**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HCM**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**(UIT)**



**Báo cáo**

**ỨNG DỤNG NGHE NHẠC TRỰC TUYẾN**

**MUSICLOVER**

**Học phần : Đồ án mã nguồn mở**

**Lớp : SE111.J11.PMCL**

**Giảng viên : Vũ Thanh Nguyên**

**Nhóm : Trần Thị Thanh Thảo – 15520820**

**Trương Ngọc Sơn – 15520740**

**Nguyễn Đức Thông – 15520856**

**Mật Ngọt Trang Thanh -**

***TP.HCM, 16-12-2017***

Table of Contents

[1 GIỚI THIỆU 4](#_Toc532677409)

[1.1 Về đề tài: 4](#_Toc532677410)

[1.2 Tính năng 4](#_Toc532677411)

[1.3 Nhóm thực hiện 5](#_Toc532677412)

[2 CÔNG NGHỆ SỬ DỤNG 6](#_Toc532677413)

[2.1 React Native 6](#_Toc532677414)

[2.2 Fetch API 6](#_Toc532677415)

[2.3 AsyncStorage 6](#_Toc532677416)

[2.4 Rn-fetch-blob 6](#_Toc532677417)

[2.5 React Native Firebase 6](#_Toc532677418)

[2.6 React Native Video 7](#_Toc532677419)

[2.7 React Native Music Control 7](#_Toc532677420)

[2.8 React Navigation 7](#_Toc532677421)

[2.9 Redux 7](#_Toc532677422)

[3 THIẾT KẾ KIẾN TRÚC 9](#_Toc532677423)

[4 THIẾT KẾ MÀN HÌNH 10](#_Toc532677424)

[5 CÁC KỸ THUẬT CHÍNH 11](#_Toc532677425)

[5.1 Fetch Data From Nhaccuatui.com 11](#_Toc532677426)

[5.2 Save Song Data To Local 12](#_Toc532677427)

[5.3 Saving Song Data to Cloud Firestore 13](#_Toc532677428)

[6 CÀI ĐẶT 16](#_Toc532677429)

[7 KẾT LUẬN 17](#_Toc532677430)

[7.1 Môi trường phát triển và Môi trường triển khai 17](#_Toc532677431)

[7.2 Kết quả đạt được 17](#_Toc532677432)

[7.3 Hướng phát triển 18](#_Toc532677433)

[8 TÀI LIỆU 19](#_Toc532677434)

# GIỚI THIỆU

## Về đề tài:

*Anh/Chị hãy trình bày nội dung đề tài trong phần này.*

*Dưới đây là một số gợi ý. Anh/Chị tự quyết định những nội dung nào phù hợp với đề tài của mình để trình bày phần Phát biểu bài toán*

* *Trình bày khảo sát hiện trạng:*
  + *Nhu cầu thực tế của đề tài*
  + *Hiện trạng của đơn vị cần xây dựng phần mềm (cơ cấu tổ chức, các quy trình nghiệp vụ hiện đang thực hiện, hiện trạng về mặt tin học của đơn vị…)*
  + *Các phần mềm cùng loại hiện đang có, hoặc hệ thống Tin học đang được sử dụng*
  + *Những hạn chế hay những vấn đề còn tồn tại trong những phần mềm đang có*
* *Xác định các yêu cầu của hệ thống. Có thể trình bày các biểu mẫu và quy định.*

Đây là một web site nghe nhạc trực tuyến giúp mọi người sử dụng tiếp cận âm nhạc dễ dàng hơn.

All music data is fetched from nhaccuatui.com.

Đồ án được phát triển dựa trên: [*https://github.com/aakashns/reactmusic*](https://github.com/aakashns/reactmusic)

Mã nguồn của đồ án: [*https://github.com/alwayssmile11a1/MusiClover*](https://github.com/alwayssmile11a1/MusiClover)

## Tính năng

* Listen to music.
* Top 100 songs in different genres.
* Top artists.
* Save songs to online personal account.
* Download songs to your local device.
* Create online or offline lists.
* Background music listening.

