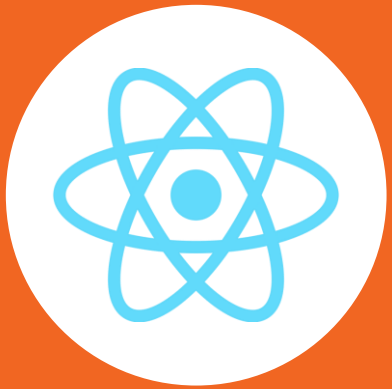
FPT Education

FPT POLYTECHNIC



LẬP TRÌNH ĐA NỀN TẢNG VỚI REACT NATIVE

BÀI 4: GIỚI THIỆU VỀ ỨNG DỤNG CHỤP ẢNH, LẤY ẢNH VÀ PHÁT NHẠC

PHẦN 1: CHỤP ẢNH VÀ LẤY ẢNH TỪ THƯ VIỆN

Mục tiêu

- [ ] Cài đặt thư viện chụp, lấy hình ảnh
- [ ] Tạo ứng dụng chụp ảnh
- [ ] Tạo ứng dụng lấy ảnh từ thư viện

Giới thiệu

- Chụp ảnh và lấy ảnh từ ứng dụng của điện thoại là một chức năng phổ biến, hầu như mọi ứng dụng nào đều có. Để có thể sử dụng được chức năng này trong ứng dụng React Native bạn cần cài đặt thư viện thứ 3, có khá nhiều thư viện hỗ trợ bạn làm điều này. Ở bài học này sẽ sử dụng thư viện **react-native-image-picker**

  - **react-native-image-picker** tương thích với iOS/Android với hỗ trợ máy ảnh, video.

Cài đặt thư viện

- Chạy lệnh sau để cài đặt thư viện

```
npm i react-native-image-picker --save
```

- Với thiết bị Android, bạn không cần phải khai báo cấp quyền (**saveToPhotos** yêu cầu kiểm tra quyền).

launchCamera()

```
import {launchCamera} from 'react-  native-image-picker';
```

- Khởi động camera để chụp ảnh hoặc quay video

```
launchCamera(options?, callback);

// Bạn có thể sử dụng một promise mà không cần 'callback':
const result = await launchCamera(options?);
```

launchImageLibrary()

```
import {launchImageLibrary} from 'react-  native-image-picker';
```

- Khởi chạy thư viện để chọn hình ảnh hoặc video

```
launchImageLibrary(options?, callback)

// You can also use as a promise without 'callback':
const result = await launchImageLibrary(options?);
```

Options

- Tuỳ vào mục đích sử dụng camera, hay chọn ảnh từ thư viện thì bạn sẽ có những tuỳ chỉnh trong options của module. Dưới đây là danh sách các options tuỳ chỉnh của **react-native-image-picker**

Option	Mô tả
mediaType	**photo** hoặc **video** hoặc **mixed** (**launchCamera** trên Android không hỗ trợ '**mixed**').
maxWidth	Để thay đổi kích thước hình ảnh
maxHeight	Để thay đổi kích thước hình ảnh

Options

videoQuality	**low**, **medium**, hoặc **high** trên iOS, **low** hoặc **high** trên Android.
durationLimit	Thời lượng tối đa của video (tính bằng giây)
quality	0 đến 1
cameraType	'**back**' or '**front**' (Có thể không hỗ trợ trên một số thiết bị Android).
includeBase64	Nếu **true**, hãy tạo chuỗi base64 của hình ảnh (Tránh sử dụng trên các tệp hình ảnh lớn do ảnh hưởng đến performance).

Options

includeExtra	Nếu đúng, sẽ bao gồm dữ liệu bổ sung, yêu cầu quyền thư viện.
saveToPhotos	Chỉ để khởi chạy **launchCamera**, lưu tệp hình ảnh / video được chụp vào ảnh công khai.
selectionLimit	Hỗ trợ cung cấp bất kỳ giá trị số nguyên nào. Sử dụng 0 để cho phép bất kỳ số lượng tệp nào trên iOS phiên bản > = 14 & Android phiên bản > = 13. Mặc định là 1.
presentationStyle	Kiểm soát cách trình bày bộ chọn. **currentContext**, **pageSheet**, **fullScreen**, **formSheet**, **popover**, **overFullScreen**, **overCurrentContext**. Mặc đinh là currentContext.

