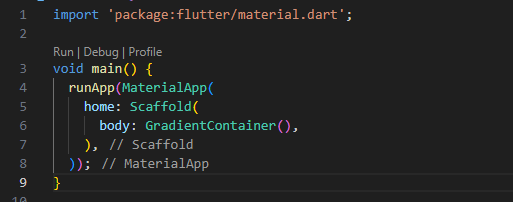
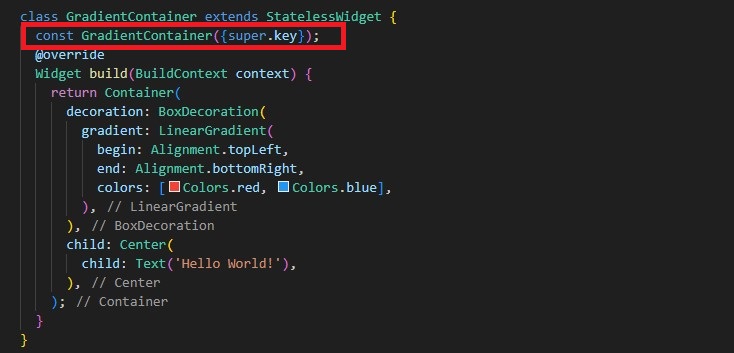
## Working with Constructor Functions



* Tạo GradientContainer để chứa cấu hình liên quan đến màu sắc.



* Tạo một class mới. Các điểm cần chú ý khi tiến hành tạo một class mới:

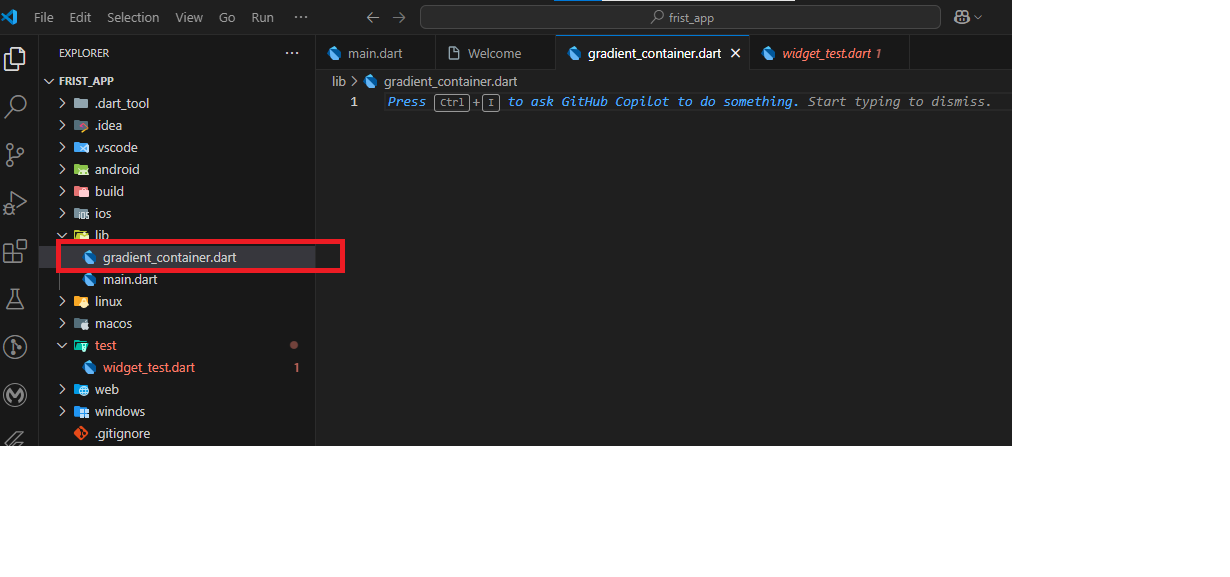
1. Lớp mới phải kế thừa **StatelessWidget**, bởi vì một class cũng phải là một Widget.

2. Lớp mới cần có một **constructor**, constructor sẽ chứa **super key** và **các biến** .

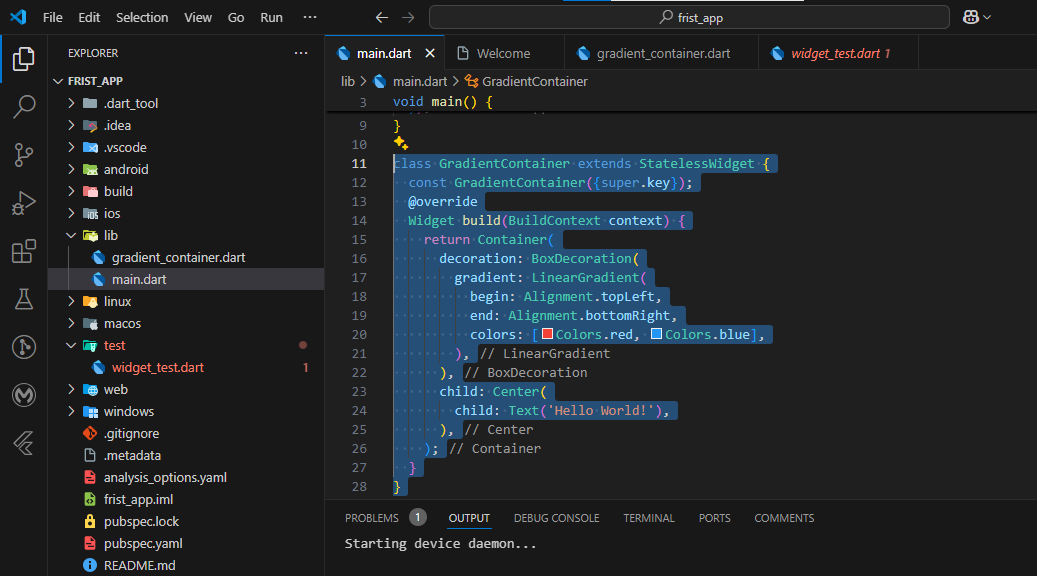
3. Cần ghi đè **@override Widget build** để tạo một widget mới.

