Undergraduate Final Year Project

Research and apply Nodejs and Reactjs to cutting and sharing music as the social media website.

Vu Thanh Trung

Bachelor of Science with Honours in Computing

Banner ID: 001176776

Lecturers: Mr Thanh

Contents

Table of Figures

ACKNOWLEDGEMENT

First, I would like to thank the administration, along with the professor at the University of Greenwich, for their dedicated teaching and best support for my study during my studies. Helping me learn a lot of new and useful knowledge and studying in an international environment are really valuable experiences and knowledge for me personally. I feel that I am equipped with solid equipment for my future development.

Next, I would like to thank the school for providing research facilities as well as materials and the professors of the school who have taught me wholeheartedly on the issue that I still have questions about. Besides, I would like to express my deep gratitude to Mr. Thanh, who supervised me during the project implementation. I really admire the invaluable knowledge that he has supported me in the process of completing my final project. With the enthusiasm, sincerity, and motivation that he has imparted to me, he has helped me many times to overcome difficulties in the process of completing the project. He taught me how to research as well as how to implement the project in the right process, logically and how to write the report in the clearest and accurate way. I feel very honored to have Mr. Thanh oversee my project.

Finally, I am very grateful to my parents during the past time for always caring, caring, encouraging, and creating conditions for me throughout my development journey. In addition, I also want to send my thanks to my brothers and sisters who have always supported them and helped me during the last time. Besides, my classmates are also the people who have accompanied and helped me a lot during the completion of the project, so I want to send them my sincerest thanks.

My project research and apply Nodejs and Reactjs to cutting and sharing music as the social media website. Although I tried very hard to complete this project well. But it will also not avoid the bad points as well as limitations. Therefore, I am willing to accept comments and evaluations to continue to improve and develop the project.

Abstract:

In the era of social networks in recent years, I have found that they really bring a lot of useful and positive things to each of us's lives. However, social networks also have their limitations. As seen, users can freely make statements that are misleading, controversial as well as inciting others. There is also the issue of sharing offensive, depraved images and videos. Sharing fake information causes a stir. Because of the issues that I mentioned above, I decided to implement the project of an audio social network, a social network where users can only post an article with a song or piece of music available in the list. the site's music book only. Since the music will be managed and approved by myself in the posts of this social network, I can manage the most comprehensive way. I'll search for free sounds and tracks on their behalf. Accordingly, I will integrate more tools to help them cut music and download it. Users can download the whole article or cut the download to use according to their intended use.

In this research project, I will apply Reactjs and Nodejs to my project. I will use Reactjs for the client and Nodejs for the server of the project. Using Reactjs for the client, according to me, is very effective for this social networking project of mine. Because Reactjs uses virtual Dom to speed up the application, while traditional methods display all the data to the DOM, Reactjs only displays the right part of the data that needs to be displayed(1). Besides, Reactjs can reuse a component many times, so maintenance, as well as upgrade changes, will be very easy (1). Next is Nodejs, because Nodejs and Reactjs are both JavaScript libraries, that's why I chose them to accompany each other in this research. Nodejs gives me asynchronous (non-blocking) processing for different tasks of the website for the fastest and most optimized information transmission and processing (2). Nodejs also make it easier for me to maintain and extend my system. In addition, receiving and handling multiple connections with just a single thread of Nodejs is also something that I am very interested in. Through the above, I see that Reactjs and Nodejs applied to my research and development of audio social networking projects are really reasonable and optimal.

In the continuous development of technology, it is inevitable that the following studies will inherit the good values and overcome the disadvantages of the previous studies. And the same goes for my project. My research project will bring practical and good values of a social network to people. I will research a social network where sound will inspire and spread positive emotions for users. And especially the absolute guarantee on the issue of fake and unhealthy information as well as violent videos, images, and depraved cultural products.

CHAPTER 1: INTRODUCTION

My link code project: https://github.com/trungvu0612/Social_Media_Website.git

1.1. Overview:

My project overview is based on the research and development objective of an entertainment social networking website specializing in audio. To develop the positive aspects of social networking and remove its drawbacks. Music is something that helps us, humans to be truly entertained when immersed in it, helps us read a shared article, a line of someone's confided, easily absorb and feel the content in a meaningful way. most comfortable. According to Gethealthystayhealthy, "While the effects of music on people are not fully understood, studies have shown that when you hear music to your liking, the brain actually releases a chemical called dopamine that has positive effects on mood. Music can make us feel strong emotions, such as joy, sadness, or fearsome will agree that it has the power to move us"(3). Some research sites also show that music has a very good effect on human health. That's why I came up with this audio social network idea and at the same time integrated the function of downloading music and cutting music for those who want to use the audio clip for their other purposes. In addition to the above outstanding features, my social network is like other social networks. Users can create and post articles with any music they want. It can be to share your feelings about this song, it can be to include a status line of yourself that day or it can also be to inspire your inspiration to others. All will be enhanced to perceive as well as absorb everyone's writing thanks to music. Besides, users can also surf the web while listening to music from the songs shared by others, they can also add that song to their personal favorites to listen to again later. They will also be able to change their own personal information. Besides, they can also like and comment on other people's posts.

In terms of technology, the main technologies that I apply for the research and development of this project include 2 large libraries of Javascript that are quite famous today, which are Reactjs and Nodejs. Besides, I will also use some other technologies applied to the research and development of projects such as SASS, React Hook, Redux, Mp3 Cutter, MongoDB, and some other small technologies. With my research project on social networks, the real-time factor is one of the essential requirements of the project, so I choose Reactjs and Nodejs as two Javascript libraries that use JavaScript technology. Immovable processing technology with virtual DOM makes it possible that even small changes to the data will be rendered instantly on the web page without having to reload the entire page. Also, I will also study and use Mp3 Cutter one of

the libraries of Javascript and Nodejs module that allows me to cut mp3 files. These are all the main technologies that I researched and applied to my project.

1.2. Purpose:

In this research project, I am the only person doing research and development in the project. As mentioned above, the fact that social networks are now so popular with the world today, although they bring people a lot of great benefits, they are not without limitations. . In addition those benefits, they also carry potential risks for users such as fake news, fraudulent fake pages, spreading information as well as unhealthy images and videos. That's why my purpose in this research project is to research and apply Reactjs and Nodejs to social networking sites about audio and through that also add some utilities for typical users such as cut music and download music. It will be very safe for users because all the music used in this social networking site will be directly censored and uploaded by me to users. In addition, I am also the person who directly manages all posts, if there is a post that is not in line with user safety regulations, I will directly delete that post. Music is also a tonic for everyone to help people entertain and feel the messages that the poster conveys better. With the benefits that this research project brings to me, I think this research is really worth it. The project was implemented from January 15, 2021, to October 10, 2021. The project will be implemented with fairly new and popular technologies today, which are Reactjs, Nodejs, and MongoDB. The development of these technologies also provides a sizable support community and as for me, I can judge that the project can complete all the basic functions for users. As for the function of cutting music, I am researching a module of Nodejs to apply for this, which is MP3 Cutter. And I hope with my ability I can successfully research and apply it to the project.

1.3. Scope:

My capable audio social network project includes the main functions of a social network. The website will allow users to log in, users can change their information such as name, avatar, phone number, and age. Users can search for music on the search bar. They can create articles with available music on the page uploaded by the admin, can download that song, or cut a part of their favorite and download it. All user posts will be displayed in two places, the homepage of the website and their personal page. Other users can see everyone's posts on the public home page, and if there's a song they like, they can like, comment, or add the song to their favorites. Every time a user goes to the web and clicks on a song, the music will be played directly to the user. Every time someone comments on someone's post, they will receive a notification that someone

commented on their post. My social networking site also has its own admin page to manage. The admin page can be used to upload music for users to use, can edit song information, as well as delete music if not needed. The data in the admin page will also be visualized providing information about the number of posts published and some other information about the website. Here admin will also directly manage the posts and they can delete the post if really necessary. The above are the basic functions of my research project. With the online chat function, in my opinion, as well as collecting opinions from users, most of them think that adding a social networking site that sounds like mine is unnecessary. Since there are actually so many social networks with large userbases that have developed this chat function now and users are so used to chatting on those sites that they won't really need the function. This is on a fresh website. However, with my ability and doing all the projects alone, it is inevitable that the website will be limited in terms of features for users.

1.4. Organization:

Below is the full structure of my report on the audio social networking site research project. Arranged by me in order to help readers easily go deep and understand this research of mine.

- Section 1: Overview of the social networking site of the time, as well as the purpose and overview of the audio social networking research project.
- Section 2: Survey on issues surrounding social networks. Provides an overview of social networks. Provide solutions to the problems of current social networks. From there come up with ideas for projects to overcome the above problems. The technologies are selected with the appropriate advantages for the project.
- Section 3: Focus on studying Reactjs and Nodejs technologies. And other small technologies are integrated and used in the project.
- Section 4: Analysis of the requirements of the research project.
- Section 5: Research and design for the project.
- Section 6: Illustrate how to implement a research project and how to apply its technologies.
- Section 7: Check the project's defects and study the functions and technologies that can be applied in the future to develop the project.

- Section 8: Summarize the research and development of the research project. Difficulties, lessons, and experiences have been drawn after completing the research.

CHAPTER 2: LITERETURE REVIEW

Currently, the rapid development of technology 4.0 has led to an increasing number of people's entertainment needs. And a large part of the impact that leads to the gradual digitization of entertainment is due to the strong outbreak of the covid-19 epidemic around the world. Therefore, people's demand for online entertainment is increasing day by day due to the complicated evolution of the epidemic and because their living standards are also increasing due to the growing society. But this also means that entertainment without proper and reasonable selection is really dangerous and leads to consequences that we do not want. Especially for today's young people, being exposed to technology too early can easily make their development go wrong.