Giá trị trả về

- Khi bạn gọi hàm **launchCamera** hoặc **launchImageLibrary.** Kết quả trả về sẽ là một object chứa các thông tin sau:
  - **didCancel**: **true** nếu người dùng hủy quá trình
  - **errorCode**: Kiểm tra **ErrorCode** cho tất cả các mã lỗi

Code	Mô tả
camera_unavailable	Máy ảnh không có sẵn trên thiết bị
permission	Quyền không được đáp ứng
others	Các lỗi khác (kiểm tra errorMessage để biết mô tả)

Giá trị trả về

- **errorMessage**: Mô tả lỗi, chỉ sử dụng nó cho mục đích gỡ lỗi
- **assets**: Mảng media đã chọn.

key	Photo/Video	Mô tả
base64	PHOTO ONLY	Chuỗi base64 của hình ảnh
uri	BOTH	Uri tệp trong bộ nhớ cache dành riêng cho ứng dụng. Ngoại trừ khi chọn video từ thư viện Android, nơi bạn sẽ nhận được uri nội dung chỉ đọc, để lấy uri tệp trong trường hợp này, hãy sao chép tệp vào bộ nhớ cụ thể của ứng dụng bằng bất kỳ thư viện react-native nào.

Giá trị trả về

originalPath **(Android)**	BOTH	Đường dẫn tập tin gốc
width	BOTH	Kích thước của hình ảnh
height	BOTH	Kích thước của hình ảnh
fileSize	BOTH	Kích thước tệp
type	BOTH	Loại tệp
fileName	BOTH	Tên tệp

Giá trị trả về

duration	VIDEO ONLY	Thời lượng video đã chọn tính bằng giây
bitrate (**Android**)	VIDEO ONLY	Tốc độ bit trung bình (tính bằng bit/giây) của video đã chọn, nếu có.
timestamp	BOTH	Timestamp của file. Chỉ được bao gồm nếu '**includeExtra**' là **true**
id	BOTH	Số nhận dạng cục bộ của ảnh hoặc video. Trên Android, điều này giống như tên tệp
type	BOTH	Loại tệp
fileName	BOTH	Tên tệp

Chụp ảnh

- Các bạn đã được giới thiệu sơ lược tất cả những gì mà thư viện **react-native-image-picker** cung cấp. Bây giờ chúng ta sẽ bắt đầu xây dựng một ứng dụng chụp ảnh:
  - Đầu tiên, các bạn cần khai báo các option của camera chúng ta cần.

```
// Đây là option sẽ sử dụng chung cả chụp ảnh và chọn ảnh
const commonOptions: OptionsCommon = {
        mediaType: 'photo',
        maxWidth: 500,
        maxHeight: 500,
    };
```

Chụp ảnh

- **cameraOptions**: Cung cấp thêm option **cameraType: “front”** để chụp ảnh bằng camera trước, **saveToPhotos: true** để lưu hình ảnh trong thư viện

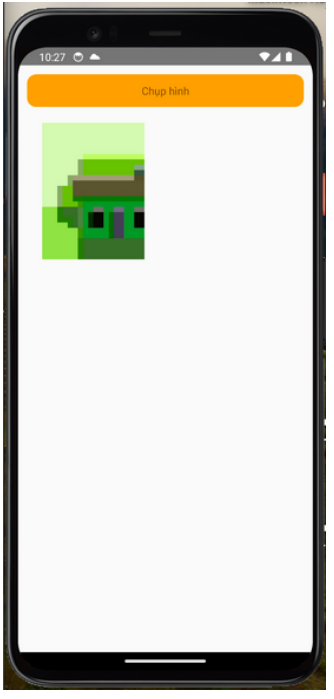
```
const cameraOptions: CameraOptions = {
    cameraType: 'front',
    saveToPhotos: true,
    ...commonOptions,
};
```

Chụp ảnh

- Hàm sử lý khi người dùng nhấn mở camera. Kèm theo xử lý thông báo lỗi khi người dùng thao tác sai với camera.