4. Trả về - **return** widget cần tách ra, hoặc logic, cấu hình cần tạo.

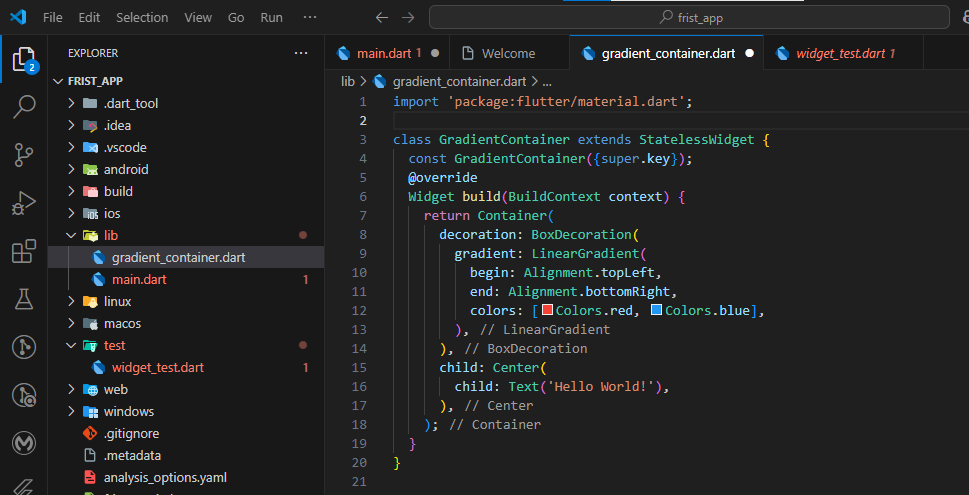
## Chia nhỏ mã nguồn (Splitting Code Across Files)

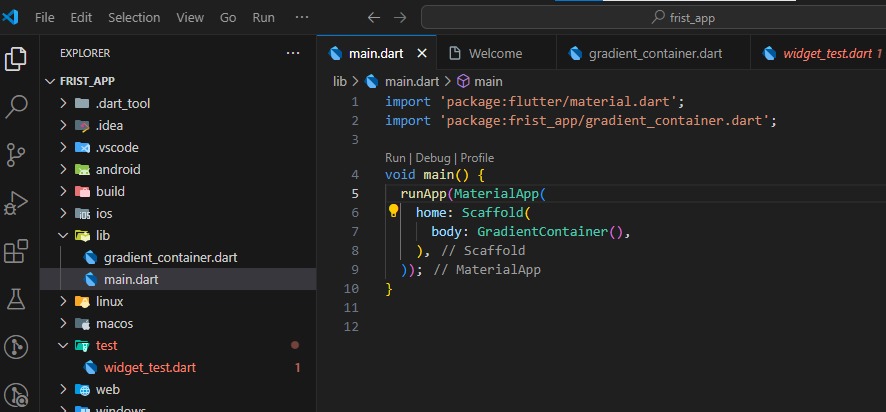


* Tạo một file mới để chuyển GradientContainer riêng.



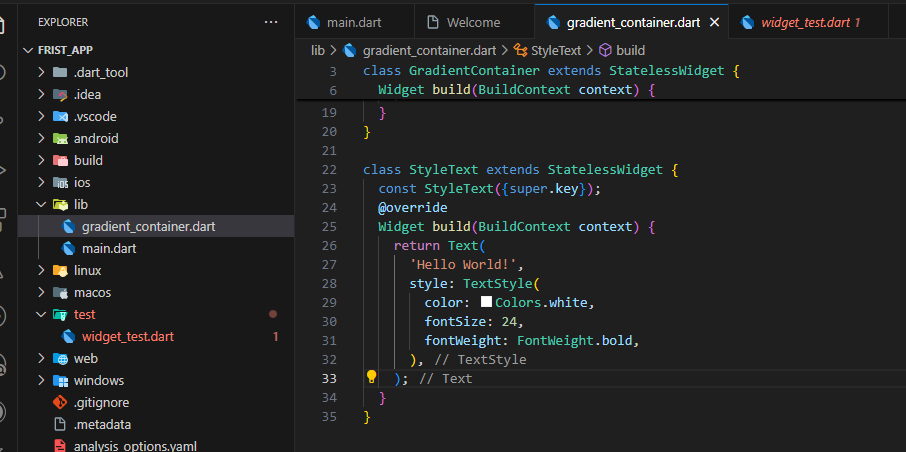
* Copy và parse GradientContainer vào.



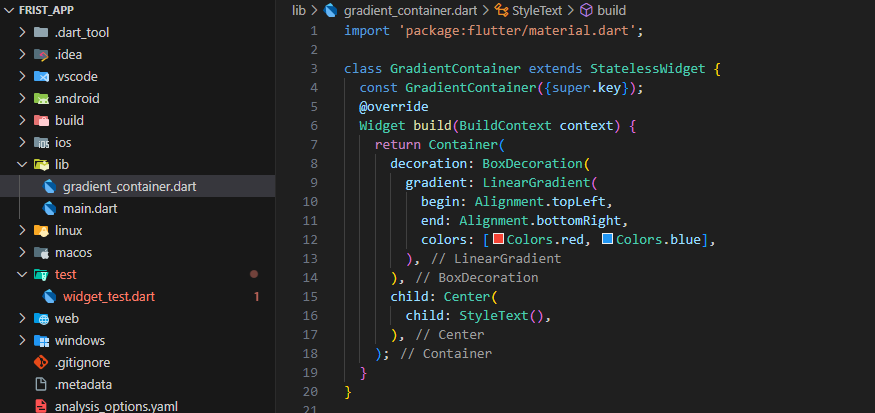


* Trong file main thêm phần import là đường dẫn của file vừa tạo.

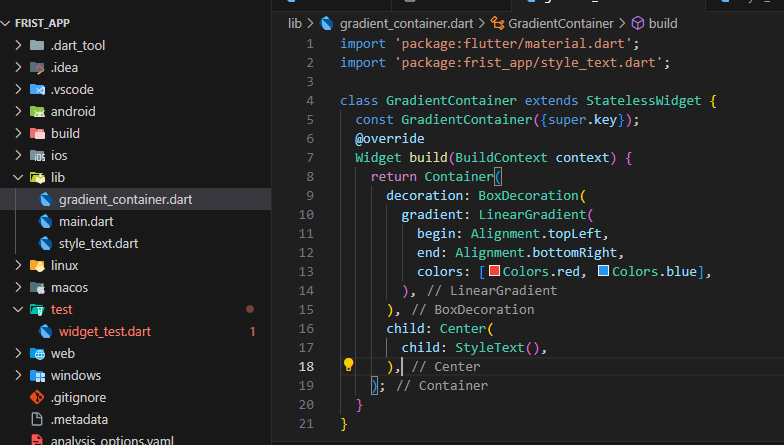
## Tạo Widget tùy chỉnh (Practice: Create a Custom Widget)



