**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HCM**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**(UIT)**



**Báo cáo**

**WEB NGHE NHẠC TRỰC TUYẾN**

**Học phần : Công nghệ web và ứng dụng**

**Lớp : SE341.J11.PMCL**

**Giảng viên : Trần Anh Dũng**

**Nhóm : Lê Xuân Tiến – 15520884**

**Nguyễn Đức Thông – 15520856**

Table of Contents

[1 GIỚI THIỆU 4](#_Toc533242326)

[1.1 Về đề tài: 4](#_Toc533242327)

[1.2 Tính năng 4](#_Toc533242328)

[2 CÔNG NGHỆ SỬ DỤNG 5](#_Toc533242329)

[2.1 React JS 5](#_Toc533242330)

[2.2 Axios 5](#_Toc533242331)

[2.3 BodyParser 6](#_Toc533242332)

[2.4 R 6](#_Toc533242333)

[2.5 R 6](#_Toc533242334)

[2.6 R 6](#_Toc533242335)

[2.7 R 6](#_Toc533242336)

[2.8 R 6](#_Toc533242337)

[2.9 Redux 6](#_Toc533242338)

[3 THIẾT KẾ KIẾN TRÚC 7](#_Toc533242339)

[3.1 Kiến trúc tổng thể 7](#_Toc533242340)

[3.2 Diễn giải kiến trúc tổng thể 7](#_Toc533242341)

[4 THIẾT KẾ MÀN HÌNH 8](#_Toc533242342)

[4.1 Màn hình trang chủ 8](#_Toc533242343)

[5 CÁC KỸ THUẬT CHÍNH 9](#_Toc533242344)

[5.1 Lấy data từ Nhaccuatui.com 9](#_Toc533242345)

[5.2 Lưu trữ dữ liệu bài hát vào Local Storage 10](#_Toc533242346)

[5.3 Lưu trữ dữ liệu bài hát vào Cloud Firestore 11](#_Toc533242347)

[6 CÀI ĐẶT 14](#_Toc533242348)

[7 KẾT LUẬN 15](#_Toc533242349)

[7.1 Môi trường phát triển và Môi trường triển khai 15](#_Toc533242350)

[7.2 Kết quả đạt được 15](#_Toc533242351)

[7.3 Hướng phát triển 16](#_Toc533242352)

[8 TÀI LIỆU 17](#_Toc533242353)

# GIỚI THIỆU

## Về đề tài:

Trong thời đại công nghệ như ngày nay, internet là thứ không thể thiếu trong cuộc sống hằng ngày của chúng ta, góp phần đẩy mạnh công cuộc công nghiệp hóa hiện đại hóa đất nước.

Theo tờ báo internetworldstats, Việt Nam đứng thứ 13 trong top 20 quốc gia sử dụng internet nhiều nhất thế giới, với 64 triệu người sử dụng Internet trong số 96.5 triệu dân, xếp sau các nước:

1. Trung quốc
2. Ấn Độ
3. Mỹ
4. Brazil
5. Indonesia
6. Nhật
7. Nga
8. Nigeria
9. Mexico
10. Bangladesh
11. Đức
12. Philippine.

Trong đó 75% người dung internet vào mạng để nghe nhạc trực tuyến. Chính vì lý do đó mà nhóm chúng em đã làm một website nghe nhạc trực tuyến, mục đích chính là để học tập them kiến thức về các công nghệ và các cách thức xây dựng một website. Đây là một web site nghe nhạc trực tuyến giúp mọi người sử dụng tiếp cận âm nhạc dễ dàng hơn.

Dữ liệu của trang web chúng em lấy từ mp3.zing.vn.

## Tính năng

Sau đây là các chức năng chính của website nghe nhạc:

* Nghe nhạc.
* Top 100 bài hát Âu Mỹ.
* Top 100 bài hát Hàn Quốc.
* Top 100 bài hát Việt Nam.
* Top album
* Top ca sĩ.
* Tìm kiếm bài hát
* Thêm bài hát vào playlist.
* Đăng nhập.
* Tải bài hát.
* Tạo playlist cá nhân

# CÔNG NGHỆ SỬ DỤNG

## React JS

**React.js** là 1 thư viện JavaScript tạo ra bởi Facebook.

*http://facebook.github.io/react/*

Như khái niệm trên trang web chính thức “A JavaScript library for building user interface”, React.js là một thư viện sinh ra để xây dựng giao diện người dùng (UI). Nó không phải là Framework mà chỉ là thư viện, do đó trong MVC nó sẽ tương ứng với phần V.

Ngoài Facebook và Instagram – nơi làm ra và maintain React.js, Yahoo hay Airbnb là những ví dụ nổi bật có sử dụng thư viện này. Hiện nay, đây là một thư viện thu hút được rất nhiều sự quan tâm.

## Axios

Axios là một HTTP client được viết dựa trên Promises được dùng để hỗ trợ cho việc xây dựng các ứng dụng API từ đơn giản đến phức tạp và có thể được sử dụng cả ở trình duyệt hay Node.js.

Việc tạo ra một HTTP request dùng để fetch hay lưu dữ liệu là một trong những nhiệm vụ thường thấy mà một ứng dụng Javascript phía client cần phải làm khi muốn giao tiếp với phía server. Các thư viện bên thứ 3, đặc biệt là jQuery từ xưa đến nay vẫn là một trong những cách phổ biến để giúp cho các browser API tương tác tốt hơn, rõ ràng mạch lạc hơn và xóa đi những điểm khác biệt giữa các browser với nhau.

Khi ngày càng có nhiều developer thích dùng các native DOM API được nâng cấp và cải thiện hơn jQuery, hay các thư viện lập trình front-end UI như React và Vue.js thì việc sử dụng cả một thư viện jQuery chỉ với một giá trị sử dụng đó là tính năng của hàm $.ajax đã trở nên bất hợp lý hơn bao giờ hết.

## BodyParser

Body-parser được dùng để parse dữ liệu lấy về thành file JSON để tiện cho việc thao tác xử lý.