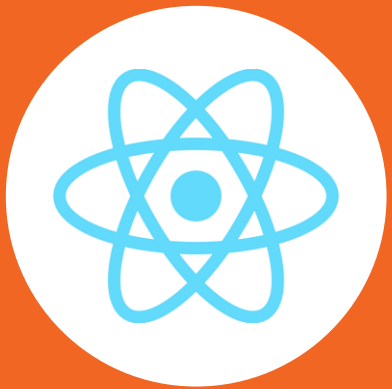
```
const onOpenCamera = async () => {
  const response = await launchCamera(cameraOptions);
  if (response?.assets) {
    setImages(response.assets);
  } else {
    Alert.alert('Có lỗi xảy ra', response.errorMessage);
  }
};
```

- Chạy chương trình và chúng ta có kết quả sau:



FPT Education

FPT POLYTECHNIC



LẬP TRÌNH ĐA NỀN TẢNG VỚI REACT NATIVE

BÀI 4: GIỚI THIỆU VỀ ỨNG DỤNG CHỤP ẢNH, LẤY ẢNH VÀ PHÁT NHẠC

PHẦN 2: TẠO ỨNG DỤNG PHÁT NHẠC

https://caodang.fpt.edu.vn/

Mục tiêu

- Tìm hiểu về thư viện **react-native-track-player**
- Tạo một ứng dụng phát nhạc

Giới thiệu

- Các ứng dụng phát nhạc, là một phần không thể thiếu trong thế giới lập trình ứng dụng di động. Để phát triển các ứng dụng phát nhạc bạn cần tạo các native module, để có thể phát được âm thanh. Việc viết bằng native module này khá tốn nguồn lực. May thay, đã có một số thư viện đã hỗ trợ rất tốt cho việc phát âm thanh.
- Ở bài học này, sẽ hướng dẫn sử dụng thư viện **react-native-track-player,** ở thời điểm viết slide này, đây là thư viện hỗ trợ mãnh mẽ nhất cho việc phát nhạc.

Cài đặt

- Chạy lệnh sau để cài đặt thư viện:

```
npm install --save react-native-track-player
```

Setup

- Cài đặt hoàn tất, bây giờ các bạn cần một số setup ban đầu để thư viện có thể hoạt động được
- Dưới đây là 2 bước setup cơ bản bạn cần thực hiện:
- **Bước 1**, bạn cần đăng ký **playback service** ngay sau khi đăng ký thành phần chính của ứng dụng (thường là trong tệp **index.js** ở gốc dự án):

```
// AppRegistry.registerComponent(...);
TrackPlayer.registerPlaybackService(() => require('./service'));
```

```
// service.js
module.exports = async function() {
    // Service này cần được đăng ký để module hoạt động
    // nhưng nó sẽ được sử dụng sau trong phần 'Receiving Events'
}
```

- **Bước 2**, bạn cần thiết lập trình phát. Quá trình này thường mất ít hơn một giây:

```
import TrackPlayer from 'react-native-track-player';

await TrackPlayer.setupPlayer()
// Trình phát sẵn sàng để sử dụng
```

- Đảm bảo rằng phương pháp thiết lập đã hoàn tất trước khi tương tác với bất kỳ chức năng nào khác trong **TrackPlayer** để tránh sự mất ổn định.

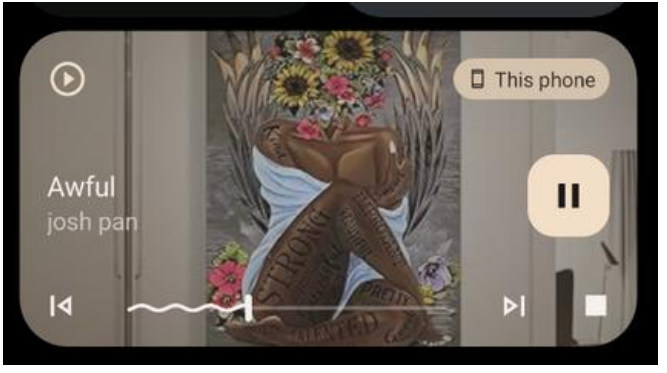
- Xây dựng **hook** riêng để xử lý các hàm thao tác với **TrackPlayer**, nơi đây chứa các setup, function điều khiển, trạng thái âm thanh.

