**CÔNG TY CÔNG NGHỆ & KỸ THUẬT STECH**

**VIỆN NGHIÊN CỨU & SÁNG TẠO KHOA HỌC MÁY TÍNH – IRICS**

**A logo with orange letters

Description automatically generated**

**BÁO CÁO NGHIÊN CỨU FLUTTER**

**TASK 8 – NGHIÊN CỨU FLUTTER**

Người hướng dẫn**: MR. NGUYỄN TẤN HÙNG ANH**

**MR. VÕ THÀNH LUÂN**

Người thực hiện**: ĐINH NGỌC AN THƠ**

**THÀNH PHỐ HỒ CHÍ MINH NĂM 2024**

**TÓM TẮT**

Flutter là một framework mã nguồn mở do Google phát triển, dùng để xây dựng các ứng dụng đa nền tảng (iOS, Android, Web, và Desktop) từ một cơ sở mã nguồn duy nhất. Ra mắt lần đầu vào tháng 5 năm 2017, Flutter nhanh chóng trở thành một công cụ phổ biến nhờ khả năng giúp lập trình viên tiết kiệm thời gian và công sức khi phát triển ứng dụng cho nhiều nền tảng. Flutter sử dụng ngôn ngữ lập trình Dart, cũng do Google phát triển, nổi bật với khả năng biên dịch nhanh và hiệu suất cao.

Một trong những điểm mạnh của Flutter là khả năng tạo ra giao diện người dùng đẹp và linh hoạt. Nó cung cấp một bộ widget phong phú và có thể tùy chỉnh, giúp các nhà phát triển dễ dàng xây dựng giao diện ứng dụng theo ý muốn. Flutter cũng hỗ trợ tính năng "hot reload", cho phép lập trình viên xem ngay lập tức các thay đổi trong mã nguồn mà không cần khởi động lại toàn bộ ứng dụng, từ đó tăng hiệu quả phát triển.

Flutter còn nổi bật với hiệu suất cao nhờ việc sử dụng đồ họa Skia, giúp ứng dụng chạy mượt mà và hiển thị đồ họa phức tạp một cách dễ dàng. Hệ sinh thái của Flutter cũng rất phong phú với nhiều thư viện và plugin hỗ trợ, giúp mở rộng chức năng của ứng dụng một cách linh hoạt. Nhờ các ưu điểm này, Flutter đã được nhiều công ty và lập trình viên trên toàn thế giới tin dùng để xây dựng các ứng dụng di động và đa nền tảng.

**MỤC LỤC**

[CHƯƠNG I: MAIN.DART 5](#_Toc170219121)

[CHƯƠNG II: HOME\_PAGE.DART 6](#_Toc170219122)

[CHƯƠNG III: MEDIA\_PROVIDER.DART 9](#_Toc170219123)

[CHƯƠNG IV : VIDEO\_VIEW.DART 12](#_Toc170219124)

[CHƯƠNG V : WEBRTC\_SERVICE.DART 14](#_Toc170219125)

[CHƯƠNG VI: TỔNG HỢP 15](#_Toc170219126)

**DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT**

**DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ**

[Hình 1: class trong Main 5](#_Toc170219223)

[Hình 2 : home\_page.dart 6](#_Toc170219224)

[Hình 3 : home\_page 7](#_Toc170219225)

[Hình 3: media\_provider 9](#_Toc170219226)

[Hình 4: video\_view 12](#_Toc170219227)

*Hình 5 : Webrtc\_service…………………………………………………………………………………..14*

CHƯƠNG I: MAIN.DART

**A screen shot of a computer program

Description automatically generated**

Hình 1: class trong Main

- flutter/material.dart : thư viện cơ bản để xây dựng giao diện người dùng

‘provider’ : gói quản lý trạng thái trong ứng dụng Flutter

‘home\_page.dart’ và ‘media\_provider.dart’ : các file dart khác nhau trong project

- changeNotifierProvider : widget của ‘provider’ để cung cấp đối tượng ‘MediaProvider’ cho cây widget của ứng dụng.

- MaterialApp là widget gốc của ứng dụng, cung cấp nhiều tính năng của Flutter Material Design.