## LyricParser

Lyric-parser được dùng để parse file .lrc sang dạng JSON để tiện cho việc hiển thị lời bài hát. Ví dụ chúng ta có file lời nhạc như sau:

[ar: Camila Cabello, Young Thug]

[ti: Havana]

[al: Havana (Single)]

[length: 03:36]

[00:00.00]Song: Havana

[00:02.00]Singer: Camila Cabello, Young Thug

[00:04.00]

[00:08.86]Havana ooh na-na (ayy)

[00:11.56]Half of my heart is in Havana

LyricParser sẽ parse file sau thành:

{

ar: 'Camila Cabello, Young Thug',

ti: 'Havana',

al: 'Havana (Single)',

length: '03:36',

scripts: [ { start: 0, text: 'Song: Havana', end: 2 },

{ start: 2, text: 'Singer: Camila Cabello, Young Thug', end: 4 },

{ start: 8.86, text: 'Havana ooh na-na (ayy)', end: 11.56 },

{ start: 11.56,

text: 'Half of my heart is in Havana ',

end: 14.11 }

}

## Mongoose

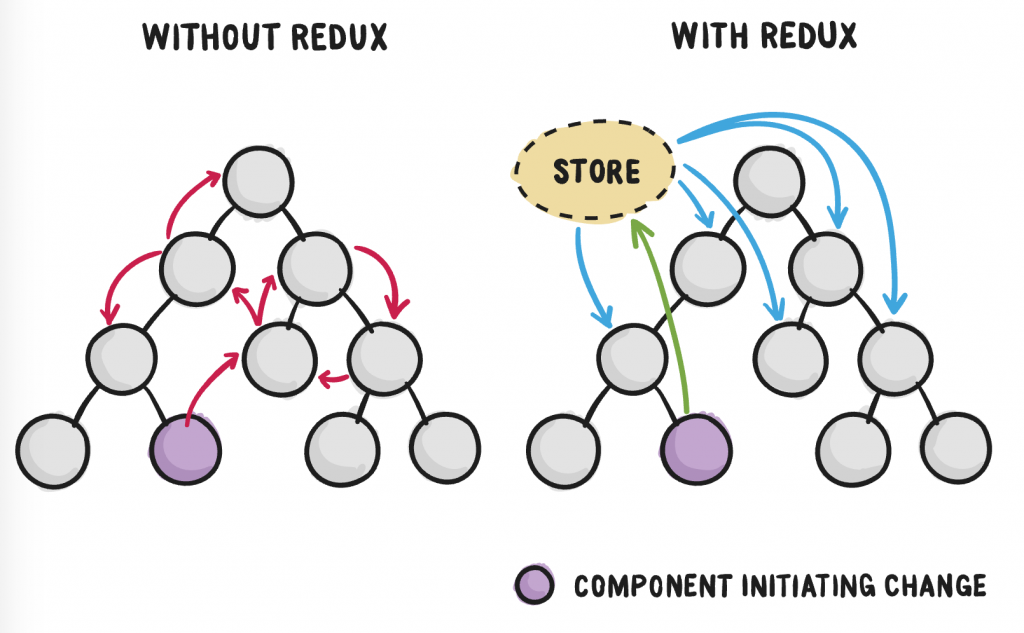
Mongoose là một thư viện mô hình hóa đối tượng (Object Data Model - ODM) cho MongoDB và Node.js. Nó quản lý mối quan hệ giữa dữ liệu, cung cấp sự xác nhận giản đồ và được sử dụng để dịch giữa các đối tượng trong mã và biểu diễn các đối tượng trong MongoDB.

MongoDB là một cơ sở dữ liệu NoSQL. Bạn có thể lưu trữ các JSON trong đó, và cấu trúc của các tài liệu này có thể thay đổi vì nó không bắt buộc như các cơ sở dữ liệu SQL. Đây là một trong những lợi thế của việc sử dụng NoSQL vì nó tăng tốc độ phát triển ứng dụng và giảm sự phức tạp của việc triển khai.

## Redux

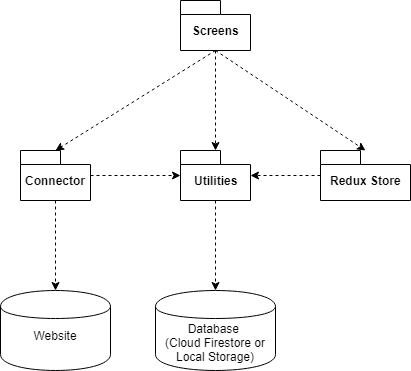
Khi làm việc với React Native hay các dự án ứng dụng Single Page nói chung, có một vấn đề khá đau đầu là làm sao quản lý được trạng thái của ứng dụng đó.

Redux là một thư viện Javascript giúp chúng ta giải quyết vấn đề đau đầu này bằng cách lưu trữ state của các component ở một store bên ngoài chung. Như vậy, việc quản lý các state này sẽ dễ dàng thuận tiện và có thể truy cập ở bất kì component nào mong muốn. Redux có thể kết hợp với React Native một cách hoàn hảo, bởi vì nó dựa trên nền tảng tư tưởng của Flux – do Facebook giới thiệu một vài năm trước.