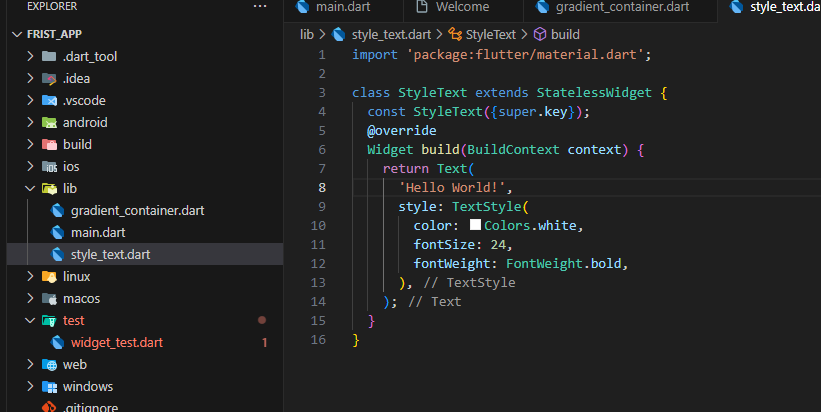
* Cần tách style text sang một class riêng và để ở một file riêng.



* Xóa phần style text trước đó và thay bằng tên lớp chuẩn bị tạo.

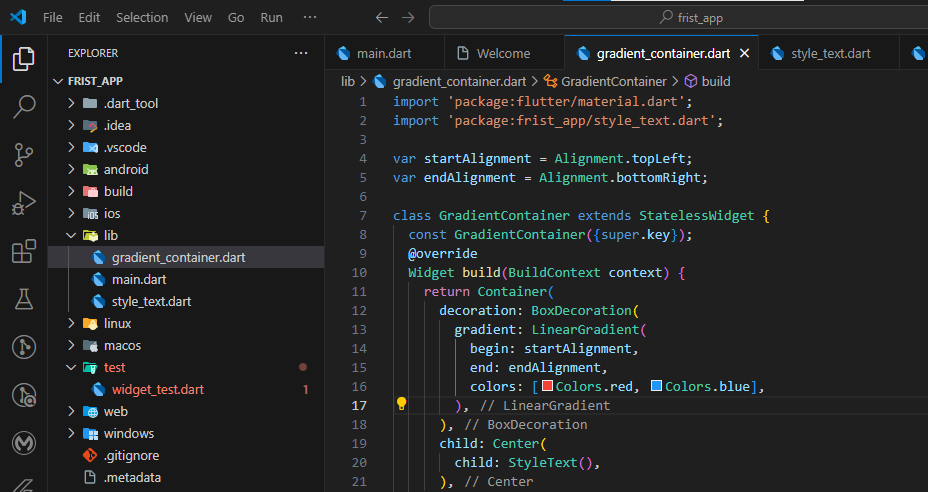


* Tạo một file có tên là style\_text.



* Tiến hành tạo class như trên hình.

## Giới thiệu về biến (Introducing Variables)



* Trong flutter ta có thể khai báo các biến để lưu trữ những nội dung cần thay đổi (var), hoặc những nội dung không thể thay đổi (const, final).

## Biến & kiểu dữ liệu - Kết hợp hai khái niệm chính (Variables & Types - Combining Two Key Concepts)

### **1. Biến và kiểu dữ liệu**

| **Khái niệm** | **Mô tả** | **Ví dụ** |
| --- | --- | --- |
| **Biến** (Variable) | Một vùng nhớ được đặt tên để lưu trữ giá trị. Giá trị có thể thay đổi trong quá trình chạy chương trình. | int age = 25; |
| **Kiểu dữ liệu** (Type) | Xác định loại dữ liệu mà một biến có thể lưu trữ, như số nguyên, chuỗi, số thực, boolean, v.v. | int, String, double, bool. |

### **2. Các kiểu dữ liệu phổ biến**

| **Kiểu dữ liệu** | **Mô tả** | **Ví dụ giá trị** |
| --- | --- | --- |
| **int** | Lưu trữ số nguyên (không có phần thập phân). | 10, -5, 0 |
| **double** | Lưu trữ số thực (có phần thập phân). | 3.14, -7.89 |
| **String** | Lưu trữ chuỗi ký tự. | "Hello", 'World' |
| **bool** | Lưu trữ giá trị logic: đúng (**true**) hoặc sai (**false**). | true, false |
| **List** | Lưu trữ danh sách các phần tử cùng kiểu. | [1, 2, 3], ["a", "b"] |
| **Map** | Lưu trữ cặp giá trị dạng key-value (khóa-giá trị). | {"key": "value"} |
| **dynamic** | Lưu trữ bất kỳ kiểu dữ liệu nào. Có thể thay đổi kiểu trong quá trình chạy chương trình. | var x = 5; x = "text"; |

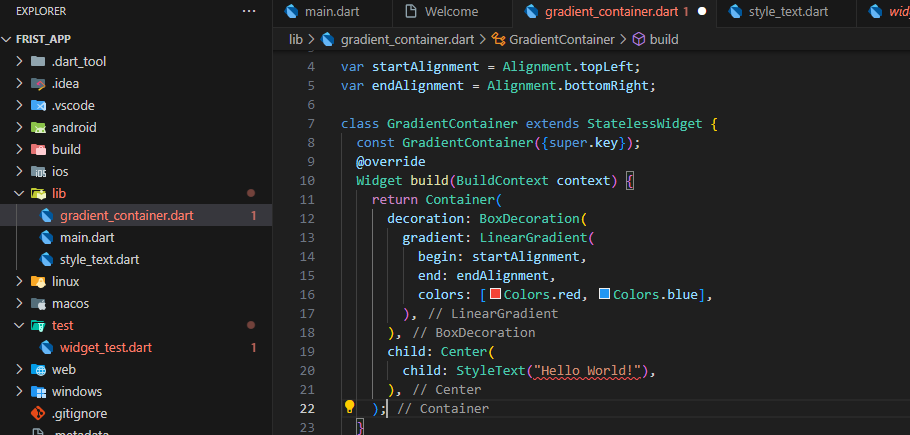
### **3. Cách khai báo biến kết hợp với kiểu dữ liệu**

| **Cách khai báo** | **Mô tả** | **Ví dụ** |
| --- | --- | --- |
| **Cụ thể kiểu dữ liệu** | Xác định rõ kiểu dữ liệu khi khai báo biến. | int age = 25; |
| **Sử dụng từ khóa var** | Dart tự suy ra kiểu dữ liệu dựa trên giá trị khởi tạo. | var name = "Alice"; |
| **Sử dụng từ khóa dynamic** | Biến có thể thay đổi kiểu dữ liệu trong quá trình chạy chương trình. | dynamic value = 10; value = "text"; |
| **Khai báo hằng số** | Sử dụng từ khóa final hoặc const để tạo hằng số (giá trị không thể thay đổi). | final pi = 3.14; const appName = "My App"; |