According to the data that I collected from the Cftsoft site, the top 10 apps downloaded from the Apple App Store and Google Play Store as of December 25, 2020, include Tiktok, zoom, WhatsApp, Facebook, Messenger, Instagram, Google Meet, Google Classroom, Youtube and Microsoft Teams(hotro@cooftech.com, P., 2019). Among those apps, entertainment social networking platforms account for 60%. Among them, applications like TikTok and Facebook are two entertainment social networking platforms that have the largest number of users in recent years. With Facebook, as of the first quarter of 2021, there are about 2.85 billion global monthly active users(Statista, 2020). As for TikTok, there are about 689 million global active users as of January 2021(Social Media Marketing & Management Dashboard, 2021). Those huge numbers really prove to us that the entertainment demand during the epidemic season and in the future of current users is very high and potential. And according to statistics from the Statista site, the age group from 25-34 years old is the age group using Facebook the most among the ages(Statista, 2020). As for Tiktok, about 69% of TikTok users are aged 13-24. according to backlinks source(News, V, 2021). Through the data that I have given above, I have drawn that the number of users of technology products on the internet today is young people. After hours of studying or working stress, the need for entertainment is inevitable and young people have many ways to relieve stress. And sites like Facebook or TikTok are extremely interesting ways of entertainment for young people. It will be civilized and positive if young people know where the limit is and should stop. On the contrary, it will be dangerous if you get too caught up in it. In

addition, today the traditional games: kite flying, shuttlecock, chess, rubric, chasing, and so on are no longer attractive to young people as much as they were in the past because now technology has developed too much(Backlinko, 2020). Every home has a computer, a phone equipped with a very powerful configuration and the internet is now ubiquitous everywhere with an increasingly improved access speed. This is an extremely good condition for online games and social networking sites to reach young people too soon and this can negatively affect the physical and mental development of young people. nowadays. Young people are the future of the country, of the world, so I came up with an idea that why don't we create a social networking platform to share audio. I understand that young people often like to show their ego and mood to everyone, but sometimes it's quite difficult to express it directly. My social network will be a social network specializing only in sharing the sound and status of the poster, I think it will be quite suitable and interesting for young people. Besides, it can also prevent the bad effects of social networks on users, especially young people who are not aware enough about society because of them. And I also want audio-related content creators to be able to find the right music and audio for their work on my audio social network. They can split a download or upload it for others to see and evaluate.

Currently, about the entertainment social network, as I mentioned above, there are pages like Facebook, TikTok that are the most popular ones today. Besides, there is another social network that has emerged recently, spoon radio, which specializes in audio. The websites on them have been many years ahead of me and have had the tremendous success that no one can deny. However, nothing is comprehensive. Regarding Facebook, they are the pioneers in this field and have existed until now so that clearly shows how good and useful their product is. Some of Facebook's most distinctive strengths are better socialization, better people-to-people connections, freebies, and business growth(Vinayak SP, 2019). Facebook is really successful and stands out for helping you and those around you access information from people everywhere in the world. From there, it gives users an overall view of the picture in the world as well as the surrounding places. Besides, the convenient connection only requires the internet, users can communicate and connect with their friends anywhere. Facebook also makes it easier for customers to find the items they need to know through ads, and businesses are more accessible than ever to their customers. However, they still have some disadvantages that I found through the process of using and surveying. The problem of stealing information and spreading unhealthy

and false information greatly affects the politics and economy of many countries around the world (LuatVietnam, 2021). Regarding Tiktok, the advantage is that it allows users to freely create good video content, providing quite quality entertainment moments for users (NamKyLanBlog, 2020). Finally, it's Spoon Radio. Their advantage is that young people can unleash their creativity, draw more attention to people with their voices, diverse and rich content. Extremely suitable for young people who are shy about their appearance(Nghĩa, 2020). However, the above social networking products all share many common disadvantages of today's social networks. The first is that they can't control the content that users post, users can unleash their creativity as well as post any content they want, so it will lead to the status of posting false information, unhealthy. In Vietnam, "The Department of Cybersecurity and High-Tech Crime Prevention and Control, the Ministry of Public Security, had to ask Facebook to remove 544 posts, personal accounts and fan pages on Facebook with bad and toxic content, trade-in counterfeit money, instructions for making weapons, explosives, and narcotics…"(banconannhandandientu, 2021). The second is addictive, reducing human-to-human interaction. Users are overwhelmed by the virtual world created by these social networks so that they immerse themselves in it and live a virtual life and only communicate with each other through their social networks. The third is the fact that they don't have clear and realistic creative audio charts yet. Social networks are where users want to unleash their creativity and catch up with the world's trends. But I see that users now see something on someone's social network, they like the sound but can't know the name or find the audio clip. They have to keep asking the poster what is the name of this song, what is this audio clip. And when they have that audio clip, they have to download and access an application or website that specializes in audio to cut it correctly. I find it really inconvenient.

It is for the above reasons that my audio social network was born to fulfill its mission. Create a healthy social network that involves only music and good tunes. They can only download, edit, and review audio tracks that I've rigorously vetted and posted to the music database. And the content that my social network is aiming for is the sound that will overcome the problems related to posting unhealthy content, spam content. Users can go to the music database to get to their account, create articles and post them on their personal pages, as well as rate the tune, that song on the chart and write a status line with it. Posting that melody or song on a personal page has helped amateur content creators and an audio social network to exchange reviews of that piece of music and sound. As well as helping them to edit, cut and

download that piece of music without having to spend as many steps as before. When users rate audio clips, melodies, or songs, they will be put on a ranking of music or audio tracks. Thereby users can grasp current music sound trends.

About the technology, I use for Reactjs and Nodejs projects l. These two technologies are quite new and have been used by many projects in recent times. I chose to use Reactjs for the front-end part of this project because Reactjs is a very suitable library for social media-related projects. The first thing that I choose Reactjs because of not using templates. Web applications now often use HTML templates but Reactjs is different, it uses components. This makes it easier to maintain and expand the system later, and to fix system security errors related to XSS(NordicCoder, 2018). The latter is well-suited for projects with constantly changing amounts of data such as social networks. React js will compare the change of values of the following renders with the previous render making the least change to the DOM. Finally, the fact that Reactjs works on the virtual DOM makes the web much faster. The virtual DOM is extremely useful for organizing the UI so that it can adapt quickly to changes and improve the user experience(CodeHub, 2021). Thanks to the above, Reactjs will better structure the UI to enhance and optimize the UX. It would be easier for me to increase user interaction with the system without fear of operational problems. And especially Reactjs Dom update is also very limited and system maintenance helps to save the maximum budget for my project. Next is Nodejs, I use Nodejs to interact Front-end with the database or collectively known as the back-end of my audio social networking system. The same is the library of Javascipt, so it is easier for me to interact between the two. Besides, such use is also easier for me to access and research and apply them to the project. My project is an audio social network project and they need to work in real-time so Nodejs is very suitable for this. Nodejs has a very fast processing speed because it uses an asynchronous mechanism. Microservices must interact with one another, and Node.js is one of the fast data processing engines available. Non-blocking algorithms are one of the key advantages of Node.js for web application development(techmaster.vn, 2021). Like Reactjs, Nodejs is very easy to extend and develop the project later. Because my project in the future will develop even more. Load balancing for each operating CPU core is handled by a series of modules in Node.js. One of the numerous advantages of the Node js module is that I can operate numerous nodes at once, with the environment automatically balancing the burden. Horizontal scalability is possible with Node.js, thus I can split my application into many instances. Depending

on the user's age, interests, location, language, and other factors, I offer them multiple versions of the app. This improvement personalization while lowering effort. Subprocesses, or actions that communicate quickly with one another and have a common origin, are how Node does this (techmaster.vn, 2021). The common point for my choice of these two technologies for this project is that both are Javascript libraries, so learning their syntax will be easier for me. Both are very easily accessible. They are all well suited for developing a single-page project. Both handle data very well for websites that retrieve information quickly and in real-time.

CHAPTER 3: TECHNOLOGY RESEARCH:

In this section, I will discuss some of the technologies which I have used in my project. By doing this, I can have a deeper understanding of each technology that will be used in my project

3.1. Sassy Cascading Style Sheets (SCSS):

First of all, Scss is known as Sassy Cascading Style Sheets which is an upper-level version of Cascading Style Sheet (CSS). Scss is a file that contains not only related CSS things but also more other features. Scss helps the developers make their codes look clean and clear by shortening the code with the created variables (GeeksforGeeks. 2021). Hampton Catlin has come up with an idea to design Scss and Natalie Weizenbaum developed it in 2006. In 2008, Chris Eppstein, a Compass designer, subsequently joined main and developer Weizenbaum to continue extending Scss . The first published of Scss in 28th November 2006, which means 14 years ago. Through 12 years of improvement, the very first steady version, which is the 1.0.0 version, was appeared on 26th March 2018.