# THIẾT KẾ KIẾN TRÚC

## Kiến trúc tổng thể

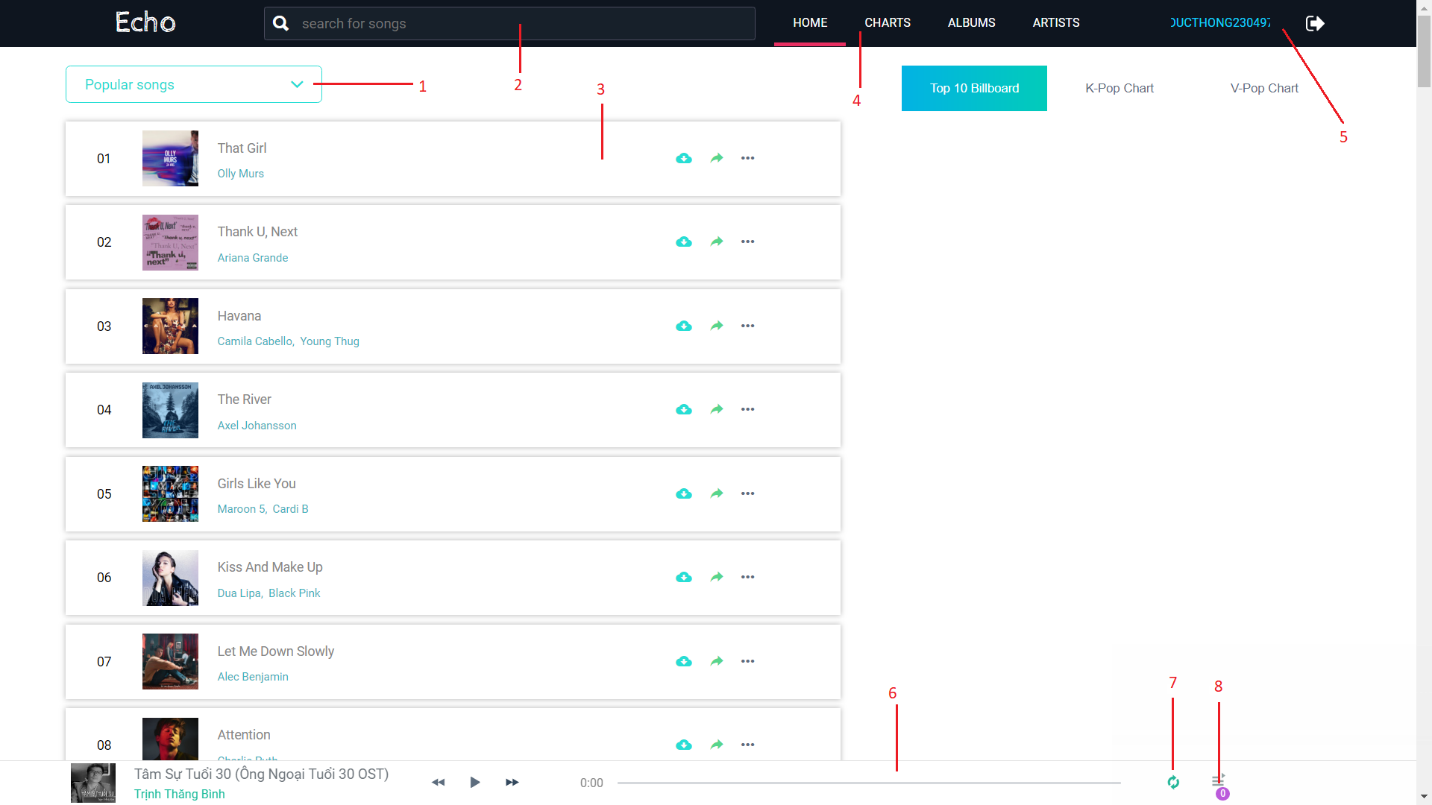


## Diễn giải kiến trúc tổng thể

|  |  |  |
| --- | --- | --- |
| **STT** | **Thành phần** | **Diễn giải** |
|  | Screens | Chứa các màn hình (UI) của ứng dụng |
|  | Connector | Chức các phương thức giúp cho việc lấy dữ liệu từ website nhaccuatui.net |
|  | Utilities | Chức các phương thức giúp cho việc kết nối với cơ sở dữ liệu (Cloud Firestore hoặc Local) cũng như việc tương tác với cơ sở dữ liệu từ ứng dụng |
|  | Redux Store | Quản lý trạng thái của ứng dụng |

# THIẾT KẾ MÀN HÌNH

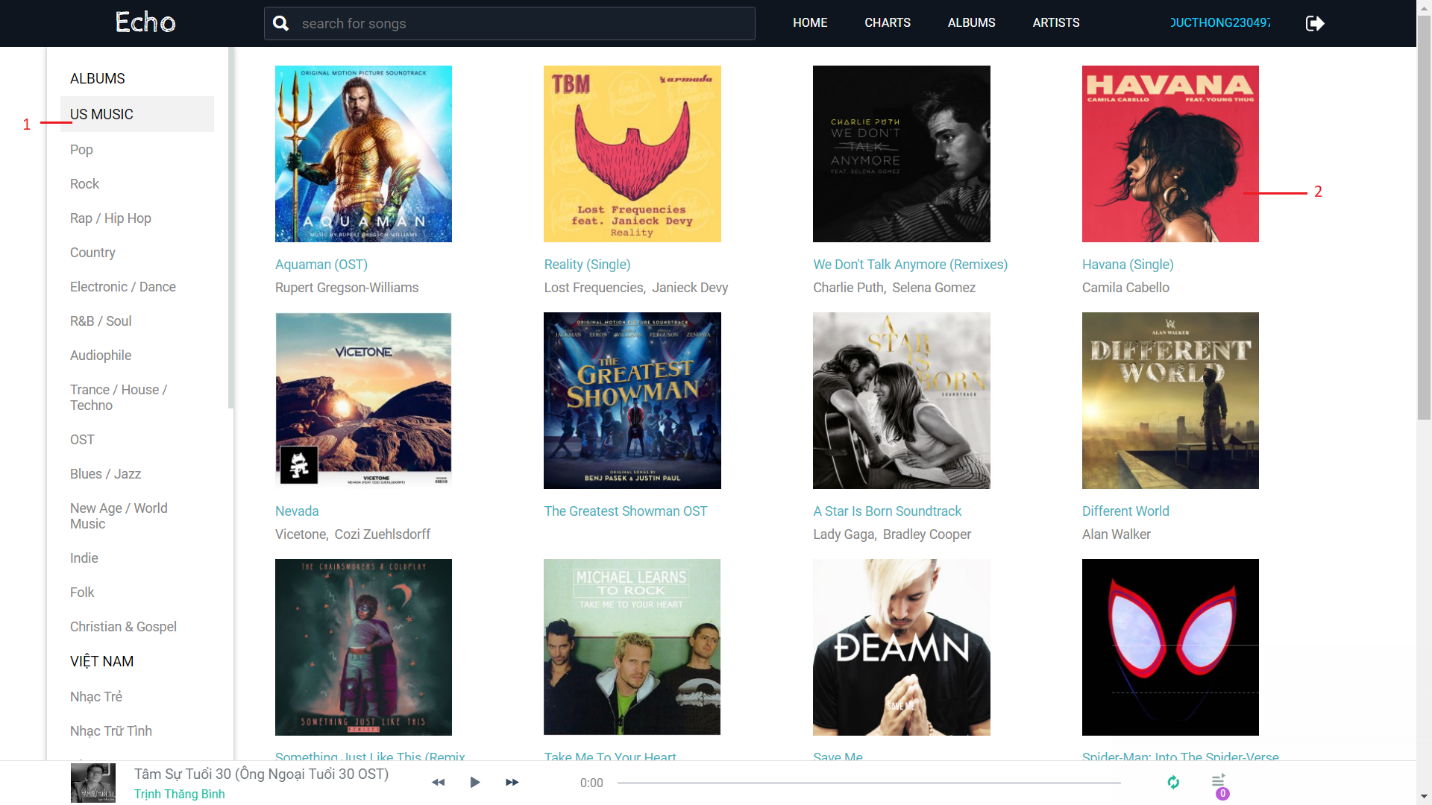
## Màn hình trang chủ



|  |  |
| --- | --- |
| 1 | Top các bài hát phổ biến: TopUS, TopKpop, TopVpop |
| 2 | Thanh tìm kiếm các bài hát |
| 3 | Danh sách 100 bài hát |
| 4 | Tab bar |
| 5 | Đăng nhập, đăng xuất |
| 6 | Trackbar |
| 7 | Chế độ lặp lại bài hát |
| 8 | Danh sách các bài hát trong playlist online đang nghe |