- title là tiêu đề của ứng dụng.

- theme xác định chủ đề của ứng dụng với màu chủ đạo là màu xanh (Colors.blue) và mật độ hiển thị theo nền tảng (VisualDensity.adaptivePlatformDensity).

- home : xác định trang chủ của ứng dụng, trong trường hợp này là HomePage với tiêu đề 'WebRTC'.

CHƯƠNG II: HOME\_PAGE.DART

**A screen shot of a computer screen

Description automatically generated**

**A screen shot of a computer code

Description automatically generated**

Hình 2 : home\_page.dart

-flutter/material.dart cho các widget Flutter, provider để quản lý trạng thái, media\_provider.dart và video\_view.dart cho các phần khác của ứng dụng.

-HomePage là một class kế thừa từ StatelessWidget. Đây là màn hình chính của ứng dụng.

title là một thuộc tính nhận vào tiêu đề của màn hình, được khởi tạo thông qua constructor.

-Phương thức build xây dựng giao diện người dùng:

Scaffold cung cấp cấu trúc cơ bản cho một màn hình với AppBar ở trên cùng và body chứa nội dung chính.

Consumer<MediaProvider> là một widget của gói provider dùng để lắng nghe và sử dụng đối tượng MediaProvider từ context.

Column chứa các widget con xếp theo chiều dọc:

-Nếu mediaProvider.isGettingDevices là true, hiển thị thông báo và vòng tròn tải để chỉ ra rằng danh sách thiết bị đang được lấy.

Nếu không, kiểm tra mediaProvider.isLoading để quyết định có hiển thị vòng tròn tải và thông báo rằng đang lấy media từ thiết bị hay không.

Nếu không tải, hiển thị VideoView để xem video từ thiết bị thông qua mediaProvider.localRenderer.

**A screen shot of a computer program

Description automatically generated**

Hình 3 : home\_page

Hai DropdownButton<String> cho phép người dùng chọn thiết bị video và âm thanh từ danh sách các thiết bị có sẵn.

items là danh sách các DropdownMenuItem được tạo từ các thiết bị video (videoDevices) và âm thanh (audioDevices).

Khi người dùng chọn một thiết bị, hàm onChanged sẽ cập nhật thiết bị đã chọn thông qua các phương thức selectVideoDevice và selectAudioDevice.

ElevatedButton cho phép người dùng chuyển đổi giữa các camera (nếu có nhiều hơn một camera).

Khi nút được nhấn, phương thức toggleCamera của mediaProvider sẽ được gọi để chuyển đổi camera.

CHƯƠNG III: MEDIA\_PROVIDER.DART

**A screenshot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

Hình 3: media\_provider

- MediaProvider kế thừa từ ChangeNotifier để quản lý trạng thái và thông báo cho các widget lắng nghe khi trạng thái thay đổi.

\_webrtcService là một instance của WebRTCService dùng để tương tác với các dịch vụ WebRTC.

\_localRenderer là một instance của RTCVideoRenderer dùng để hiển thị video.

- Các thuộc tính để quản lý trạng thái:

\_isLoading và \_isGettingDevices để theo dõi trạng thái tải.

\_isFrontCamera để xác định camera trước hoặc sau.

\_mediaDevices để lưu danh sách các thiết bị media.

\_selectedVideoDeviceId và \_selectedAudioDeviceId để lưu ID của thiết bị video và âm thanh được chọn.

MediaProvider() {

\_initializeRenderers();

\_getMediaDevices();

\_getUserMedia();

}

Constructor của MediaProvider khởi tạo các renderers, lấy danh sách thiết bị media, và lấy media từ thiết bị.

Future<void> \_initializeRenderers() async {

await \_localRenderer.initialize();

}

\_initializeRenderers khởi tạo RTCVideoRenderer.

Future<void> \_getMediaDevices() async {

\_isGettingDevices = true;

\_mediaDevices = await \_webrtcService.getMediaDevices();

\_isGettingDevices = false;

notifyListeners();

}

-\_getMediaDevices: lấy danh sách các thiết bị media từ WebRTCService và cập nhật trạng thái.