Any technology always has its pros and cons. Sass can help developers efficiently write coding and quickly by reducing the code lines that can be repeated. With the large library and opensource, scss is compatible with any version of CSS. Moreover, SCSS also provides some of the extensions for developers to choose from and so on. On the other hand, Sass is a technique that needs tons of time to learn and practise. And to compile as CSS, some extra steps need regularly to be done and it is very hard to solve because the browsers could not know the scss syntax

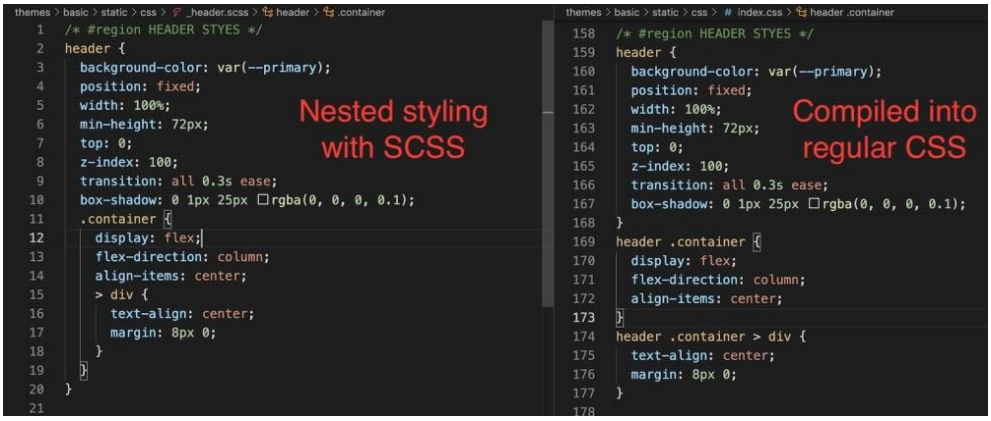

Figure 1: SCSS

3.2. JavaScript.

In the next thing, I will introduce the Javascript technique. JavaScript is a computer language that is widely used in web development, online apps, game creation, and other applications. It enables you to integrate dynamic features on web pages that you would not be able to perform with only HTML and CSS (6). Netscape initially created it as a way to bring dynamic and interactive components to web pages. JavaScript is a client-side scripting language, which implies that the source code is interpreted by the client's web browser rather than the webserver. Brendan Eich wrote a javascript prototype for Netscape in 1995 to defend their notion about Javascript, which Marc Andreesen termed Mocha. Early in December 1995, as Java's popularity grew, the language was renamed again, this time in JavaScript .

There are manifold advantages of javascript. As JavaScript is an 'understood' language, it decreases the time necessary for compiling in other programming languages such as Java. JavaScript is also a client-side script, which speeds up program execution by eliminating the time necessary to connect to the server. In addition, JavaScript offers developers a variety of APIs for generating eye-catching web pages. Drag-and-drop components or sliders may provide a rich interface to websites. This improves user interaction on the website. Moreover, JavaScript integrates seamlessly with other programming languages, which is why many developers use it for creating a wide range of applications. We may include it in any webpage or any computer language's script. However, the JavaScript code is visible to the user, it might be used maliciously by someone else. Using the source code without authentication is one of these behaviours. Furthermore, it is relatively simple to insert code into the site that jeopardizes the security of data transmitted via the website. Besides that, JavaScript is interpreted differently in various browsers. As a result, before publishing the code, it must be tested on a variety of systems. Some

new functionalities are not supported by older browsers, therefore we must double-check them and also (9). All web pages with portable elements are supported by Javascript, for example, the drop-down list is one of those.

Figure 2: Javascript

3.3. Node.JS.

Nodejs will be discussed in this section. We can understand Node.js comprises an open-source server-side runtime environment based on the V8 JavaScript engine in Chrome. It offers an event-driven, non-blocking (asynchronous) I/O and cross-platform operating system for developing extremely scalable server-side JavaScript applications. Node.js can also be used to develop a wide range of applications, including command-line programs, web-based applications, real-time chat applications, REST API servers, and so on. Nevertheless, it is mostly used to construct network applications such as web servers, much the same as PHP, Java, or ASP.NET. The first node.js version was released about 12 years ago which was 27th May 2009 by OpenJS Foundation. And It was written and presented by Ryan Dahl in the same year .

In 2018, there was a survey report made by the NodeJs team. This showed that there was 85 per cent of users had priority to use the Javascript runtime for their web apps. Nodejs has many advantages which make lots of people choose it. One of the pros of Nodejs is the ability to create supercharged apps that display results in a matter of seconds. The ability of Node.js based web apps to multitask is quite beneficial. Its single-threaded, event-driven design, unlike other platforms, effectively processes numerous concurrent requests without cluttering RAM. Furthermore, its event-loop and non-blocking I/O operations allow code to be implemented at a

rate that has a substantial influence on the overall performance of the program. One more benefit of Nodejs is cost-effective. Because Node.js allows developers to create server-side code in Javascript, so the developers can write code for both the frontend and the backend with simplicity. One of the most significant advantages of node.js is that it eliminates the unneeded two resource teams, as well as saves a significant amount of time, money, and resources for overall project development. The development of LinkedIn had regained the performance impulse by cutting down 27 severs from 30 servers. This was very helpful at that time because they could now turn their emphasis away from troubleshooting and toward application development. Nevertheless, The frequent API changes, which are often backwards-incompatible, that drive Node.js users up the wall are one of the most major complaints mentioned by Node.js users. Consequently, this compels them to alter the access code regularly to keep up with the current version of the Node.js API. Because Nodejs has been created in recent years so there is a lack of documentation and library support for amateur developers.

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Figure 3: Nodejs

3.4. React.JS:

React exactly is a JavaScript toolkit that allows you to construct speedy and interactive user interfaces for websites and mobile apps, which is quite significant. It is a component-based, open-source front-end library that is exclusively responsible for the application's view layer. The view layer in Model View Controller (MVC) architecture is in charge of how the app appears and performs. Especially React may also be used to for the most part render on the server using Node, and it can power native apps with React Native in a subtle way. React supports one-way reactive data flow, which eliminates boilerplate and makes reasoning easier than conventional

data binding, which is fairly significant. On 29th May 2013, Jordan Walke, a software developer at Facebook, invented React.

One of the benefits of Reactjs is using virtual DOM from JavaScript. Because JavaScript virtual DOM is fairly quicker than conventional DOM, this will increase app speed, which is quite significant. More than that, Reactjs is able to use multi-framework, from client to server-side and so on, very contrary to popular belief. In addition, Reactjs provides the readable of elements and data models which is very helpful in maintaining larger apps. The most advantage of Reactjs is simple to learn and simple to use. Most programmers with a thorough understanding of JavaScript can learn to React in a matter of days. With React, the developer only really needs a fundamental understanding of HTML and CSS. Because it mostly is widely used, we may spread a bigger net for developing talent and acquiring fairly more qualified individuals . Nonetheless, because React Js just covers the app's display layer, the programmers will also need to combine it with some other technologies to develop a comprehensive development toolkit. Because of whole React ecology is growing so quickly, and technological advances are being released so rapidly, documentation is frequently having trouble keeping up .

```
<html>
  <head>
    <div>
          <div>

              <form method="post" action="#" id="formvalue" onkeyup="
              drawChart()" />

              </form>
          </div>
   </div>



   <script type="text/javascript" src="https://www.google.com/jsapi"></
     script>
   <script type="text/javascript">

   var bid = 43;
   var ask = 21;


     google.load("visualization", "1", {packages:["corechart"]});
     google.setOnLoadCallback(drawChart);
     function drawChart() {
       var data = google.visualization.arrayToDataTable([
         ['Price', 'Quantity'],
         ['Value #1', bid],
         ['Value #2', ask],
       ]);
```

Figure 4: Reactjs

3.5. React Hook:

React Hooks are built on three key React principles: declarative, component-based, learn once, and write everywhere. They attempt to encapsulate state management by making use of existing JavaScript functionalities. As a consequence, we no longer need to acquire and comprehend particular React capabilities; we can just utilize Hooks with our current JavaScript understanding. We can fix all of the above-described difficulties by using Hooks. Hooks are simple functions that may be called in function components, therefore we really don't need to utilize class components any longer. We also no longer need to employ higher-order components and render properties for contexts because we can simply use a Context Hook to acquire the data we want. Furthermore, Hooks allow us to reuse stateful functionality between components without developing higher-order components. React Hook is introduced on 26th December 2018 by Ryan Florence Sophie Alpert, and Dan Abramov.

Hooks are a new functionality that was released in React 16.8. It enables developers to leverage state and other React capabilities without having to create a class. React Hook is a function that allows function components to "hook into" React state and lifecycle features. It does not function in classes. Hooks is backwards-compatible, which guarantees there are no breaking changes. It also does not substitute your understanding of React fundamentals. Hooks are a new functionality that was released in React 16.8. It enables developers to leverage state and other React capabilities without having to create a class. React Hook is a function that allows function components to "hook into" React state and lifecycle features. It does not function in classes. Hooks is backwards-compatible, which guarantees there are no breaking changes. It also does not substitute your understanding of React fundamentals. React Hooks is just only used in 2 cases. Hooks should not be called within loops, conditions, or nested functions. Hooks will be used at the highest management of React functions at all times. This condition guarantees that Hooks are requested in the same sequence whenever an element renders. Hooks cannot be called from ordinary JavaScript functions. Hooks can instead be called from React function components. Hooks can also be accessed via custom Hooks.

Figure 5 Reactjs hook

3.6. Mongo DB.

MongoDB is a data service that combines flexibility and scalability with both the querying and indexing capabilities that you require. MongoDB employs collections and documents rather than tables and rows as in traditional database systems. Documents are made up of key-value pairs, which are the fundamental unit of data in MongoDB. Collections are the equivalent of relational database tables in that they include collections of documents and functions. MongoDB is a database that first appeared in 2009.

It is a flexible database. MongoDB is a schema-less database that stores any form of data in a distinct document in a single collection. The plenty of fields, content, and size of the record might vary from one to the next. This device provides us with the flexibility and freedom to store data in various forms. Another benefit of the mongo database is sharing. We can keep vast amounts of data by distributing it among several servers linked to the application. There will be no failure condition if a server is unable to manage such large amounts of data. The name for this is "auto-sharding." Also, MongoDB is a database that is structured as a set of documents. Indexing makes it simple to find documents. As a result, it responds quickly to queries. MongoDB's performance is 100 times that of a relational database. On the other hand, for each value pair, MongoDB saves the key name. There is also data redundancy owing to the lack of join functionality. As a result, RAM is being used inefficiently. The document is no larger than 16MB in size. Nesting documents for more than 100 layers is not possible.