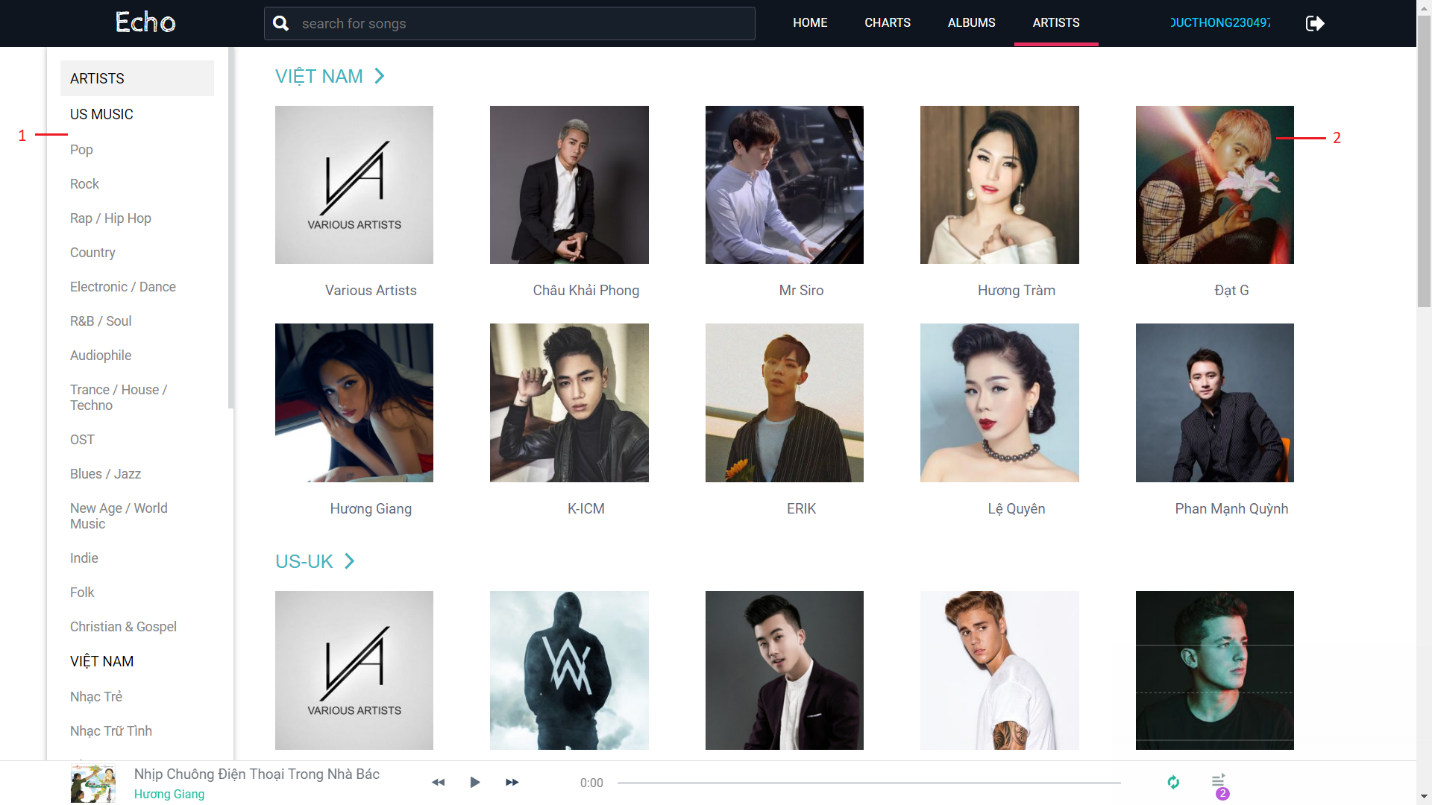
Đây là màn hình trang chủ của website, ở đây được hiển thị top 100 các bài hát, thanh tìm kiếm bài hát cho phép tìm những bài hát ta muốn, phần tab bar sẽ navigate qua các trang charts, albums, artists, ngoài ra còn có phần đăng nhập giúp chúng ta lưu lại các bài hát mình thích vào danh sách riêng. Bên dưới trang là track bar, bài hát chúng ta đang nghe sẽ được hiển thị dưới đây, chúng ta có thể chỉnh ra chế độ loop để nghe lặp lại bài hát và xem danh sách các bài hát online chúng ta đã them vào

## Màn hình albums



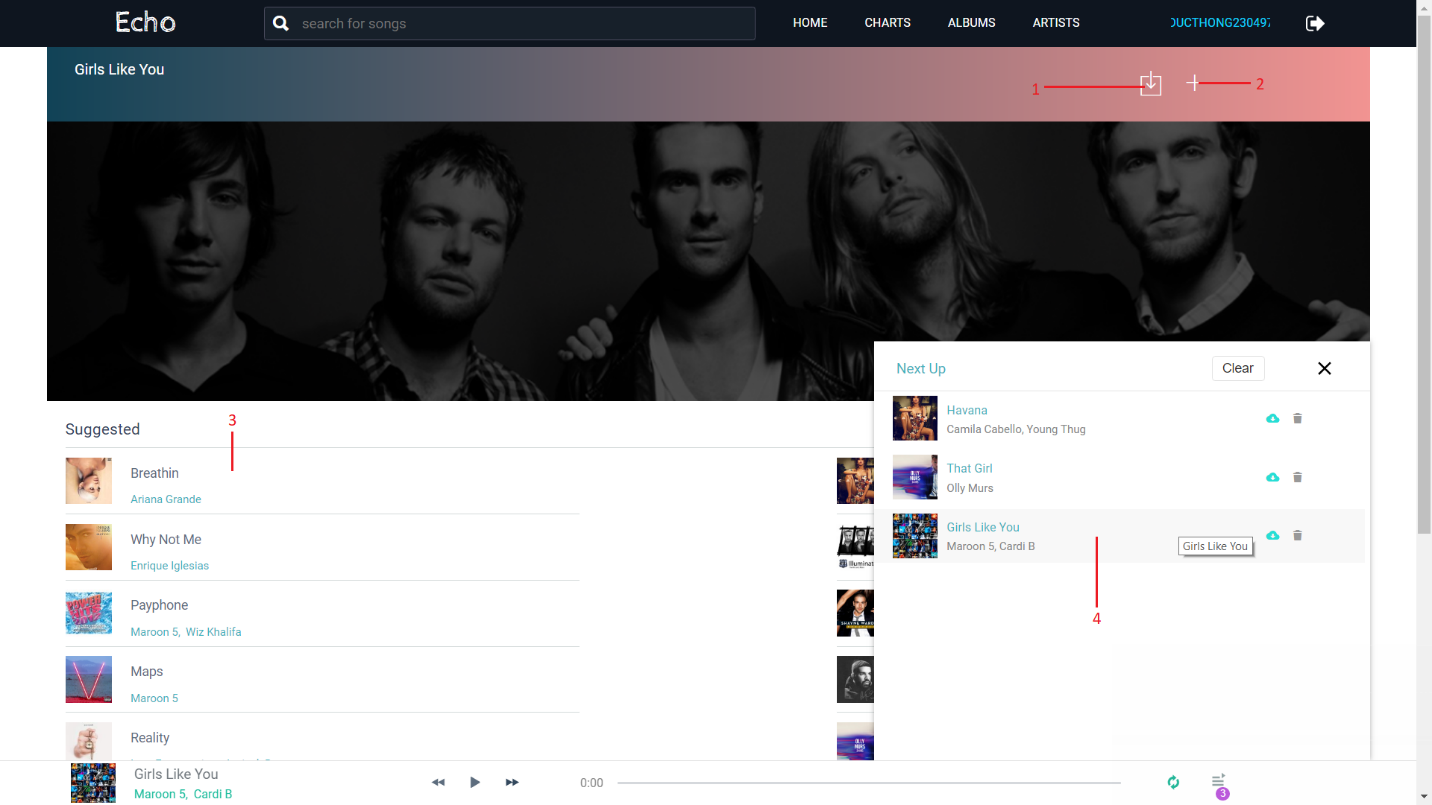
|  |  |
| --- | --- |
| 1 | Danh mục |
| 2 | Danh sách các albums |