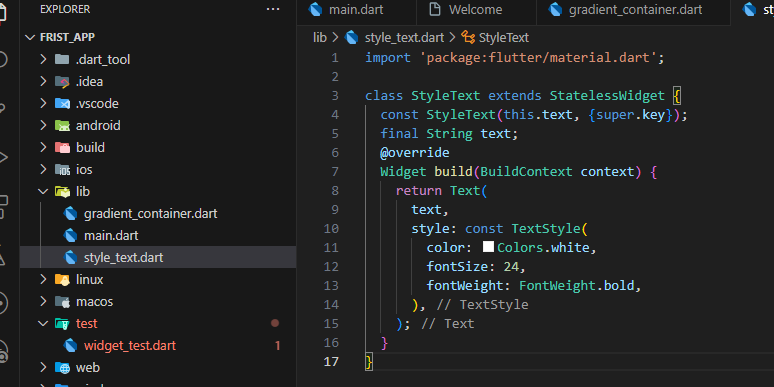
## "final" & "const" - Các loại biến đặc biệt (Special Kinds Of "Variables")

| **Đặc điểm** | **final** | **const** |
| --- | --- | --- |
| **Khởi tạo** | Chỉ khởi tạo một lần, tại runtime. | Phải khởi tạo tại compile-time. |
| **Thay đổi giá trị** | Không thể thay đổi giá trị sau lần gán. | Hoàn toàn bất biến, kể cả nội dung. |
| **Thời điểm xác định** | Runtime (khi chương trình chạy). | Compile-time (trước khi chạy chương trình). |
| **Ứng dụng** | Bảo vệ tham chiếu hoặc giá trị tại runtime. | Định nghĩa giá trị bất biến cố định. |

## Biến thực thể (Instance Variables) & Widget cấu hình (Configurable Widgets)



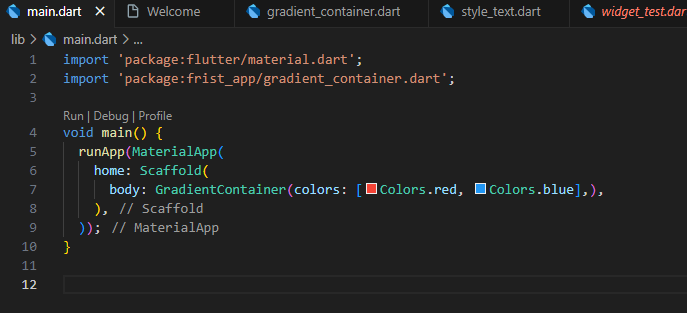
* Ta muốn có thể truyền tham số vào StyleText để có thể linh động thay đổi nội dung.



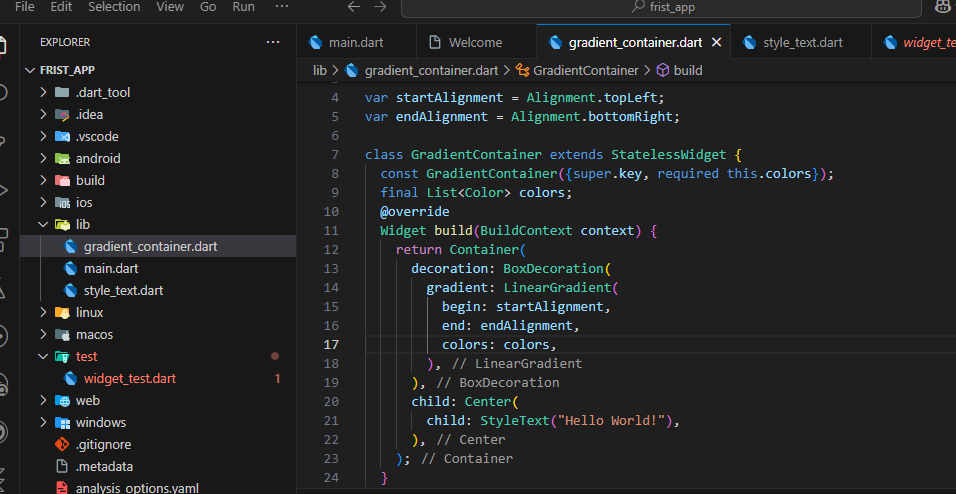
* Trong lớp StyleText ta sẽ tạo một biến final String text và một tham chiếu this.text để lấy giá trị được truyền từ bên ngoài lớp.
* Sau đó là truyền text vào phần nội dung cần thay đổi

**Lưu ý**: sẽ không thể **return const Text(…)** vì lúc này trong nội dung đã có phần có thể thay đổi.

## Thực hành: Widget tái sử dụng & hàm khởi tạo (Reusable Widgets & Constructor Functions)

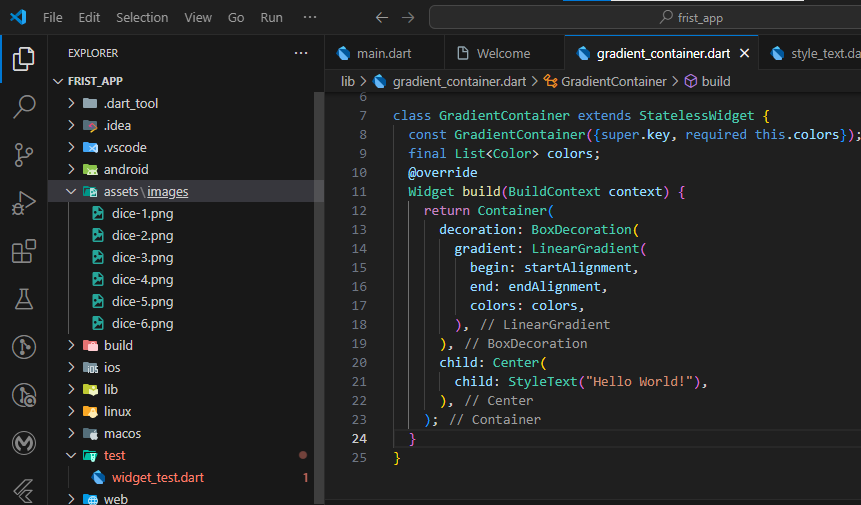


* Ta sẽ tiến hành tạo tính năng đổi màu cho lớp GradientContainer bằng cách truyền mảng màu vào lớp.