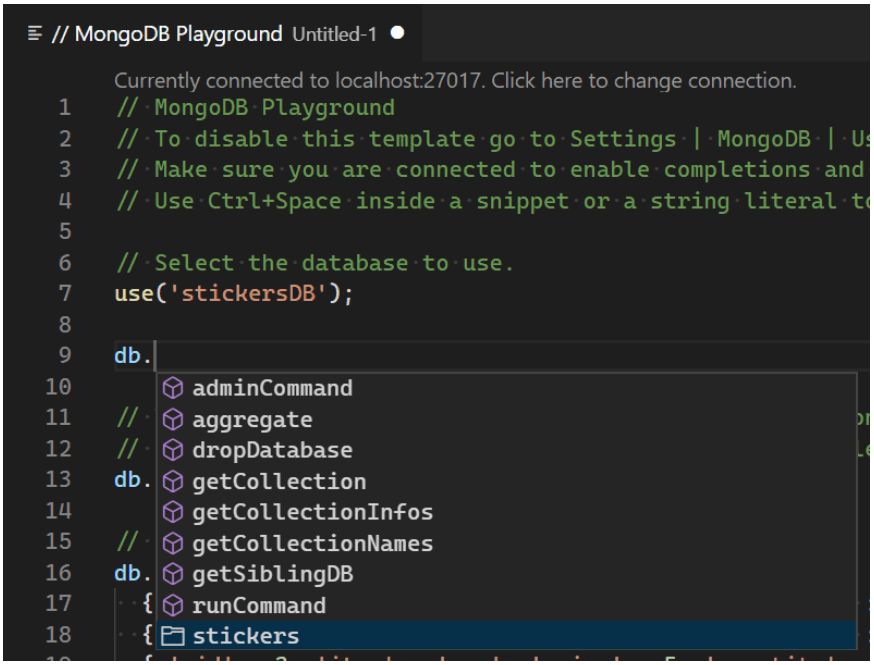

Figure 6: Mongodb

3.7. MERN Stack(Mongodb, ExpressJs, ReactJs, NodeJS):

MERN stack has become one of the technological stacks that are getting traction nowadays. The software platform is a collection of tools and frameworks used in application development. Those sets are properly chosen to collaborate in order to create well-functioning applications .

Other frequently used website development technology stacks include:

LAMP (Linux, Apache, MySQL, PHP)

MEAN (MongoDB, ExpressJS, AngularJS, NodeJS)

MERN (MongoDB, ExpressJS, ReactJS, NodeJS)

The MERN stack is a javascript-based technology stack that is intended to make the whole process go more smoothly. It speeds up and simplifies the implementations of full-stack web applications. MongoDB, Express, React, and Node.js are the four open-source components that make up MERN. These four essential technologies give an end-to-end environment for programmers to work in and play an important part in website development.

This is due to the fact that every piece of code in this structure is definitely written in JavaScript. JavaScript is a programming language that may be used to create client-side and server-side programs. Using this stack means simply using one language throughout all work levels, with no need to switch to another language in between. To utilize MERN Stack, your web developers just

need to be familiar with JS and JSON. In the event of a tech stack that includes numerous programming languages, they might need to first find out how to combine them before they can be used together. The fact that one programming language underpins the whole stack from top to bottom simplifies the web and mobile app development process. Besides that, as I mentioned before, React is a one-way data flow that is easier for programmers to program and debug. One of the primary benefits of adopting MERN Stack is the simplicity with which the app's code can be maintained.

CHAPTER 4: REQUIREMENT ANALYSIS:

4.1: User story.

With the development trends of current social networks in general as well as music in people's lives in particular. Then, users need to find a specialized social network to share their sounds as well as their feelings and thoughts through inspirational music and songs is indispensable to further upgrade the entertainment experience. as well as share your emotions with others. Besides, what users also want is a social network that ensures the accuracy, safety and health of information. The next problem is that some users today are content and video developers on a number of different social networks. They also want to find sources of audio tracks to use for their purposes. But they have difficulty in finding the right music and songs for their creative projects.

To clarify the issue more clearly, I will analyze this issue more carefully through the following user story. Besides me now, my friends and siblings all interact and use social networks such as Facebook, Tiktok and others. They share pictures, their emotional status on it. As well as viewing other people's posts to relax and entertain after a tiring day. But these things that they consider to be able to help them relax and entertain are mixed with many unhealthy things such as depraved photos and videos, false information and propaganda images (Drahošová, M. and Balco, P., 2021) (Etactics | Revenue Cycle Software. 2021). In addition, young people now even adults are now very interested in using tiktok to create good videos and clips, but audio people still do not seem to meet their current large demand. They want new sound sources that allow them to download or mix according to their needs. These sound sources are appreciated and used by many people. Then there are some people who like to have a private space to enjoy

music and read the lines shared by others. It can be a confide line, it can be a shared comment about this song or it can be a story that the other person wants to share with music that matches that emotion. Music has a variety of colors and genres, so there are all ages, genders and personalities of each person. It always holds a special affection for people in many ways. But all in a positive and healthy direction (Mitch de Klein . 2021).

It was because of the stories above what I saw that I came up with the idea for this project of mine. Why don't we create a social networking site focused solely around music. With the above needs of users and the benefits that music brings, this is completely attractive enough for me to create an audio social network.

4.2. Use case.

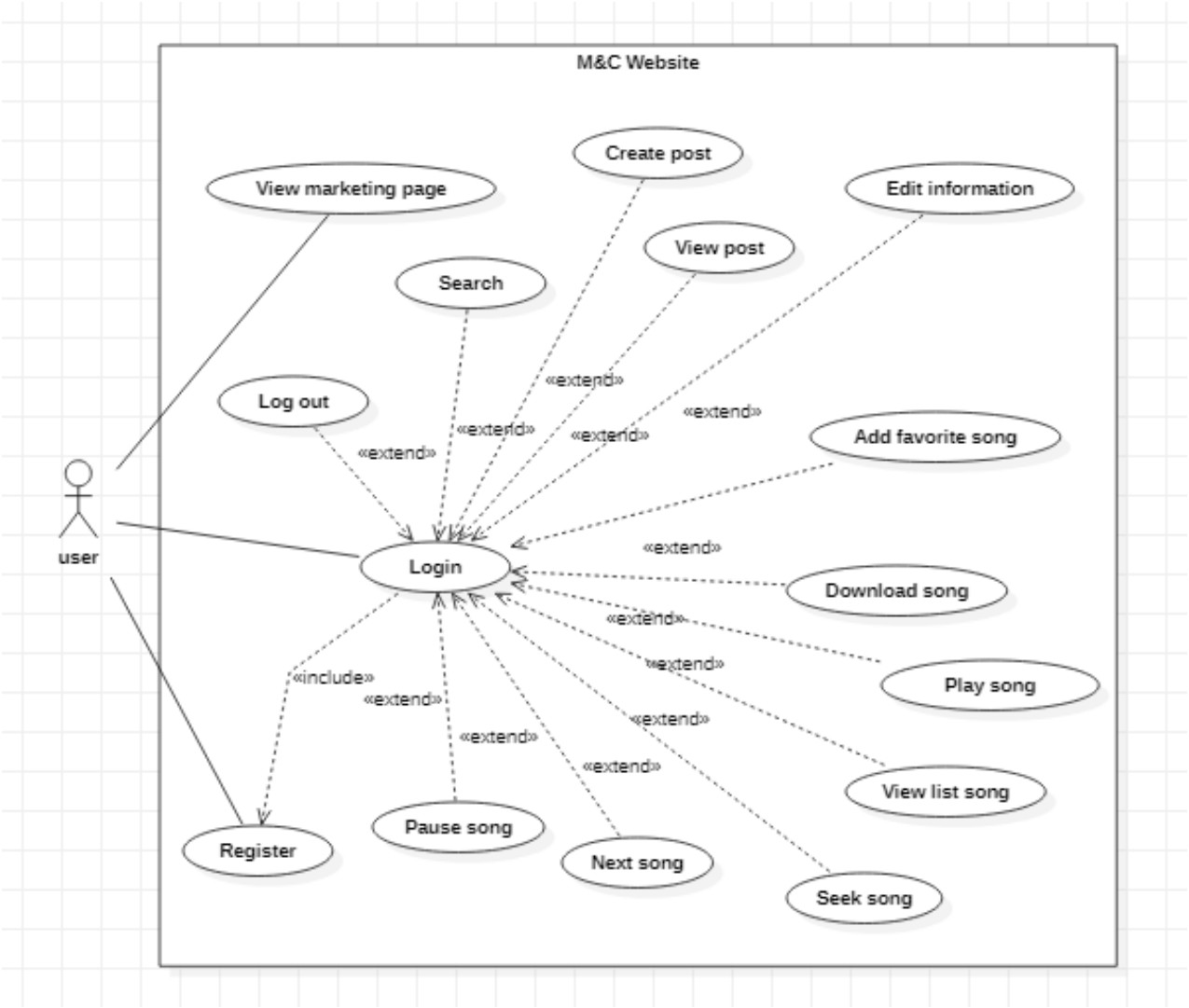

Figure 7: User use case

ID:	User
Title:	User system
Description:	Allows users to register an account, log in, create articles, view articles, edit personal information, play music, pause music, rewind music, skip songs, add music to favorites, download music, search