## Màn hình nghệ sĩ



|  |  |
| --- | --- |
| 1 | Danh mục |
| 2 | Danh sách các nghệ sĩ |

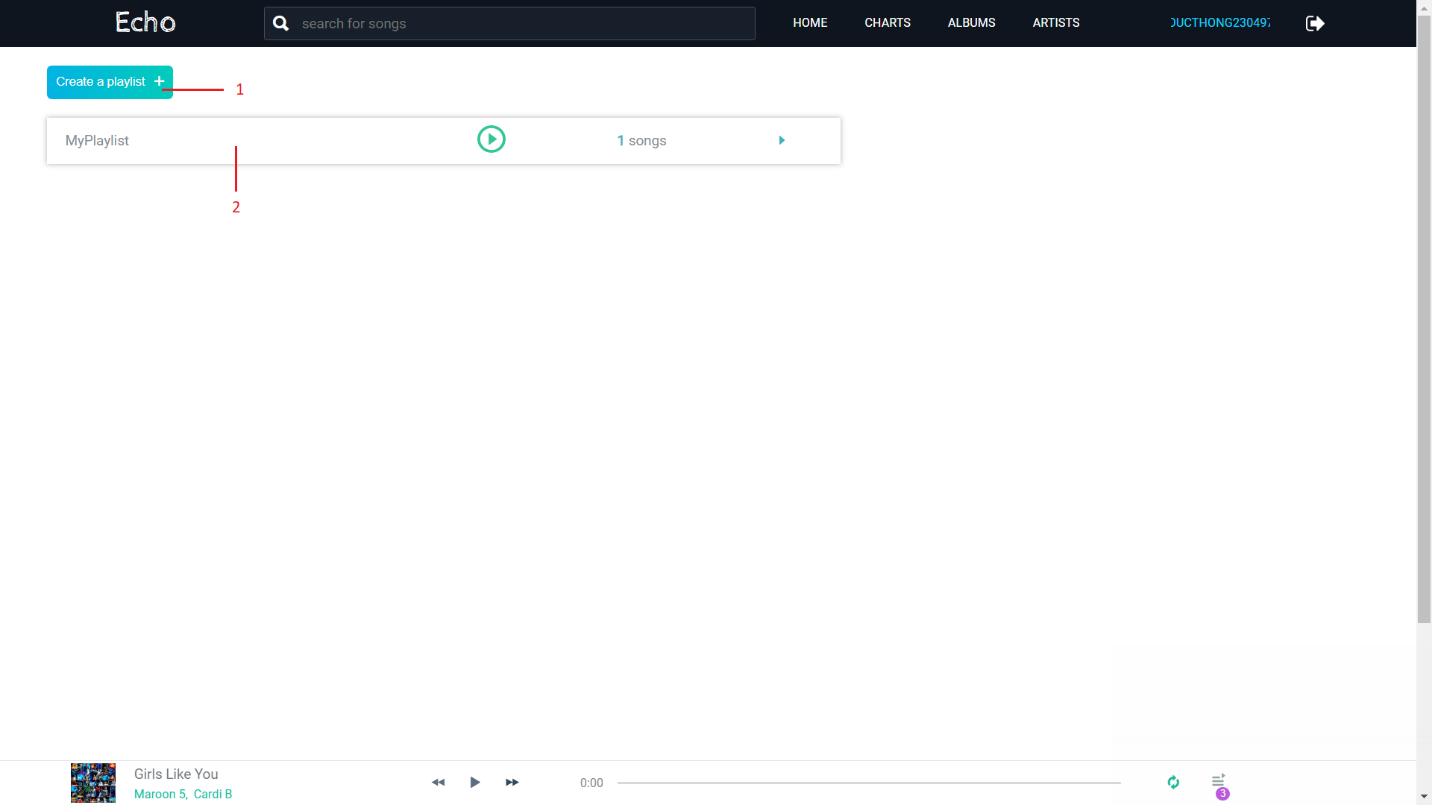
## Màn hình nghe nhạc



|  |  |
| --- | --- |
| 1 | Tải nhạc |
| 2 | Thêm nhạc vào playlist riêng |
| 3 | Danh sách các bài hát đề nghị |
| 4 | Danh sách các bài hát đang nghe |

Đây là màn hình hiển thị bài hát chúng ta đang nghe, chúng ta có thể download hoặc them vào playlist riêng, bên dưới đó là các bài hát được gọi ý cho chúng ta, chúng ta cũng có thể xem danh sách các bài hát online chúng ta đã thêm vào.