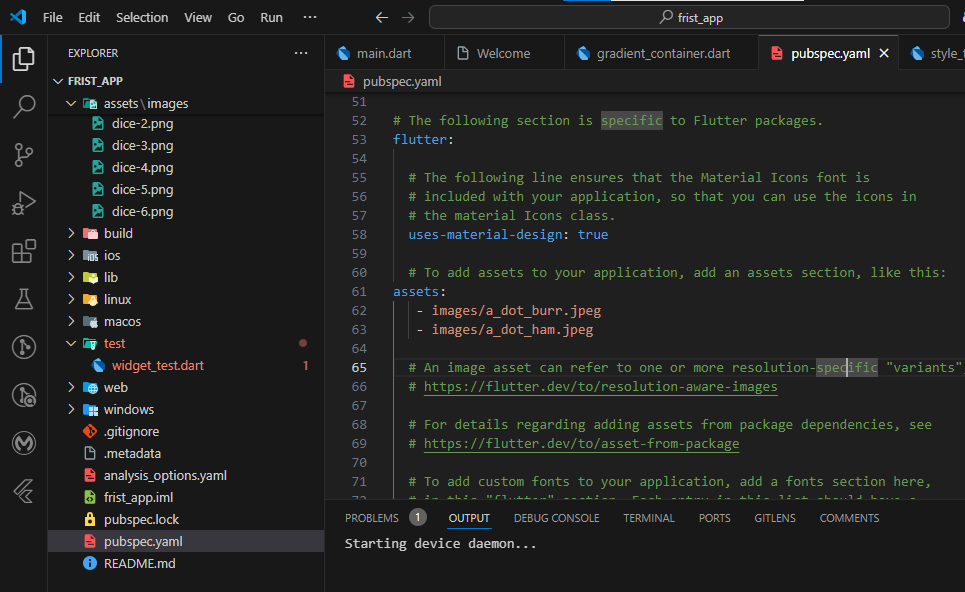


* Sử dụng **List<Color**>. Hoặc có thể sử dụng các Color riêng lẻ.

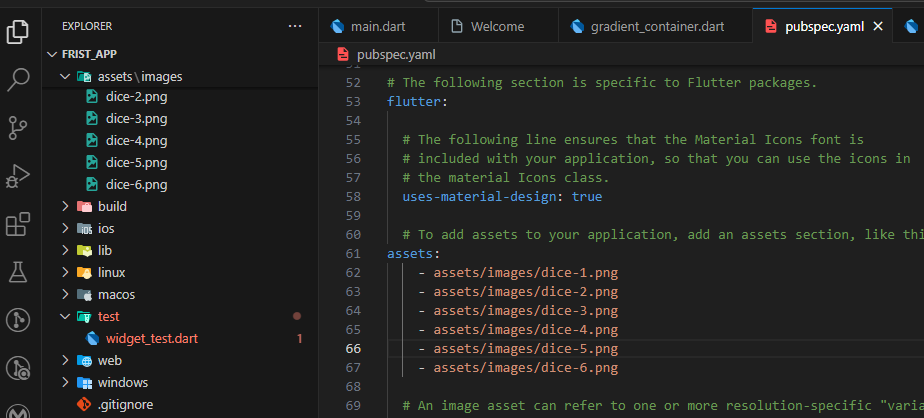
## Hiển thị hình ảnh & sử dụng nhiều hàm khởi tạo (Displaying Images & Using Multiple Constructor Functions)



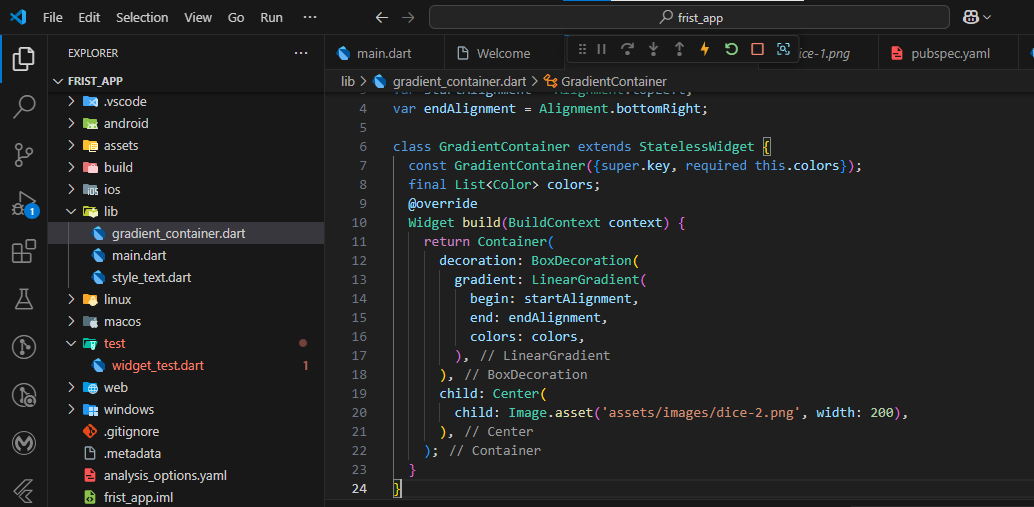
* Tạo một file để lưu trữ các hình ảnh cần.



