**University of Information Technology**

**Vietnam National University - Ho Chi Minh City (VNU-HCM)**



**Project**

**Music Streaming App – MusiClover**

**Course : Mobile Programming**

**Lecturer : Hoang Van Ha**

**Group : Tran Thi Thanh Thao – 15520820**

**Truong Ngoc Son – 15520740**

**Nguyen Duc Thong – 15520856**

***Ho Chi Minh City, December 4th 2017***

Table of Contents

[1 Introduction 4](#_Toc531967039)

[1.1 Description: 4](#_Toc531967040)

[1.2 Features 4](#_Toc531967041)

[1.3 The team 4](#_Toc531967042)

[2 Main Technology 5](#_Toc531967043)

[2.1 React Native 5](#_Toc531967044)

[2.2 Fetch API 5](#_Toc531967045)

[2.3 AsyncStorage 5](#_Toc531967046)

[2.4 Rn-fetch-blob 5](#_Toc531967047)

[2.5 React Native Firebase 6](#_Toc531967048)

[2.6 React Native Video 6](#_Toc531967049)

[2.7 React Native Music Control 6](#_Toc531967050)

[2.8 React Navigation 6](#_Toc531967051)

[2.9 Redux 7](#_Toc531967052)

[3 Interesting Techniques 8](#_Toc531967053)

[3.1 Fetch Data From Nhaccuatui.com 8](#_Toc531967054)

[3.2 Save Song Data To Local 9](#_Toc531967055)

[3.3 Saving Song Data to Cloud Firestore 10](#_Toc531967056)

[4 Installation 13](#_Toc531967057)

[5 Credit 14](#_Toc531967058)

# Introduction

*This section is just a short description of what this application can do and the team behind it.*

## Description:

MusiClover is a simple streaming music application that helps you listen to music on your mobile devices.

All music data is fetched from nhaccuatui.com.

## Features

* 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.

## The team

* Tran Thi Thanh Thao: Team leader – Sofware development fresher (Java) at VNG.
* Nguyen Duc Thong: Team member – Game development fresher (Programming) at VNG.
* Truong Ngoc Son: Team member - No experience yet.

# Main Technology

*This section will guide you through some of the main components that helped this application come to life.*

## React Native

**React Native** is a popular JavaScript framework for building native mobile apps. It uses the **React** framework and offers large amount of inbuilt components and APIs.

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

## Fetch API

Fetch API is provided by react-native to do some networking stuff such as fetch data from remote websites. If you have already been familiar with other networking APIs like **XMLHttpRequest**, getting started with React Native Fetch API is not a big deal.

*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

If you have used React Native, you know how frustrating it is to pass states from one screen to another. There are many state containers to solve this problem, but one of the most common one to use with React Native is Redux. Redux also helps you write applications that behave consistently, run in different environments and easy to test.

*Getting started:* [*https://redux.js.orgit*](https://redux.js.orgit)

# Interesting Techniques

*This section is intended to guide you through some of the techniques that you might be curious about and how you can go about them.*

## Fetch Data From Nhaccuatui.com

For example, If you want to get current top 100 pop US/UK songs, you can go to: <https://www.nhaccuatui.com/playlist/top-100-pop-usuk-hay-nhat-va.zE23R7bc8e9X.html?st=1>

Hit F12 to inspect the website, inside the elements section, you can see html elements for this specific page being exposed and this is all we need.

The problem now is only the matter of how you can get html elements down and get the data you care about. Luckily, **Fetch API** provides just enough functionality to do it.

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;

}

Look closely at the code above, you can see that **async function** is being used and the reason for this is quite obvious. When we call the **fetch function** provided by **Fetch API,** it actually returns a **Promise** toindicate that it will try to fetch the data that we want and return a response as soon as possible, and it will not block the main thread of the application. In short, Fetch is an asynchronous function. This nature of fetch function is handy and appropriate because we don’t want our application to stop running and wait until we receive the response as it can cause a bad experience for users. But at the same time, since fetch is asynchronous, there is no guarantee that our **listSong** can be filled in before it gets our of the function. So, one solution here is to wait until the fetch function finishes by put **await** keyword in front of it and mark the whole **getTop100 function** as async. In this way, we can both guarantee that we can get the result we want and still don’t block the main thread of the whole application.

Another key thing to note here is that the response actually contains information in a html-page format, so the easiest way to get appropriate data from it is to use regex. Since regex is a completely different subject, we will not discuss it here.

## Save Song Data To Local

As mentioned in section 2, **Rn-fetch-blob** and **AsyncStorage** make perfect combination to download and save song data to local. AsyncStorage is in charge of keeping track of song lists, the number of song in a list, etc. and most importantly, the local path to the song that has been downloaded by Rn-fetch-blob.

Rn-fetch-blob has done most of the heavy-lifting job, so downloading a file from a remote website is pretty straightforward.

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.

The code below shows how to save a pair key-value to local storage:

storeData = async (name, value) => {

try {

await AsyncStorage.setItem(name, value);

} catch (error) {

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

}

}

## Saving Song Data to Cloud Firestore

Although there are many tutorials on working with Firebase by **React Native Firebase** out there and you can easily find them online, we found it necessary to discuss the way we save song data to **Cloud Firestore**.

There are two key areas to take care of: Logging in and using that information to reference song data to appropriate users.