## Màn hình trang cá nhân

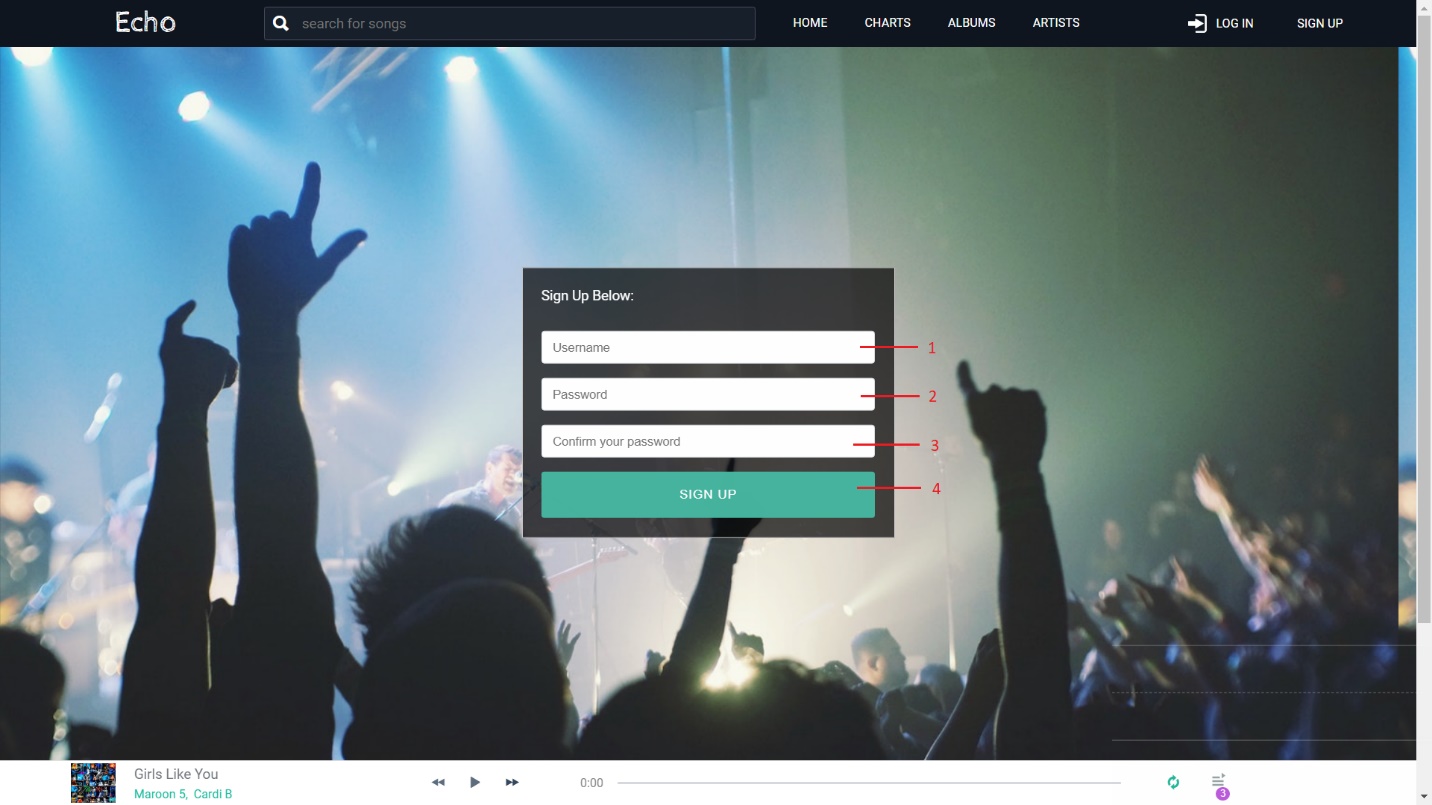


|  |  |
| --- | --- |
| 1 | Tạo playlist mới |
| 2 | Danh sách các playlist hiện có |

Đây là màn hình tạo playlist riêng khi nhấn là tên của tài khoản. Trang hiển thị tất cả các playlist chúng ta đã tạo.

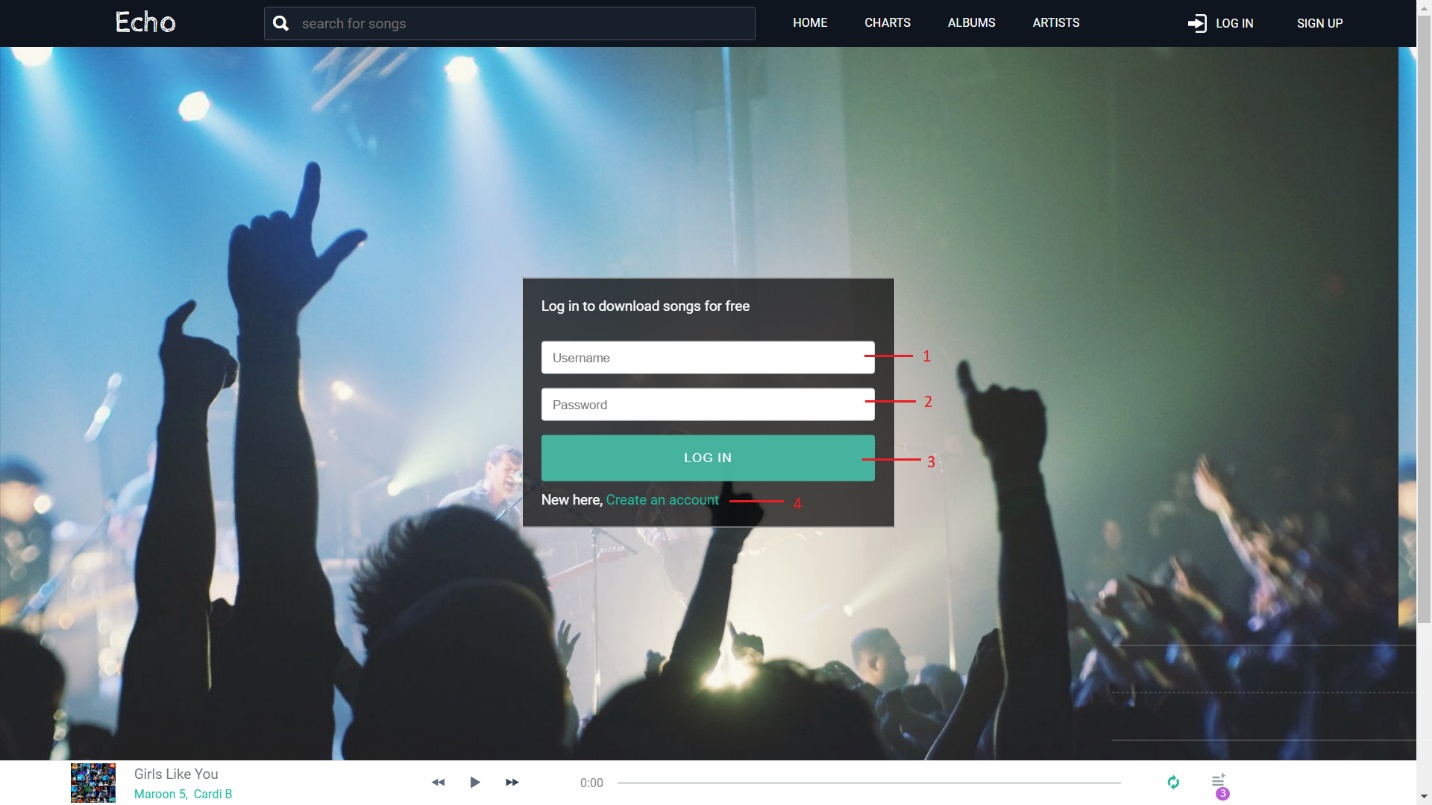
Để tạo một playlist mới chúng ta nhấn vào nút create playlist, sau đó điền tên cho playlist mà chúng ta muốn tạo và nhấn OK.