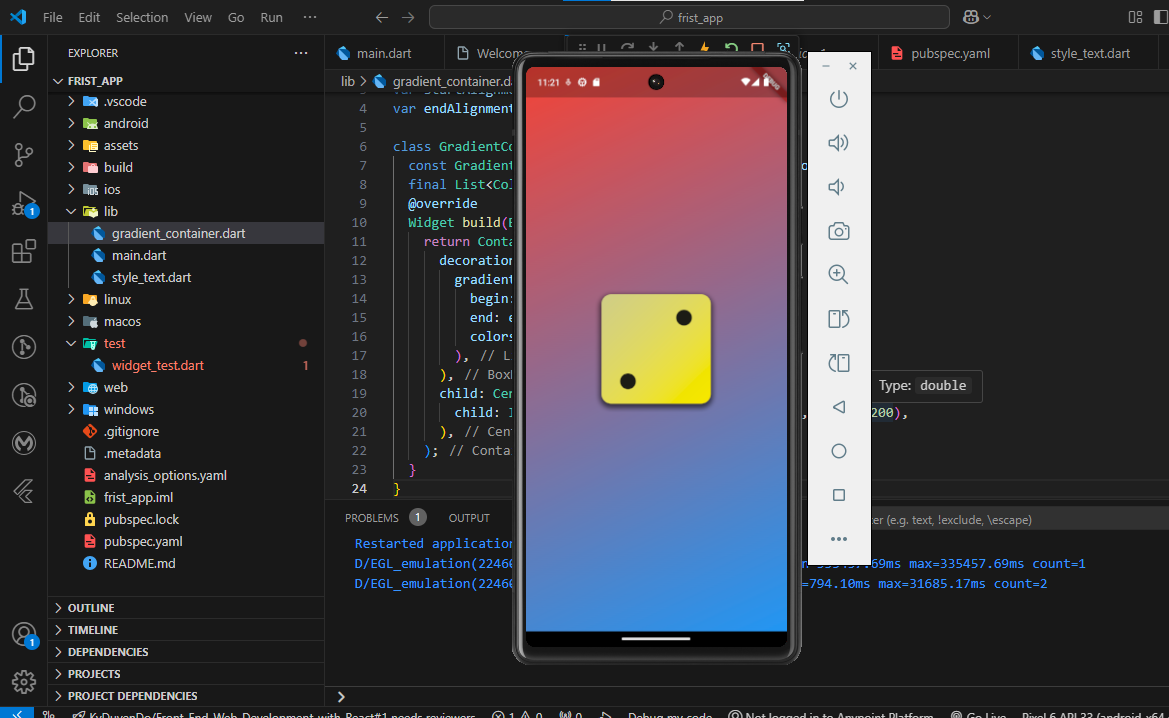
* Trong file pubspec.yaml, tìm phần assets, tắt comment và ghi các đường dẫn phần ảnh để tạo trước đó.



* Kết quả ghi.



* Trong file gradient\_container thay phần text thành image.

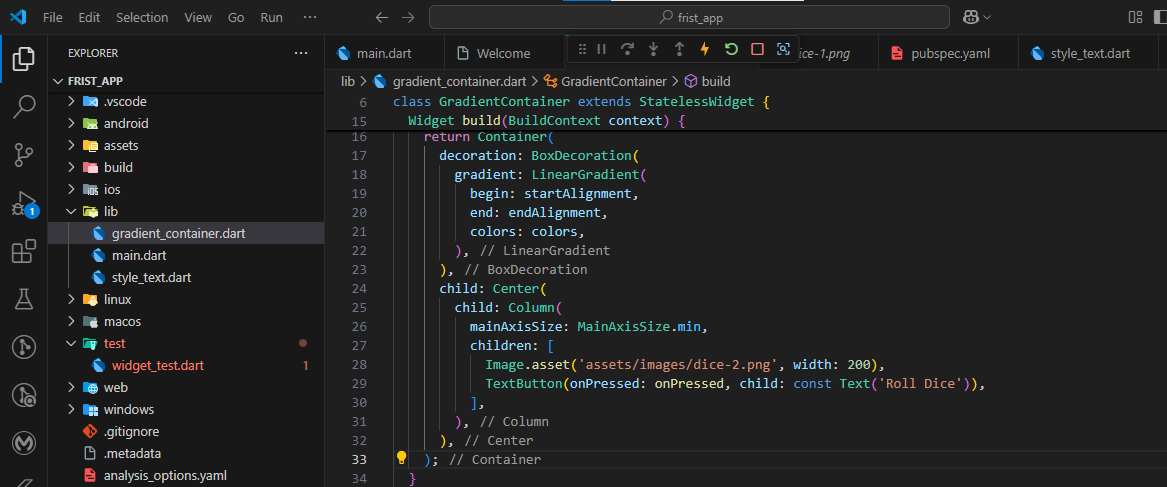


* Kết quả hiển thị như hình.

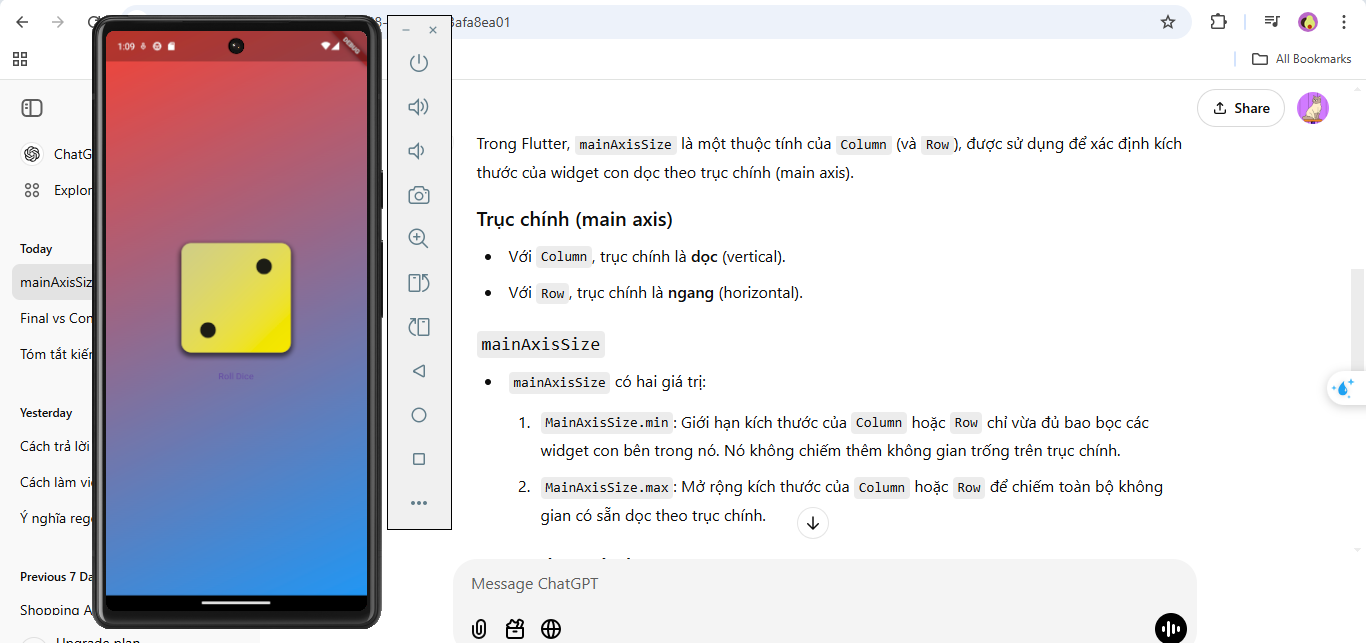
## Thêm nút & sử dụng hàm làm giá trị (Adding Buttons & Using Functions As Values)



* Các nội dung về Column và Row.