Primary Actor:	Any User
Main Success Scenario:	1. Users access the M&C website 2. If the user is using it for the first time, registration is required 3. After registration, the user can log in with the registered account. 4. The system will confirm the account and let the user access it. 5. After authentication, the user can access the next pages that the M&C website allows users to access. 6. Here users will be able to perform operations such as editing their information, posting articles, searching, playing music, stopping music, rewinding music, adding music to favorites, and downloading music. and cutting the music.
Frequency of Use:	6
Status:	Developing

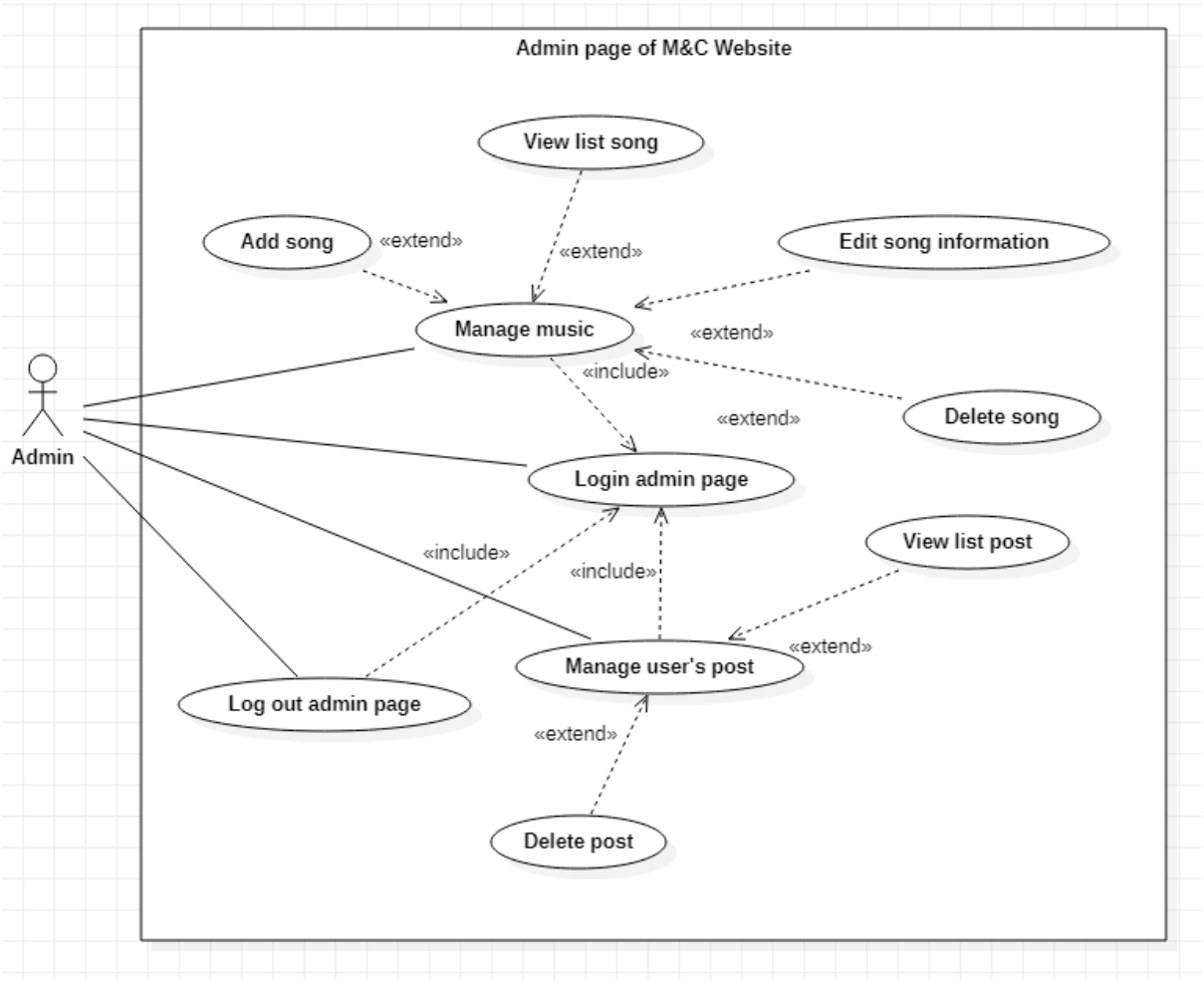

Figure 8: Admin use case

ID:	Admin
Title:	Admin system
Description:	Allow admin to login, monitor dashboard, manage music add, delete, edit, view. Manage user posts.
Primary Actor:	Admin

Main Success Scenario:	1. Admin accesses M&C website 2. Admin will log in with an account specifically assigned to admin. 3. The system will confirm the account and let the admin access the admin's homepage. 4. Here, the Admin will be able to perform actions related to music management such as adding, deleting, editing, viewing music, or managing articles such as viewing user posts, deleting that article.
Frequency of Use:	4
Status:	Developing

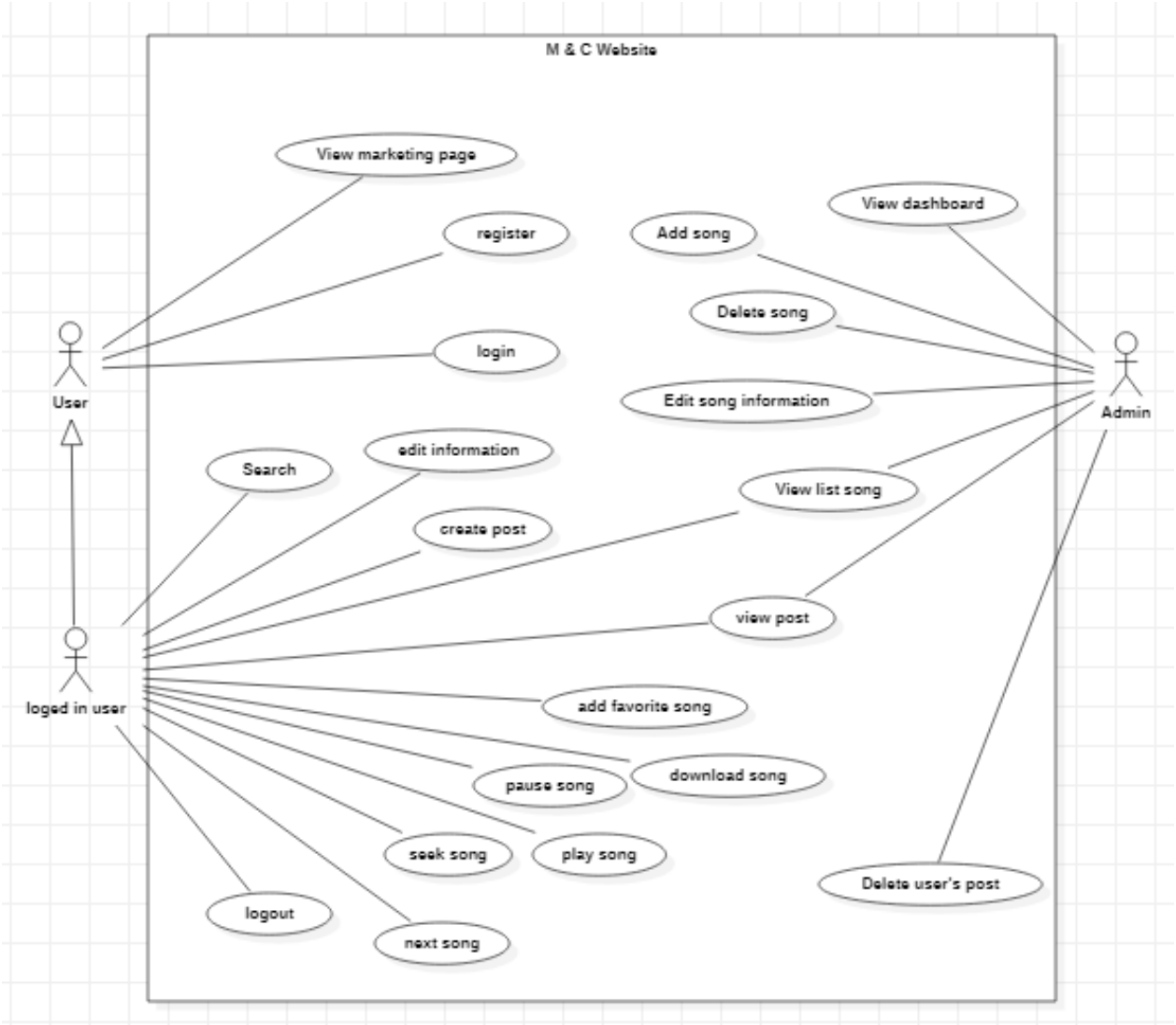

Figure 9: Use case

ID:	M & C Website
Title:	M & C Website
Description:	Allow the User to log in to the account and use the functions of the respective roles

Primary Actor:	User (Admin/ Logged in User)
Main Success Scenario:	1. Users access the M&C website 2. Users will be admitted to the marketing page 3. If you are using it for the first time, you must register first. 4. After having an account the user can log in 5. The system will verify the user's account is admin or not, if so, go to the admin page, if not, go to the page used for logged-in users. 6. Admin after successful login can perform operations to add, delete, edit, view music, or manage user's posts. 7. Successfully logged-in users can perform actions such as editing their information, posting articles, commenting on articles, searching, playing music, stopping music, rewinding music, adding music to favorites, downloading music, and cutting music.
Frequency of Use:	7
Status:	Developing