## Màn hình đăng kí



|  |  |
| --- | --- |
| 1 | Tên đăng nhập |
| 2 | Mật khẩu |
| 3 | Nhập lại mật khẩu |
| 4 | Nút đăng kí |

## Màn hình đăng nhập



|  |  |
| --- | --- |
| 1 | Tài khoản |
| 2 | Mật khẩu |
| 3 | Đăng nhập |
| 4 | Đăng kí tài khoản |

# CÁC KỸ THUẬT CHÍNH

*Mục này đề cập đến những vấn đề kỹ thuật chính mà bạn có thể sẽ quan tâm và code để thực hiện những kỹ thuật đó. Tuy nhiên, chúng ta sẽ không đề cập đến quá chi tiết những bước cài đặt trong xuyên suốt báo cáo này vì chúng có thể dễ dàng được tìm thấy trên các website chính thức đã đề cập ở mục 2.*

## Lấy data từ Nhaccuatui.com

Ví dụ, để lấy thông tin về top 100 nhạc pop US.UK hiện hành, ta có thể đi đến website:

[*https://www.nhaccuatui.com/playlist/top-100-pop-usuk-hay-nhat-va.zE23R7bc8e9X.html?st=1*](https://www.nhaccuatui.com/playlist/top-100-pop-usuk-hay-nhat-va.zE23R7bc8e9X.html?st=1)

Nhấn F12 để có thể quan sát các thành phần của trang web hiện tại. Tiếp tục đi đến tab elements, ta có thể thấy mã html cho trang web này và đây là tất cả những gì mà chúng ta cần.

Vấn đề bây giờ chỉ còn là làm cách nào để lấy được dữ liệu cần thiết từ mã html để chuẩn bị cho việc xử lý và hiển thị lên màn hình. Rất may mắn, **Fetch API** của React Native cung cấp đủ các hàm cần thiết cho việc lấy dữ liệu này.

export async function getTop100(url){

let listSong = [];

//fetch data from the specified link

await fetch(url).then((response) => {

//create regex patterns and get information about songs in response

let re = /<ul class="list\_song\_in\_album">([\s\S]\*?)<\/ul>/ig;

let result = response.\_bodyInit.match(re)

let regexListSong = /<li([\s\S]\*?)<\/li>/ig

let listItemSong = result[0].toString().match(regexListSong)

let regexSongItem = /<meta content="(.\*?)"/ig

//loop through each item and get song information from it

listItemSong.forEach(element => {

let singerName =/titleplay="(.\*?)"/ig.exec(element.toString())[1]

.toString().split("-")[1].trim()

let match = element.toString().match(regexSongItem)

let songName = match[0].toString()

.replace("<meta content=\"", "").replace("\"", "")

let songURL = match[1].toString()

.replace("<meta content=\"", "").replace("\"", "")

let songInfo = {songName: songName, artist: singerName,songURL: songURL}

listSong.push(songInfo)

});

}).catch((error) => {

console.error(error);

});

return listSong;

}

Nhìn vào code trên, ta có thể thấy hàm **async** được sử dụng và lý do cho việc này là khá rõ ràng. Khi chúng ta gọi hàm **fetch** của Fetch API, hàm trả về một **Promise** để chỉ việc nó sẽ cố gắng lấy data từ website nào đó mà ta muốn và trả về một phản hồi sớm nhất có thể mà không làm luồng ứng dụng chính bị block. Nói ngắn gọn, Fetch là một phương thức bất đồng bộ. Tính năng này của hàm Fetch là hợp lý và tiện dụng bởi lẽ ta không muốn ứng dụng dừng hoạt động hoàn toàn để chờ cho đến khi nó trả về được data mong muốn vì điều này có thể gây trải nghiệm không tốt đối với người dùng. Nhưng tính năng này cũng không đảm bảo rằng **listSong** sẽ được gán giá trị đúng đắn trước khi thoát ra khỏi hàm **getTop100.** Một cách giải quyết được sử dụng ở đây chính là đợi cho đến khi hàm fetch hoàn thành việc trả về dữ liệu bằng cách đặt từ khóa **await** phía trước hàm này và đặt từ khóa **async** phía trước hàm getTop100. Bằng cách này, chúng ta có thể cùng lúc đảm bảo lưu trữ được dữ liệu vào listSong như mong muốn và đồng thời không block hoạt động của toàn ứng dụng.

Một điều cần lưu ý ở đây là đối với việc lấy dữ liệu từ một trang web như trên, các data trả về sẽ ở dạng code html. Vì vậy, cách dễ dàng nhất đề lấy các data trong các tag html mong muốn là dùng regex. Tuy nhiên, regex là một chủ đề hoàn toàn khác với những gì ta đang thảo luận ở đây, ta sẽ không bàn về nó thêm ở đây.

## Download bài hát

Như đã đề cập ở phần 2, **Rn-fetch-blob** và **AsyncStorage** kết hợp hoàn hảo trong việc tải và lưu trữ dữ liệu bài hát vào Local Storage. AsyncStorage có nhiệm vụ lưu lại những dữ liệu như danh sách bài hát, số lượng bài hát trong danh sách, … và quan trọng nhất là lưu trữ đường dẫn đến bài hát đã được tải về bởi Rn-fetch-blob.

Rn-fetch-blob đã làm gần như tất cả các công việc nặng nhọc, vì vậy tải một bài hát từ một trang web hoặc đường link nào đó được thực hiện khá đơn giản như sau:

downloadData = async(url, appendExt) =>{

let path = null;

await RNFetchBlob.config({

// add this option that makes response data to be stored as a file,

// this is much more performant.

fileCache : true,

// by adding this option, the temp files will have a file extension

appendExt : appendExt

})

.fetch('GET', url, {})

.then((res) => {

path = res.path();

})

return path;

}

After downloading the song from the specified Url, Rn-fetch-blob returns a path to that song in our local device. The job for us now is simple, using AsyncStorage to save this path and do anything with it later.

Sau khi bài hát đã được tải xong, Rn-fetch-blob trả về một đường đến file vừa được tải về đó trong Local Storage. Công việc của chúng ta còn lại khá đơn giản, sử dụng AsyncStorage để lưu trữ đường dẫn này và làm bất điều gì mong muốn với nó sau này.

Đoạn code bên dưới chỉ ra cách lưu một thông tin đơn giản vào Local Storage:

storeData = async (name, value) => {

try {

await AsyncStorage.setItem(name, value);

} catch (error) {

console.log('Something went wrong!');

}

}

## Lưu trữ dữ liệu bài hát vào Cloud Firestore

Mặt dù có khá nhiều bài hướng dẫn trên mạng nói về việc tương tác với Cloud Firestore bằng **React Native Firebase**, tuy nhiên việc thảo luận nó ở đây là khá cần thiết, đặc biệt là cách mà dữ liệu được phân chia và lưu trữ như thế nào ở Cloud Firestore.

Có hai điều chúng ta cần quan tâm ở đây: Đăng nhập và sử dụng thông tin này để làm cầu nối trong việc lưu trữ và lấy data phù hợp cho người dùng.