- Cài đặt:

  - **Bước 1:** Thêm các listener **Media Controls** vào **registerPlaybackService**

    Media Controls là các trình điều khiển bên ngoài ứng dụng, giúp bạn có thể bật, tắt, thay đổi thời lượng....

Setup



Viết hàm **playbackService** để khai báo các listen của sự kiện **Media Controls,** hàm này sẽ được import trong file gốc **index.js**

```
export async function playbackService() {
  TrackPlayer.addEventListener(Event.RemotePause, () => {
    TrackPlayer.pause();
  });

  TrackPlayer.addEventListener(Event.RemotePlay, () => {
    TrackPlayer.play();
  });
}
```

```
TrackPlayer.addEventListener(Event.RemoteNext, () => {
  TrackPlayer.skipToNext();
});

TrackPlayer.addEventListener(Event.RemotePrevious, () => {
  TrackPlayer.skipToPrevious();
});

TrackPlayer.addEventListener(Event.RemoteSeek, ({position}) => {
  TrackPlayer.seekTo(position);
});
```

Sau đó bạn sẽ import function này vào **registerPlaybackService** để đăng ký các event cho **Media Controls**

```
AppRegistry.registerComponent(appName, () => App);

TrackPlayer.registerPlaybackService(() => playbackService);
```

- **Bước 2:** Viết hook **usePlayTrack** để gọi các hàm điều khiển âm thanh cho ứng dụng

  Đầu tiên, viết hàm **startPlayer** để setup ban đầu cho trình phát nhạc, và khai báo các trình điều khiển cho **Media Controls.**

Setup

```
const startPlayer = async setSetupDone => {
  try {
    await TrackPlayer.setupPlayer().finally(() => setSetupDone(true));
    await TrackPlayer.updateOptions({
      capabilities: [
        Capability.Play, Capability.Pause, Capability.Stop,
        Capability.SeekTo, Capability.SkipToNext, Capability.SkipToPrevious,
      ],
    });
    await TrackPlayer.setRepeatMode(RepeatMode.Off);
  } catch (error) {
    console.log('[Error player] ', error);
  }};
```

Luôn phải gọi hàm **setupPlayer()** đầu tiên, trước khi bắt đầu sử dụng các hàm nào của **react-native-track-player.**
updateOptions là các nút tính năng của **Media Controls.** Như bật, tắt, chuyển bài....
setRepeatMode cho phép lặp lại của âm thanh.

- Để sử dụng custom hook **usePlayTrack**, cần phải truyền vào danh sách âm thanh cần phát (playListData). Hook **usePlaybackState** dùng để lấy trạng thái phát nhạc hiện tại, đang phát, hay đang dừng..., **isSetupDone** là true nếu như đã setup thành công

```
export const usePlayTrack = playListData => {
  const playBackState = usePlaybackState();
  const [isSetupDone, setSetupDone] = useState(false);
```

- Ở trong hook **usePlayTrack**, bạn cần lưu trữ những state của đoạn âm thanh như: vị trí, thời lượng, tên, hình ảnh... của đoạn âm thanh. Hook **useProgress** dùng để lấy thời lượng track đang phát.

```
const {duration, position} = useProgress();
const [trackTitle, setTrackTitle] = useState();
const [trackArtist, setTrackArtist] = useState();
const [trackArtwork, setTrackArtwork] = useState();
```

Listen event

- Sử dụng hook **useTrackPlayerEvents** để bạn có thể lấy tên, hình ảnh,... của đoạn âm thanh phát hiện tại

```
useTrackPlayerEvents([Event.PlaybackActiveTrackChanged], async
event => {
  const {title, artwork, artist} = event?.track || {};
  if (event.type === Event.PlaybackActiveTrackChanged &&
!!event?.track) {
    setTrackTitle(title);
    setTrackArtist(artist);
    setTrackArtwork(artwork);
  }
});
```

Khởi tạo trình phát âm thanh

- Sử dụng **useEffect** để gọi hàm **startPlayer** để setup cho trình phát nhạc. Nếu screen **unmout** gọi hàm **reset()** để xoá tất cả đoạn âm thanh trong trình phát nhạc.

```
useEffect(() => {
  startPlayer(setSetupDone);
  return () => {
    TrackPlayer.reset();
  };
}, []);
```