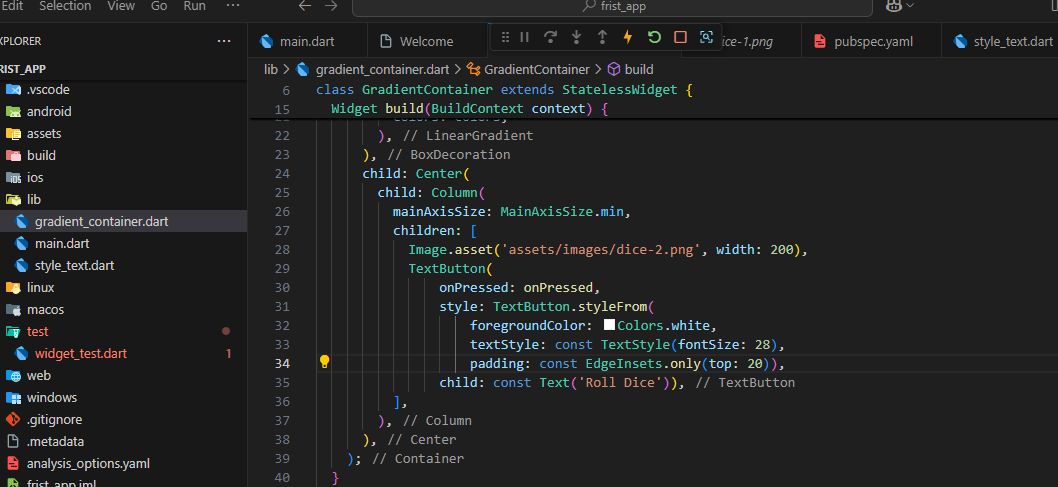


* Sử dụng **Column** để tạo dice và button theo chiều dọc
* **MainAxisSize.min** sẽ lấy chiều dọc vừa với kích thước của hai phần tử.
* Trong phần **children** chứa các phần tử cần hiển thị là **image** và **buttton**.

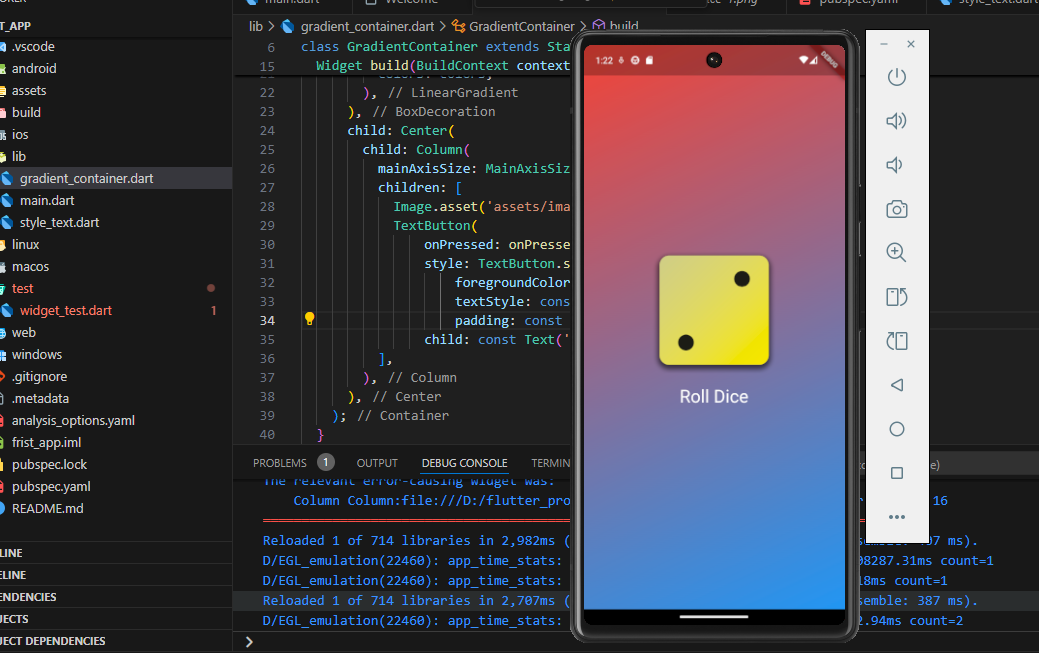


* Kết quả thực thi.

## Tùy chỉnh nút & làm việc với Padding (Styling Buttons & Working with Padding)

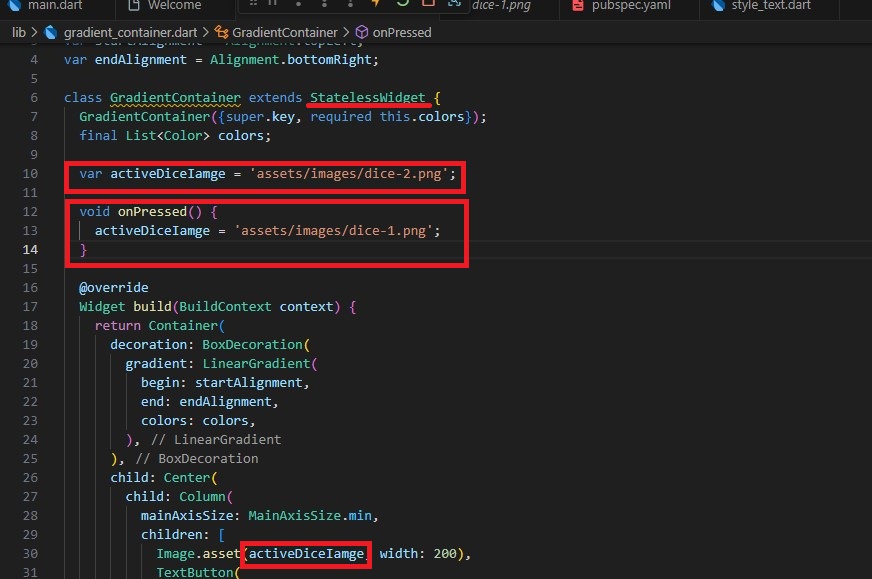


* Thêm vào **padding** và **button**. Nhưng sau đó hãy chuyển sang **sizedBox(Height: 20)** và đặt ngang với image và button.
* Thêm các style cho button, dùng **style** > **TextButton.styleFrom**.



* Kết quả như hình.