4.3. Sequence Diagrams:

4.3.1. Admin login:

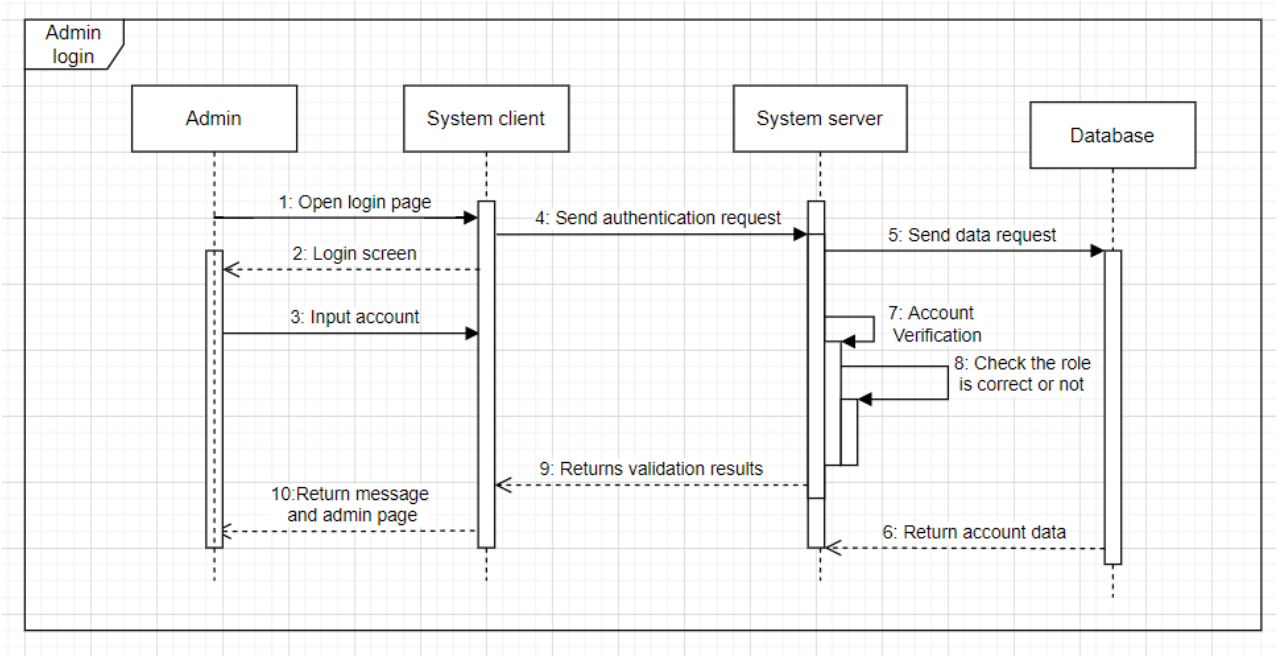

Figure 10: Admin login sequence diagram

The admin will first access the website's login page and then enter the email and password. Then the client will get that information and send it to the system's server. The server will receive information and get a list of users of the database. Next, the server will check if the login information matches or not. After the login information is authenticated, the system will check if this account's role is the admin role. If true, the Client will redirect to the admin page for the user.

4.3.2. Update user information:

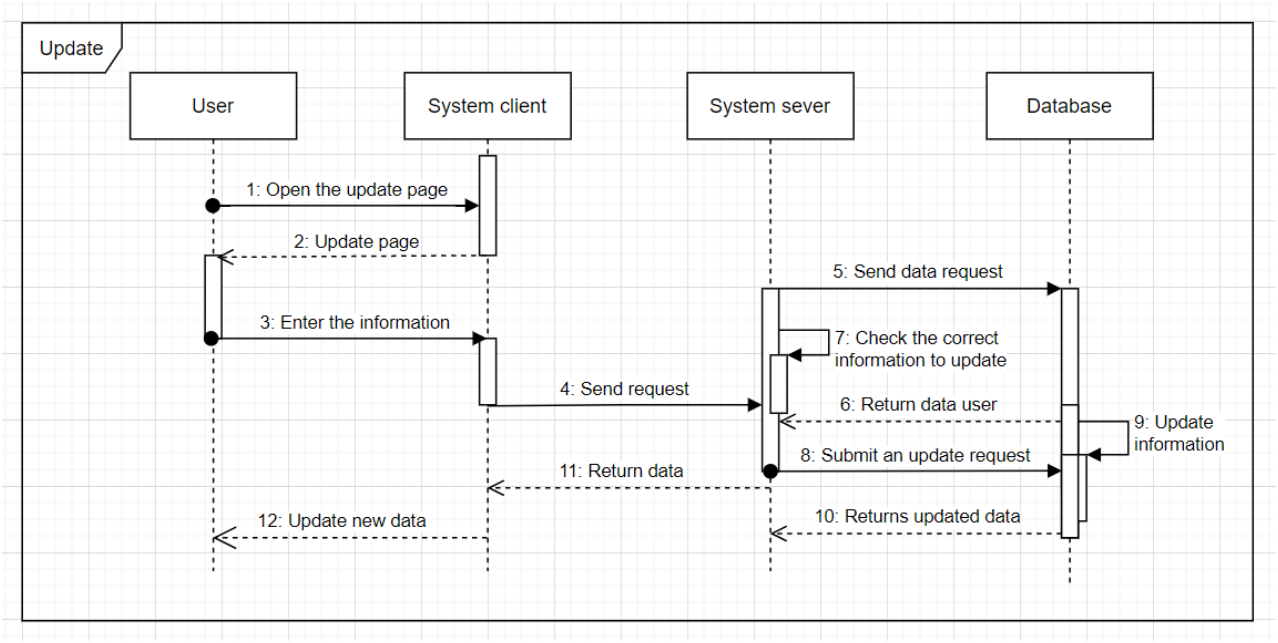

Figure 11: Update user information sequence diagram

The user will access the information editing page. The client will then return the information entry page to the user. After receiving the information entered from the user, the client will send the data back to the server. The server will check the user's id and then call the list of user information stored in the database. Next, the server will check the correct user id with the id in the information list. Then will update the information for the user with that id in the database. When the update is complete, the new information will be transmitted by the server and sent to the client. The client will return new information to the user.

4.3.3. Create post:

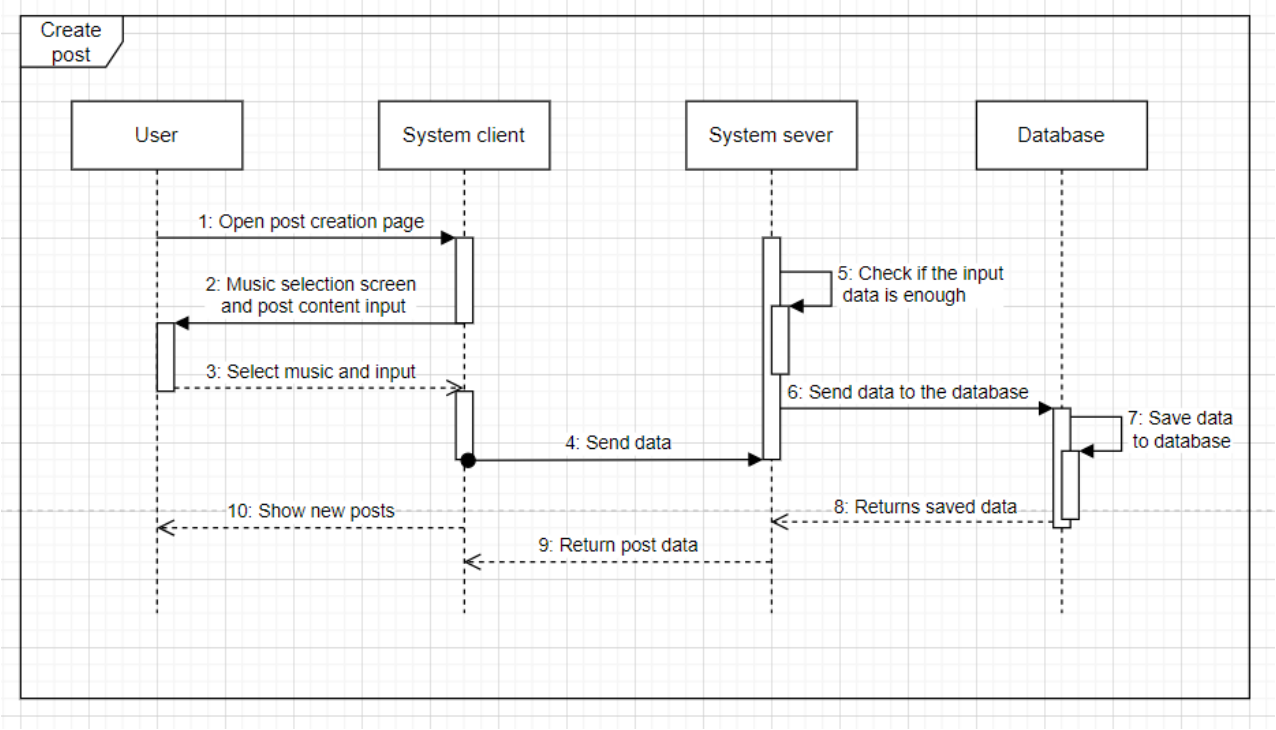

Figure 12: Create post diagram

The user will go to the post creation page. The client will display the post-creation page to the user. After entering the information to post. The client sends that data to the server. The server receives the information, then gets the poster id and the id of the song to post along with the test post in the user list and the playlist sent by the database. After checking, the server will import the poster data, content, and music data into the database. The database will store that data. The server will get the post data just saved in the database and then send it to the client. The client receives the information and displays the post to the user.