Thêm danh sách âm thanh

- Sau khi setup thành công, thêm danh sách nhạc của mình vào hàng chờ của **TrackPlayer.**

```
useEffect(() => {
  if (!!isSetupDone && !!playListData) {
    TrackPlayer.getActiveTrack().then(async activeTrack => {
      if (!activeTrack) {
        await TrackPlayer.add(playListData);
      }
    });
  }
}, [isSetupDone, playListData]);
```

Hàm phát/dừng âm thanh

- Sử dụng **playBackState** để xác định trạng thái phát nhạc hiện tại để dừng hoặc phát nhạc.

```
const onTogglePlayTrack = async () => {
  if (playBackState.state === State.Playing) {
    await TrackPlayer.pause();
  } else {
    await TrackPlayer.play();
  }
};
```

Tua âm thanh đang phát

- Gọi hàm **seekTo** để tua đến đoạn âm thanh mong muốn**. toTime** là thời lượng tính bằng giây, mà bạn muốn tua đến

```
const onSeekTo = toTime => {
  TrackPlayer.seekTo(toTime);
};
```

Tiếp theo/Trở lại bài

- Gọi hàm **skipToNext** để chuyển đến bài tiếp theo, gọi hàm **skipToPrevious** để trở lại bài trước.
- **initialPosition** là vị trí của bài trong hàng chờ mà bạn muốn chuyển đến.

```
const onSkipToNext = initialPosition => {
  TrackPlayer.skipToNext(initialPosition);
};

const onSkipToPrevious = initialPosition => {
  TrackPlayer.skipToPrevious(initialPosition);
};
```

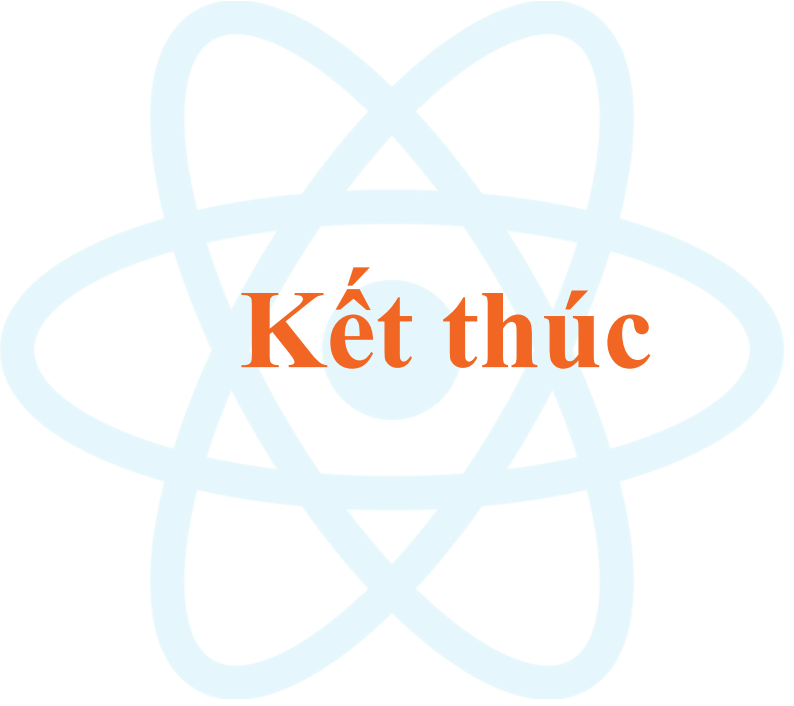
return value

- Return các giá trị trong hook, bạn có thể lấy giá trị bất kì trong hook được return để sử dụng vào trình phát âm thanh của mình

```
return {
  onTogglePlayTrack,
  onSeekTo,
  onSkipToNext,
  onSkipToPrevious,
  playBackState: playBackState.state,
  duration,
  position,
  trackTitle,
  trackArtist,
  trackArtwork,
};
```

Tổng kết

- [ ] Cài đặt thư viện chụp, lấy hình ảnh
- [ ] Tạo ứng dụng chụp ảnh
- [ ] Tạo ứng dụng lấy ảnh từ thư viện
- [ ] Tìm hiểu về thư viện **react-native-track-player**
- [ ] Tạo một ứng dụng phát nhạc



Kết thúc