## Nhóm thực hiện

* Lê Xuân Tiến - 15520884
* Nguyễn Đức Thông - 15520856

# CÔNG NGHỆ SỬ DỤNG

*Mục này tập trung giới thiệu các công nghệ chính đã được sử dụng trong việc phát triển Scheduler. Lưu ý rằng trong Scheduler có thể còn sử dụng một số công nghệ khác, dưới đây chỉ là những công nghệ cơ bản nhất mà bạn nên lưu ý.*

## 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.

## Fetch API

Fetch API được cung cấp bởi react. Fetch API là một API đơn giản cho việc gửi và nhận requesst bằng js. Với fetch thì việc thực hiện các yêu cầu web và xử lý phản hồi dễ dàng hơn so với XMLHttpRequest cũ.

*Getting started:* [*https://facebook.github.io/react-native/docs/network*](https://facebook.github.io/react-native/docs/network)

## AsyncStorage

Also provided by React Native, AsyncStorage is a simple, unencryped, asynchronous, persistent, key-value storage system. However, AysncStorage is currently only capable of saving simple, small JavaScript objects.

*Getting started:* [*https://facebook.github.io/react-native/docs/asyncstorage*](https://facebook.github.io/react-native/docs/asyncstorage)

## Rn-fetch-blob

To save large data like a song, it would be a bad idea to somehow convert it into string, save and load it later by AsyncStorage. Rn-fetch-blob solved this problem by streaming response data directly into a file in a temporary path without any file extension. Together AsyncStorage and Rn-fetch-blob make perfect combination to save data to local storage.

*Getting started:* [*https://github.com/joltup/rn-fetch-blob*](https://github.com/joltup/rn-fetch-blob)

## React Native Firebase

Firebase is Google’s mobile platform that provides a set of tools to facilitate the development of your app such as authentication, hosting, realtime database and more. To interact with Firebase, React Native Firebase offers a light-weight layer on-top of the native Firebase SDKs (IOS & Android) to make using Firebase simple.

In this specific project, React Native Firebase is used to create online personal account with Firebase Authentication and also save data remotely with Cloud Firestore.

*Getting started:* [*https://github.com/invertase/react-native-firebase*](https://github.com/invertase/react-native-firebase)

## React Native Video

React Native Video offers a really nice component to play music or video on your device. It can also play music on the background thread which suits the need of a music application.

*Getting started:* [*https://github.com/react-native-community/react-native-video*](https://github.com/react-native-community/react-native-video)

## React Native Music Control

React Native Music Control provides an easy way to display which song is now playing on lockscreen and handle control events.

*Getting started:* [*https://github.com/tanguyantoine/react-native-music-control*](https://github.com/tanguyantoine/react-native-music-control)

## React Navigation

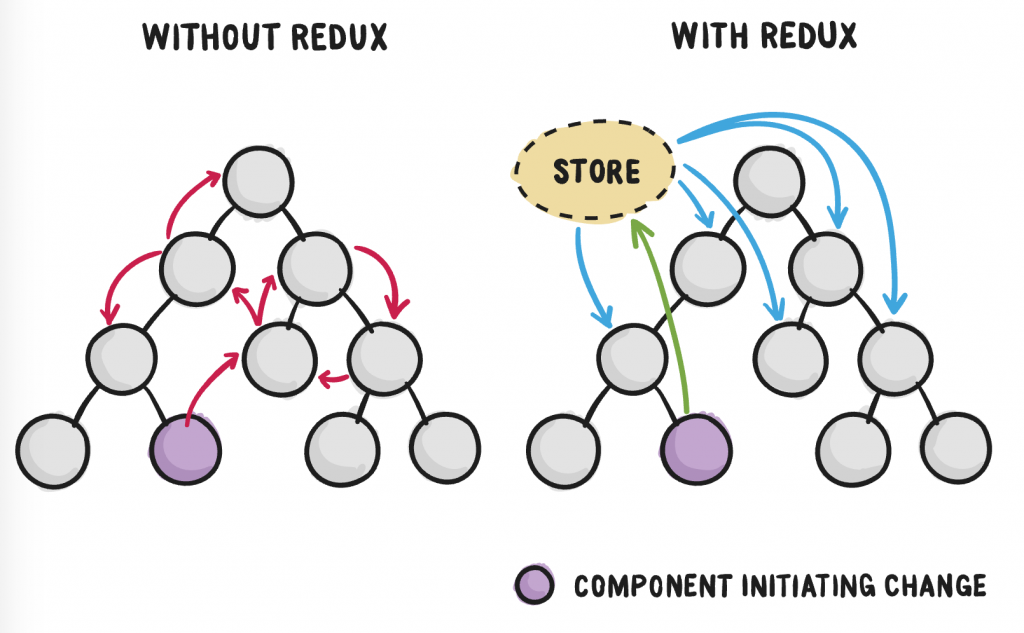
Since React Native itself doesn’t really offer any easy way to navigate between screens, React Navigation was born from the React Native community’s need for an extensible yet easy-to-use navigation solution.