4.4. Class diagram:

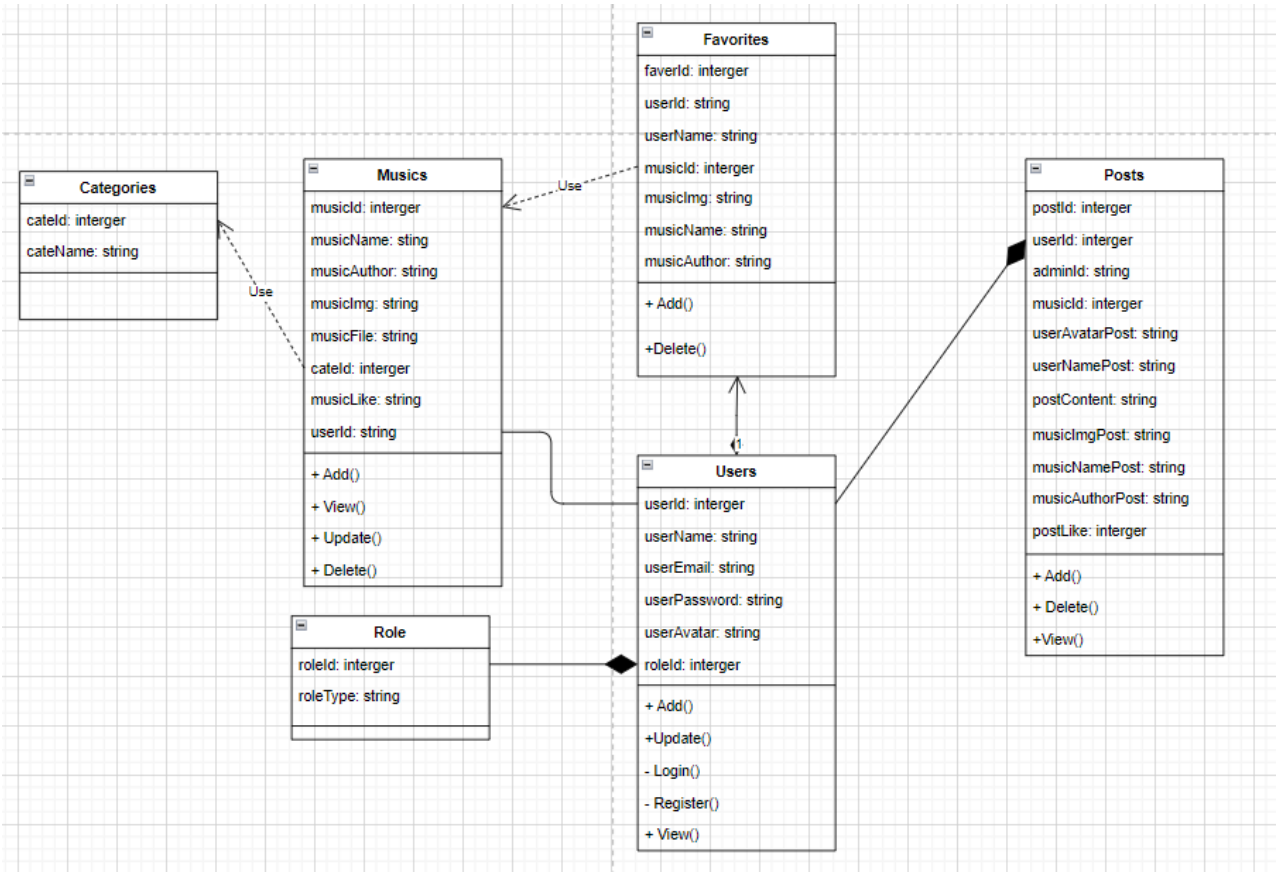

Figure 13: Class diagarm

Above is my class diagram representing the model representing the dark direction. The first is the role class, the role that is related to the user is the composition because when the role is lost, the user cannot exist independently because the user has to depend on the role to initialize. Next is the user class, the user class has a composition relationship with the post class, when the user class is lost, the post-class will also be lost. The user class also has a composition relationship with the favorite class, because the favorite class depends on the possible user class. The post class, the post class can exist depending on the user class when the user dies, the post will also die, the post class also has an agreement relationship with the music class because when the music is lost, the post can still exist independently. Next is the music class, the music class has an association relationship with the user class because music has the same relationship with the post and with the favorite as the user, the music class also has a dependent relationship on the class category because when the category changes, the music class also affects follow. Finally, the

favorite class has a dependency relationship with the music class because when the music is changed or lost, the favorite class will also be affected like the music class.