## Cách không nên xây dựng Widget tương tác (How NOT To Build Interactive Widgets)

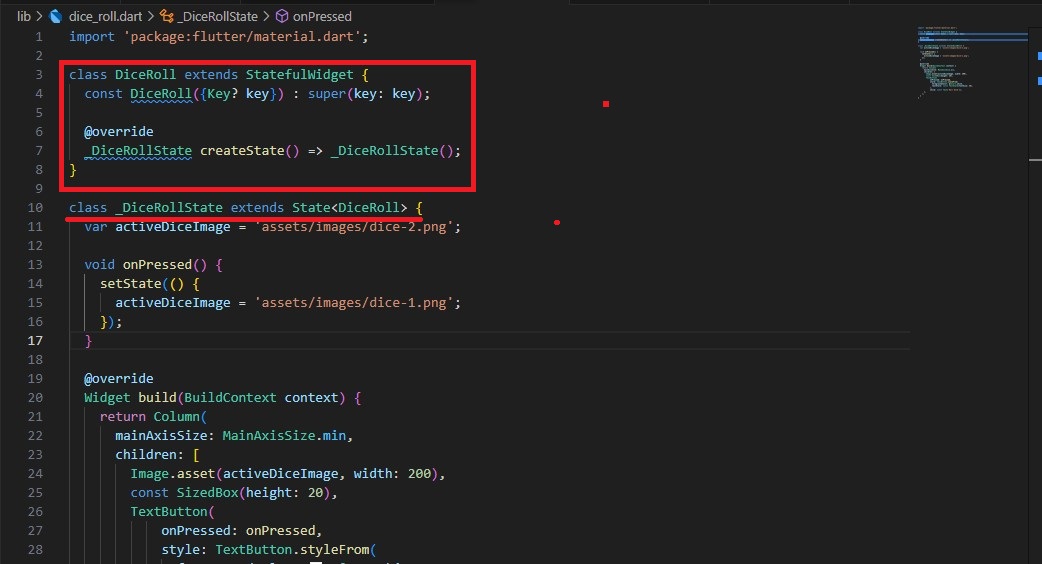


* Ta muốn tạo một biến để cập nhật lại hình ảnh số nút khi roll dice, tiến hành tạo như hình:
  + Tạo một biến để lưu đường dẫn
  + Tạo một hàm và trong đó gán lại giá trị của biến
  + Thay biến cho đường dẫn image trong children
* Kết quả không hoạt động bởi vì là **Stateless Widget**

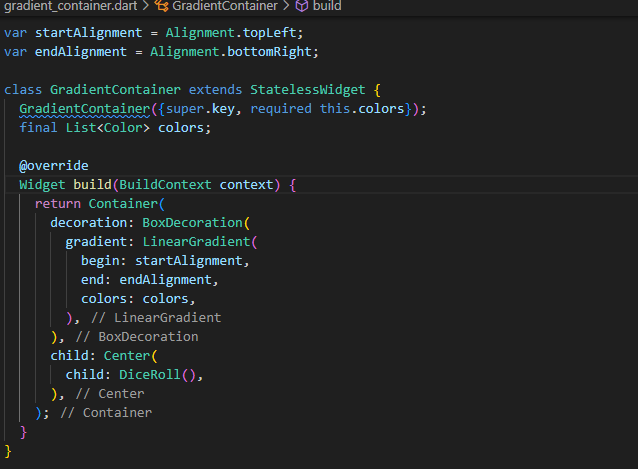
## Giới thiệu Stateful Widget (Introducing Stateful Widgets)

Trong Flutter, các **widget** được chia thành hai loại chính:

* **Stateless Widget**: Không thay đổi trạng thái trong suốt vòng đời của nó.
* **Stateful Widget**: Có khả năng thay đổi trạng thái trong suốt vòng đời và có thể cập nhật giao diện khi trạng thái thay đổi.

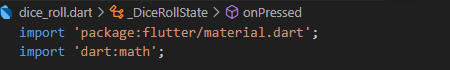


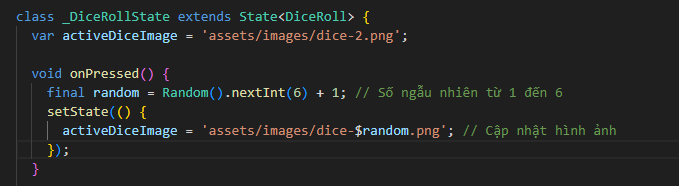
* Tạo một file mới có tên là dice\_roll ở đây sẽ tạo một Statefull widget.
  + DiceRoll là một **Stateful Widget**, được tạo ra để quản lý trạng thái thay đổi của hình ảnh xúc xắc.
  + **\_DiceRollState** là nơi quản lý trạng thái và logic thay đổi của widget.



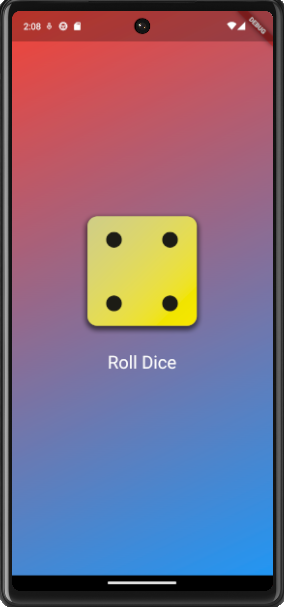
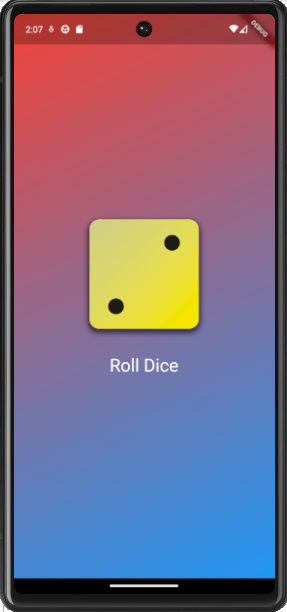
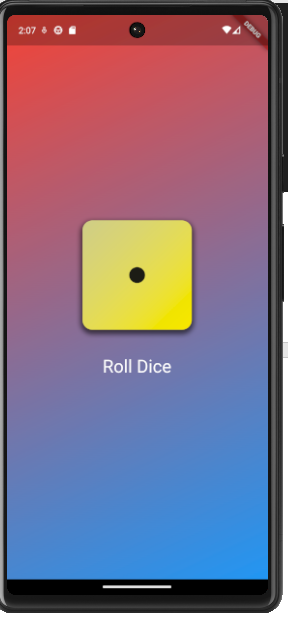
* Thay thế toàn bộ column bằng widget mới vừa tạo.

## Sinh số ngẫu nhiên (Generating Random Numbers)





* Trong dice\_roll cập nhật onPressed với random và đồng thời import vào package math.



* Kết quả thực thi ứng dụng mỗi khi bấm Roll Dice xúc xắc sẽ thay đổi.