*Getting started:* [*https://reactnavigation.org/docs/en/getting-started*](https://reactnavigation.org/docs/en/getting-started)

## 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.

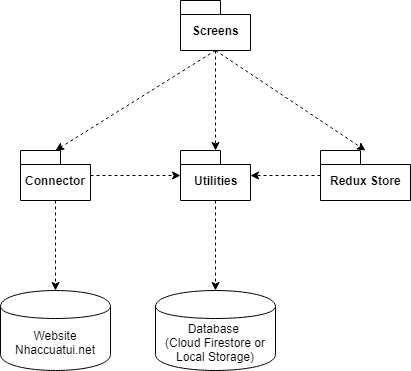
Redux là một thư viện Javascript giúp chúng ta giải quyết vấn đề đau đầu này bằng cách lưu trữ state của các component ở một store bên ngoài chung. Như vậy, việc quản lý các state này sẽ dễ dàng thuận tiện và có thể truy cập ở bất kì component nào mong muốn. Redux có thể kết hợp với React Native một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



*Bắt đầu với Redux:* [*https://redux.js.orgit*](https://redux.js.orgit)

# 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

# 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.

## Lưu trữ dữ liệu bài hát vào Local Storage

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) => {

})

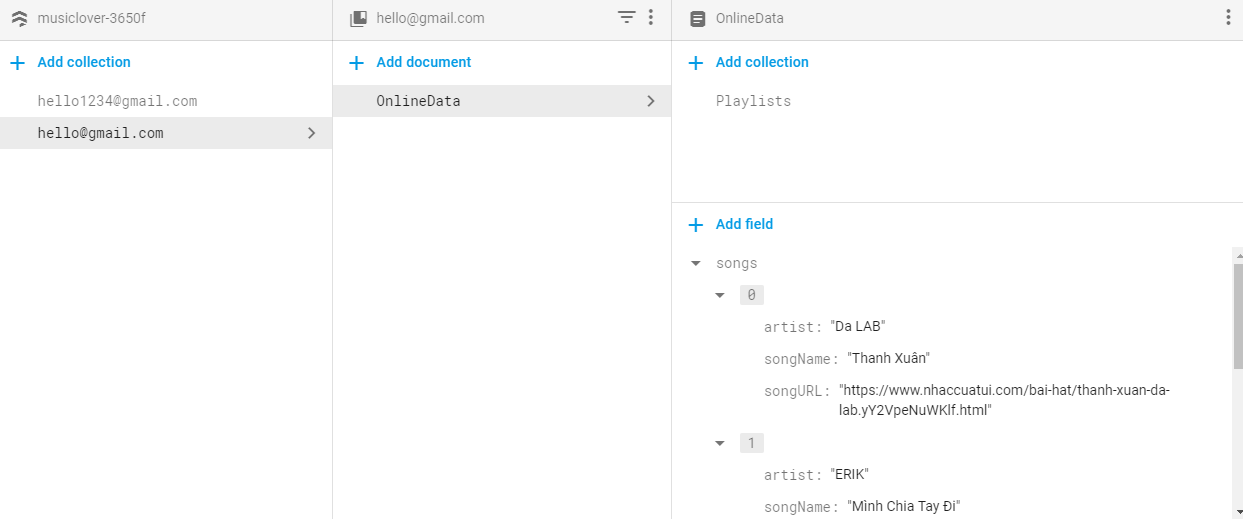
.catch((error) => {

console.log(error);