Registering and logging in to Firebase become easy with 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) => {

})

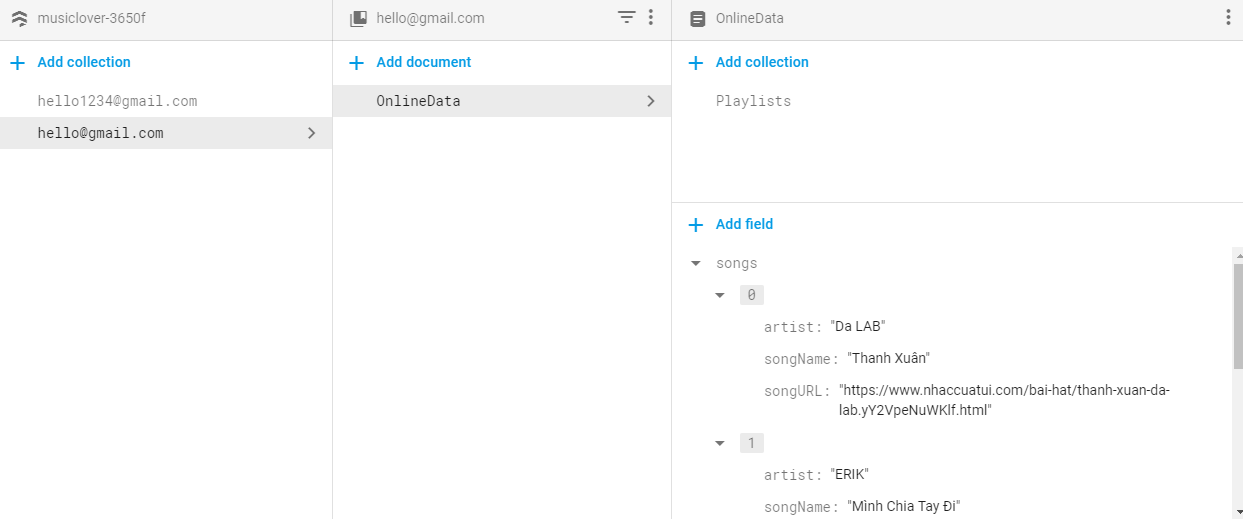
.catch((error) => {

console.log(error);

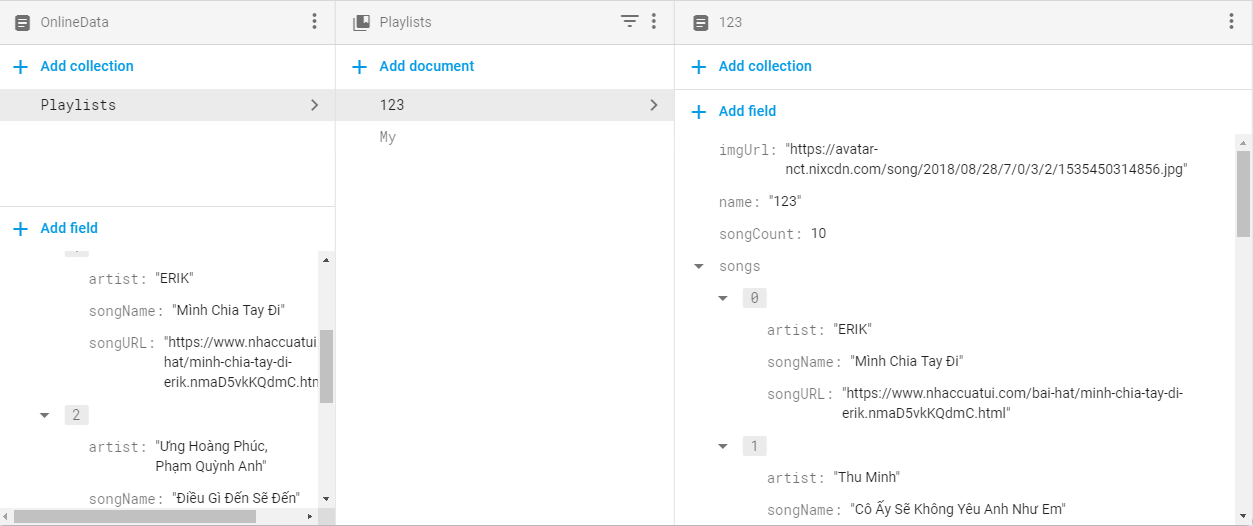
});

}

There is nothing much to discuss here since React Native Firebase have done pretty much everything for us. The problem we care about in this section is actually the way we save user data.



Look at the image above, you can see that we reference data of users by their email since it is uniquely identified. For each user, we create a document called **OnlineData** and in this specific document, you can see that we have a list of all the songs that the user have added and also a collection of playlists.



Each user can have as many playlist as they want. Each playlist contains some basic information, for example the number of songs in the list, the represented image and an array of songs.

The code example for saving these information to Cloud Firestore:

let userCollection =

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

userCollection.set({

songs: ...

})

# Installation

Open a terminal window and navigate to the project directory and run:

1. **npm start** to install all packages needed to run the app.
2. **react-native run-android** or **react-native run-ios** to run the application on your virtual device on your computer.

You can find more information about how to install and run react-native app on the offical react-native website [*https://facebook.github.io/react-native/docs/getting-started*](https://facebook.github.io/react-native/docs/getting-started)*.*

# Credit

This project was made possible thank to:

* <https://facebook.github.io/react-native>: React-native official website provides really nice tutorials on basic things about react-native.
* <https://hackernoon.com/building-a-music-streaming-app-using-react-native-6d0878a13ba4>: A small, simple tutorial on react-native-video.
* And various other sources …