Đăng ký và đăng nhập một tài khoản vào Firebase trở nên khá dễ dàng với React Native Firebase:

Firebase.auth().createUserWithEmailAndPassword(email, password)

.then((user) => {

this.login();

})

.catch((error) => {

console.log(error);

});

login() {

const { email, password } = this.state;

Firebase.auth().signInWithEmailAndPassword(email, password)

.then((user) => {

})

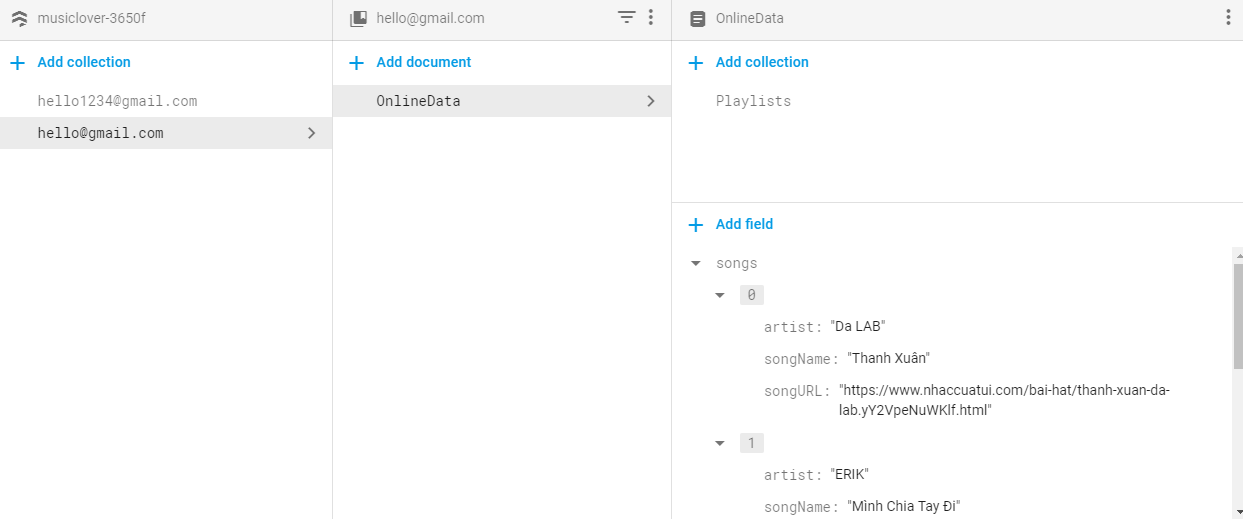
.catch((error) => {

console.log(error);

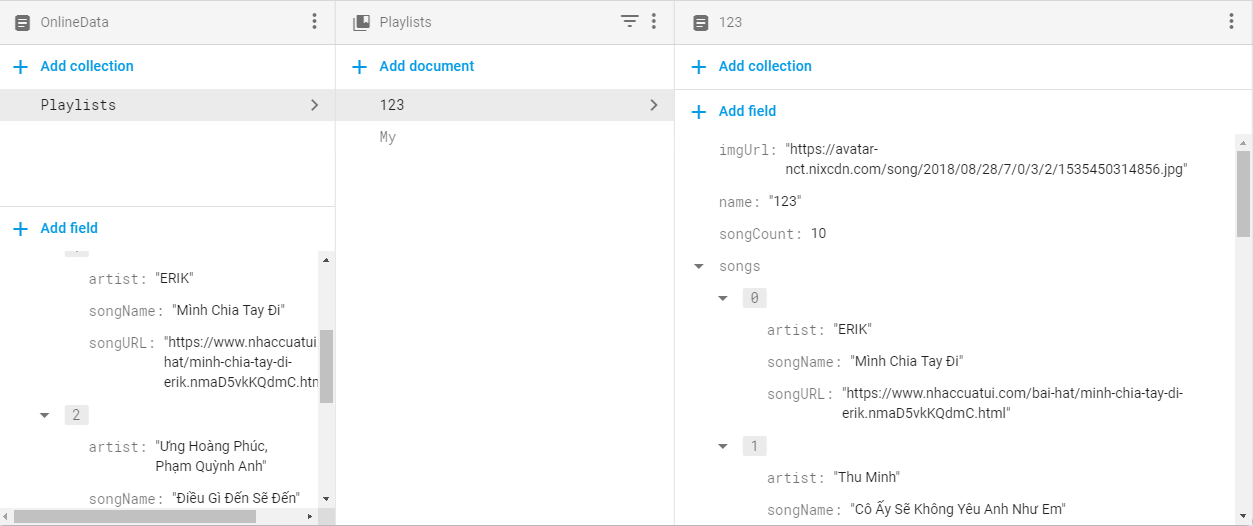
});

}

Không có nhiều điều để nói ở đây vì React Native Firebase đã làm gần như tất cả mọi công việc khó khăn cho chúng ta. Vấn đề còn lại bây giờ là làm thế nào để lưu trữ dữ liệu người dùng một cách hợp lý.



Nhìn vào hình ảnh này, ta có thể thấy rằng các data của người dùng được liên kết với email tương ứng với người dùng đó vì email là độc nhất. Cho mỗi người dùng, ta tạo ra một tài liệu **OnlineData** để lưu trữ tất cả danh sách bài hát mà người dùng đã thêm vào.



Mỗi người dùng có thể có bao nhiêu **playlist** tùy thích và mỗi playlist này chứa các thông tin cơ bản như số lượng bài hát, ảnh đại diện và một mảng các bài hát.

Code ví dụ dưới đây chỉ ra cách lưu trữ dữ liệu vào Cloud Firestore:

let userCollection =

Firebase.firestore().collection(this.props.user.email).doc('OnlineData');

userCollection.set({

songs: ...

})

# CÀI ĐẶT

Mở một terminal và đi đến thư mục chứa project và chạy:

1. **npm install** để cài đặt các module hay package yêu cầu bởi ứng dụng.
2. **npm start** để chạy.
3. Sau đó mở trình duyệt web và đi tới đường dẫn http://localhost:3000/

# KẾT LUẬN

## Kết quả đạt được

*Anh/Chị cần nêu* ***đầy đủ, chi tiết*** *kết quả đã đạt được trong đề tài: đã phân tích, thiết kế chi tiết những chức năng nào, những chức năng nào đã cài đặt hoàn chỉnh, những chức năng nào đã cài đặt nhưng chưa hoàn chỉnh, những chức năng nào chỉ có giao diện nhưng chưa xử lý*

*Cần nhấn mạnh rõ những điểm đặc sắc của đề tài (ví dụ: có khả năng thay đổi skin, có khả năng bổ sung tính năng “động” dưới dạng plug-in, cho phép thay đổi loại CSDL, ứng dụng được xây dựng theo kiến trúc MVC/.Net Tier …)*

Qua môn học này chúng em học được cách thức phát triển một ứng dụng web bằng ReactJs, cách cào dữ liệu từ một website sử dụng các công cụ hỗ trợ. Sau đây là một số tính năng mà chúng em đã làm được:

|  |  |
| --- | --- |
| 1 | Nghe nhạc |

## Hướng phát triển

*Anh/Chị hãy nêu ra hướng phát triển (nếu có) của đề tài*

# TÀI LIỆU

* Và nhiều nguồn tài liệu khác …