- \_getUserMedia: lấy media từ thiết bị với các ràng buộc (constraints) và cập nhật RTCVideoRenderer với stream media mới.

-selectVideoDevice cập nhật thiết bị video được chọn và lấy media mới.

-selectAudioDevice cập nhật thiết bị âm thanh được chọn và lấy media mới.

-toggleCamera chuyển đổi giữa camera trước và sau và lấy media mới.

CHƯƠNG IV : VIDEO\_VIEW.DART

**A computer screen shot of a program code

Description automatically generated**

Hình 4: video\_view

-VideoView là một class kế thừa từ StatelessWidget. Điều này có nghĩa là VideoView không có trạng thái riêng và chỉ dựa vào các đối tượng bên ngoài được truyền vào.

-renderer là một đối tượng RTCVideoRenderer được truyền vào thông qua constructor. Đối tượng này chịu trách nhiệm phát video từ stream media.

-Phương thức build xây dựng giao diện người dùng cho widget:

* Container là một widget hộp dùng để chứa các widget con và trang trí.
* margin: EdgeInsets.zero loại bỏ khoảng cách xung quanh container.
* width và height được thiết lập bằng cách lấy kích thước chiều rộng và chiều cao của màn hình từ MediaQuery.of(context).size.

-RTCVideoView là một widget từ thư viện flutter\_webrtc dùng để hiển thị video từ RTCVideoRenderer.

-renderer là đối tượng RTCVideoRenderer được truyền vào.

-mirror: true làm cho video được hiển thị với hiệu ứng gương (ngược trái phải), thường được sử dụng khi hiển thị video từ camera trước.

-decoration: BoxDecoration(color: Colors.black54) thiết lập nền của container là màu đen với độ trong suốt 54%.

CHƯƠNG V : WEBRTC\_SERVICE.DART

**A screen shot of a computer code

Description automatically generated***Hình 5 : Webrtc\_service*

-WebRTCService là một class cung cấp các phương thức để tương tác với các thiết bị media.

-getMediaDevices là một phương thức bất đồng bộ (async) trả về một danh sách các đối tượng MediaDeviceInfo đại diện cho các thiết bị media có sẵn trên thiết bị.

-navigator.mediaDevices.enumerateDevices() là một phương thức của WebRTC API dùng để liệt kê tất cả các thiết bị media (ví dụ: camera, microphone) có sẵn.

-getUserMedia là một phương thức bất đồng bộ (async) trả về một MediaStream chứa dữ liệu từ các thiết bị media dựa trên các ràng buộc (mediaConstraints) được cung cấp.

-mediaConstraints là một map chứa các ràng buộc về audio và video để xác định cách thức và thiết bị nào sẽ được sử dụng.

-navigator.mediaDevices.getUserMedia(mediaConstraints) là một phương thức của WebRTC API dùng để yêu cầu truy cập và lấy stream media từ các thiết bị dựa trên các ràng buộc đã cung cấp

\* class ‘WebTRCService’ cung cấp hai phương thức chính:

* ‘getMediaDevices’ : lấy danh sách tất cả các thiết bị media có sẵn trên thiết bị
* ‘getUserMedia’ : lấy stream media từ các thiết bị dựa trên các ràng buộc đã cung cấp

CHƯƠNG VI: TỔNG HỢP

**-** Định nghĩa ‘MediaProvide’ để quản lý các thiết bị media ( video và âm thanh ) trong ứng dụng Flutter để sử dụng WebRTC. Class xử dụng package ‘flutter\_webrtc’ để xử lý các chức năng liên quan đến WebRTC

- ‘Videoview’ là một widget đơn giản dùng để hiển thị video từ ‘RTCVideoRenderer’ trong ứng dụng Flutter sử dụng WebRTC. Widget thiết lập kích thước theo kích thước màn hình và có nền màu đen trong suốt

-‘WebRTCservice’ dùng để tương tác với các thiết bị media ( video và âm thanh ) trên trình duyệt hoặc thiết bị di động thông qua thư viện ‘flutter\_webrtc’