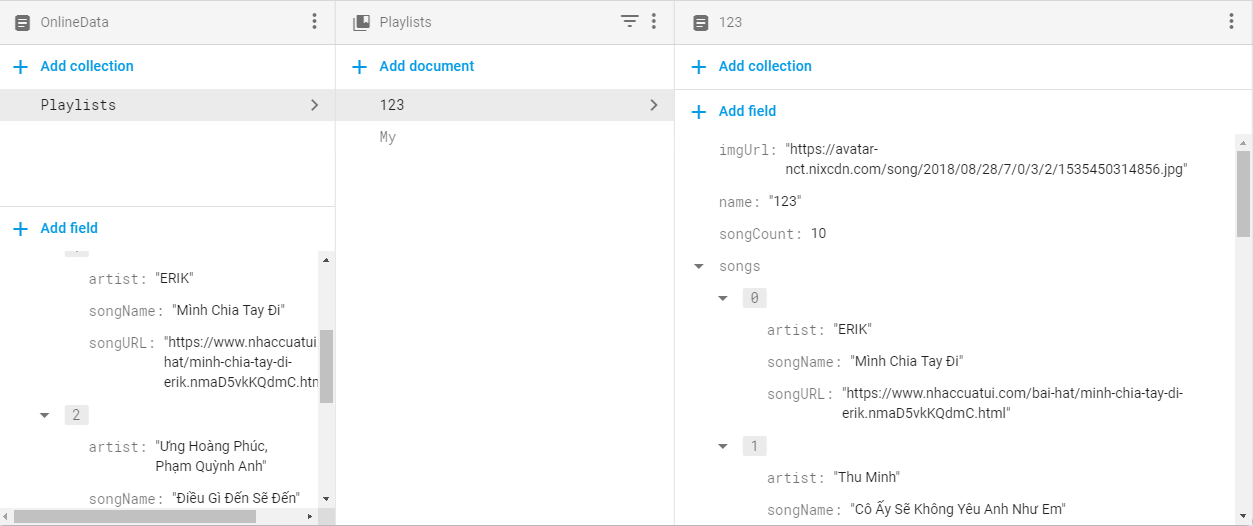
});

}

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 start** để cài đặt các module hay package yêu cầu bởi ứng dụng.
2. **react-native run-android** hoặc **react-native run-ios** để chạy ứng dụng trên máy ảo hoặc một thiết bị di động mong muốn.

Link dưới đây cung cấp cái nhìn hoàn thiện, rõ ràng hơn về việc cài đặt và chạy project:

[*https://facebook.github.io/react-native/docs/getting-started*](https://facebook.github.io/react-native/docs/getting-started)*.*

# KẾT LUẬN

## Môi trường phát triển và Môi trường triển khai

*Anh/Chị hãy ghi rõ* ***Môi trường phát triển ứng dụng*** *(chính là môi trường đã sử dụng để thực hiện đề tài):*

* *Hệ điều hành*
* *Hệ quản trị cơ sở dữ liệu (nếu có dùng)*
* *Công cụ dùng để phân tích, thiết kế (ví dụ Rational Rose…)*
* *Công cụ đã dùng để xây dựng ứng dụng (ví dụ: Visual Studio.NET 2005)*
* *Các thư viện đã sử dụng (ví dụ: Infragistics, Janus…)*

*Anh/Chị cũng ghi rõ* ***Môi trường triển khai ứng dụng*** *(cụ thể là muốn chạy được ứng dụng của Anh/Chị thì cần phải thiết lập cấu hình máy tính như thế nào, cần cài đặt những phần mềm hỗ trợ hay thư viện gì…)*

*Ví dụ:*

* ***Môi trường phát triển ứng dụng:***
  + *Hệ điều hành: Microsoft Windows XP (SP2)*
  + *Hệ quản trị cơ sở dữ liệu: không dùng*
  + *Công cụ phân tích thiết kế: Rational Rose 2002*
  + *Công cụ xây dựng ứng dụng: Visual Studio.NET 2005*
  + *Các thư viện đã dùng: Infragistics 2006 Vol. 1*
* ***Môi trường triển khai ứng dụng:***
  + *Hệ điều hành: Microsoft Windows*
  + *Cần cài đặt .Net Framework 2.0*
  + *Khi chạy ứng dụng, cần chép và cài đặt đầy đủ các tập tin DLL của thư viện Infragistics 2006 Vol. 1*

## 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 …)*

## Hướng phát triển

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

# TÀI LIỆU

* [*https://facebook.github.io/react-native*](https://facebook.github.io/react-native): Website chính thức của React Native cung cấp khá đầy đủ các hướng dẫn về cách cài đặt cũng như sử dụng React Native.
* [*https://hackernoon.com/building-a-music-streaming-app-using-react-native-6d0878a13ba4*](https://hackernoon.com/building-a-music-streaming-app-using-react-native-6d0878a13ba4): Bài hướng dẫn cơ bản về sử dụng React Native Video để hiện thực hóa ứng dụng nghe nhạc trực tuyến.
* Và nhiều nguồn tài liệu khác …