4.5. Activity diagram:

4.5.1. Login:

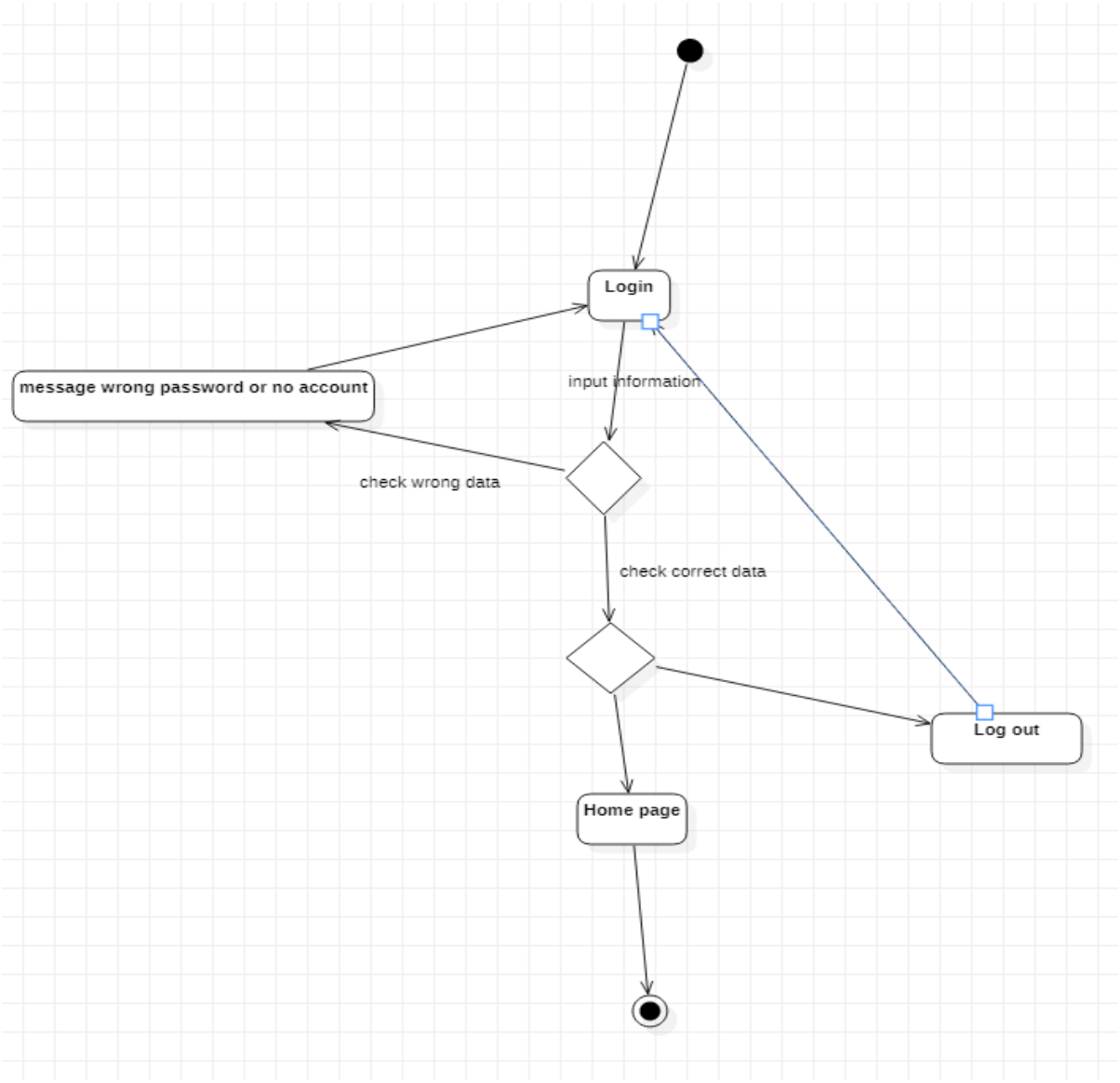

Figure 14: Login activity diagaram

4.5.2. Update information:

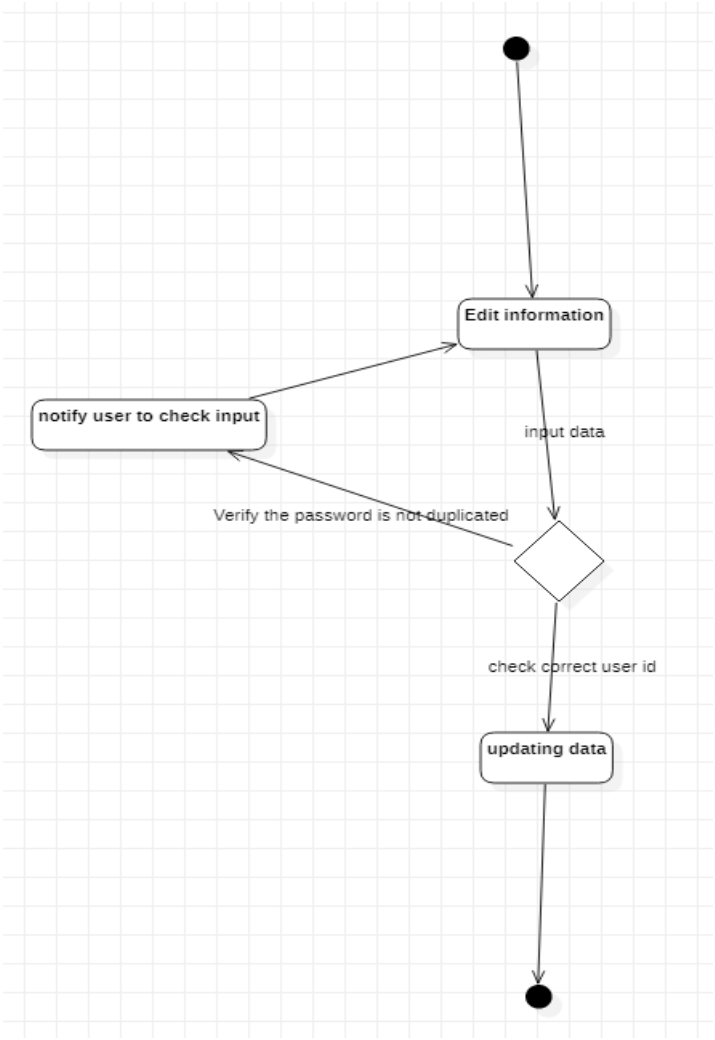

Figure 15: Update information

4.4. ERD:

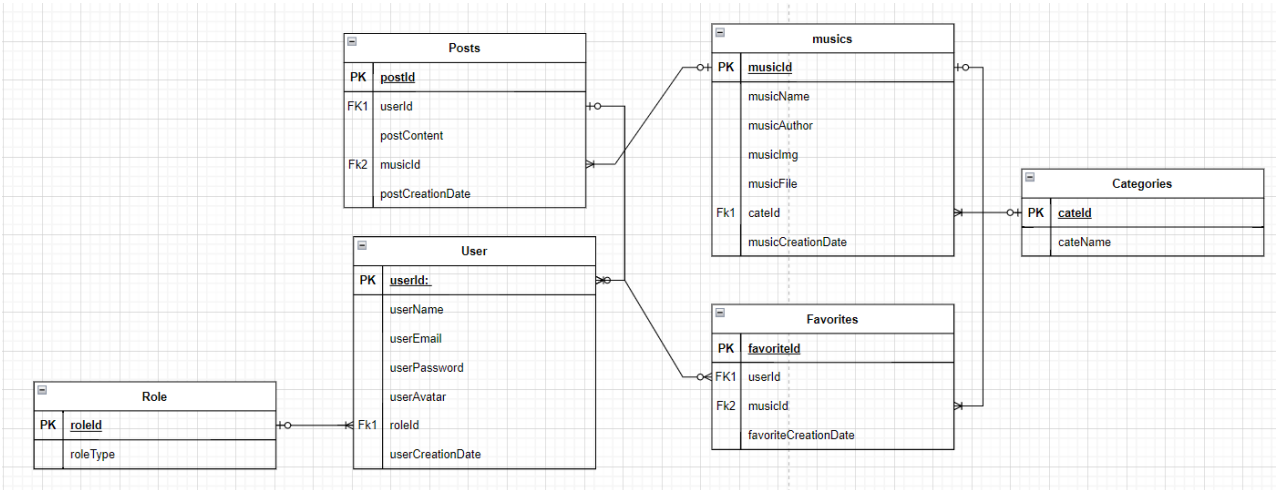

Figure 16: ERD diagram

This is my ERD. I draw on what I have learned and researched through the requirements needed for my website. First I have a role table, the roles table has a one-to-many relationship with the user table. Because a user can only have one role, but a role can have many main users, so I use a one-to-many relationship. Next is the post table, a post can only have one user and one music, but a user can have many posts and music can have many posts. That's why I use a one-to-many relationship between posts and users and music. Next is the favorites table, a favorite can only be for one user and one music, but a user can have many favorites and music can be in many favorites. Next is the music table, music can have many posts and many favorites, but a post can only have one music and a favorite can only have one music. Therefore, the relationship between music and posts and favorites is one-to-many. Finally, in the category table, a category can have much music and music can only have one category, so the relationship between category and music is one.

4.5. Sitemap:

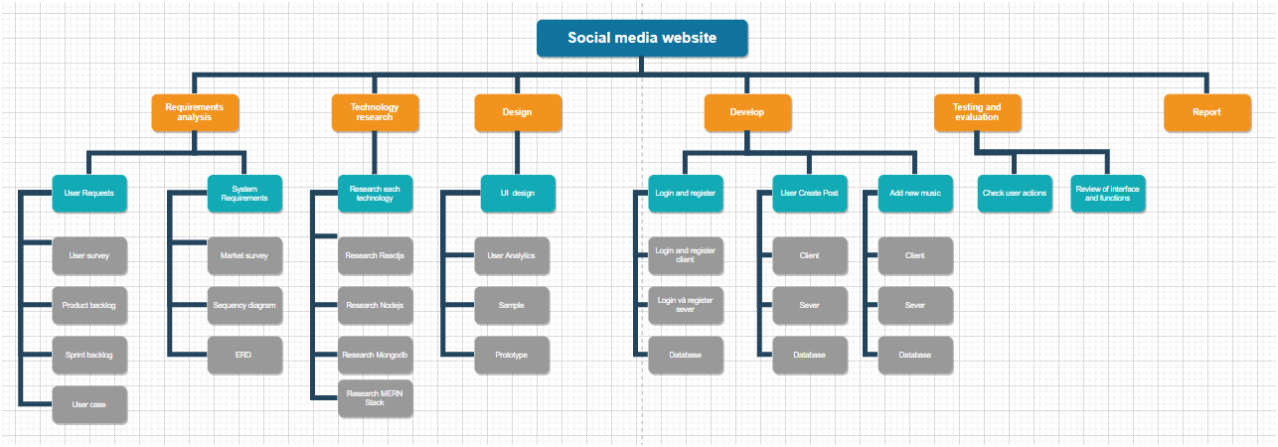

Figure 17: Sitemap

CHAPTER 5: Review of Software Development Methodologies

5.1. Waterfall:

In the software development business, the waterfall technique is one of the most well-known approaches. It has a lengthy history of assisting several successful enterprises. Before you can start implementing this function, you'll need a lot of documentation and structural requirements. It's broken down into closed stages. The following step will begin after the

preceding phase has been finished. The stages are rather rigorous, and they normally go like this: identify the project's requirements and scope, analyze those needs, design, develop, test, deploy, and then maintain. This strategy is rigid, which implies that what the client and developer determine from the start must be considered. The Waterfall technique frequently necessitates a complete reboot if any updates or late-stage faults must be addressed.

5.2. Agile:

As the waterfall technique eventually fails to satisfy demand, the task becomes more difficult. The agile technique looked to be an excellent substitute for the waterfall approach. Because there is constant connection with the client throughout the development process, this technique is extremely adaptable in project development. As a result, this approach may be readily modified to meet the needs of the user. Unlike Waterfall, Agile is well-equipped to deal with the complexities and unpredictability that development projects entail. Teams work towards the aim of producing functioning software during the sprint (or some other testable, tangible output).

5.3. Scrum:

Scrum is another technique that is anticipated to be built on the Agile core. This strategy is based on Agile, but it adds the necessity that developers work closely together on a regular basis. Scrum is a method of project management that is built on an iterative approach with the team at the center. Each team will include experienced members who will oversee each phase and have frequent meetings in accordance with the project's cycle. This necessitates a high level of organization and discipline on the part of each team member. To create software and introduce it on a frequent basis, team members break down end objectives into smaller goals at the start and execute them using fixed-length repeats — or sprints (usually lasts two weeks). Meetings are a crucial part of the Scrum process, and daily planning and demonstration meetings are held during each sprint to track progress and get feedback. This method encourages quick change and progress while also adding value to difficult projects. Scrum is a software development methodology that blends the structure and discipline of classic software development approaches with the flexibility and iterative practices of current Agile.

5.4. Spiral:

The spiral model is one of the most prominent Software Development Life Cycle models for risk management. It resembles a spiral with several loops in diagrammatic depiction. The spiral's precise number of loops is unclear, and it varies from project to project. A Phase of the software development process is defined as each loop of the spiral. Depending on the project risks, the project manager might change the number of phases required to build the product. The project manager plays an essential role in developing a product utilizing the spiral model since the number of stages is constantly determined by the project manager.

5.5. The method applied in the article:

Of the above methods, I choose the Scrum method for my system. Because as mentioned above Scrum is suitable for projects that need flexibility and speed. For my project, I need agility and discipline to complete the project. Besides, the flexibility of the project is also an important point. Because sometimes in the process of working on my project, many different cases and problems may arise, so flexibility is required to complete the project well.
I will complete each requirement outlined in each sprint backlog and will repeat the testing and evaluation at the end of each week. That way, I will be able to fix the maximum errors that occur during the project build process.

Product backlog.

Below is a list of the requirements that need to be fulfilled in this project and will be sorted by priority from start to finish. The requirements at the top will need to be asked first and must be.

- Sign in to register an account.
- Verify the user is admin or user.
- Create articles, publish articles.
- Play music, rewind music, next music, download music.
- Add music to favorites.
- Users change personal information by themselves.

- The profile page only displays the posts of the logged-in user.
- Admin can add music, delete music, edit music information.
- Admin can delete user's posts if necessary.
- Admin and user can log out.
- Search.

CHAPTER 6: Design:

6.1. Wireframe:

First, I'll use wireframe to create a skeleton for my project's UI. A wireframe (or skeleton structure) is a tool for designing the user interface and completing the website's structure. Or A wireframe is a rough draft of a website before it is designed. Although many aspects are required to develop and design a website, employing a wireframe is an essential stage in the process. This tool focuses on the website's structure.

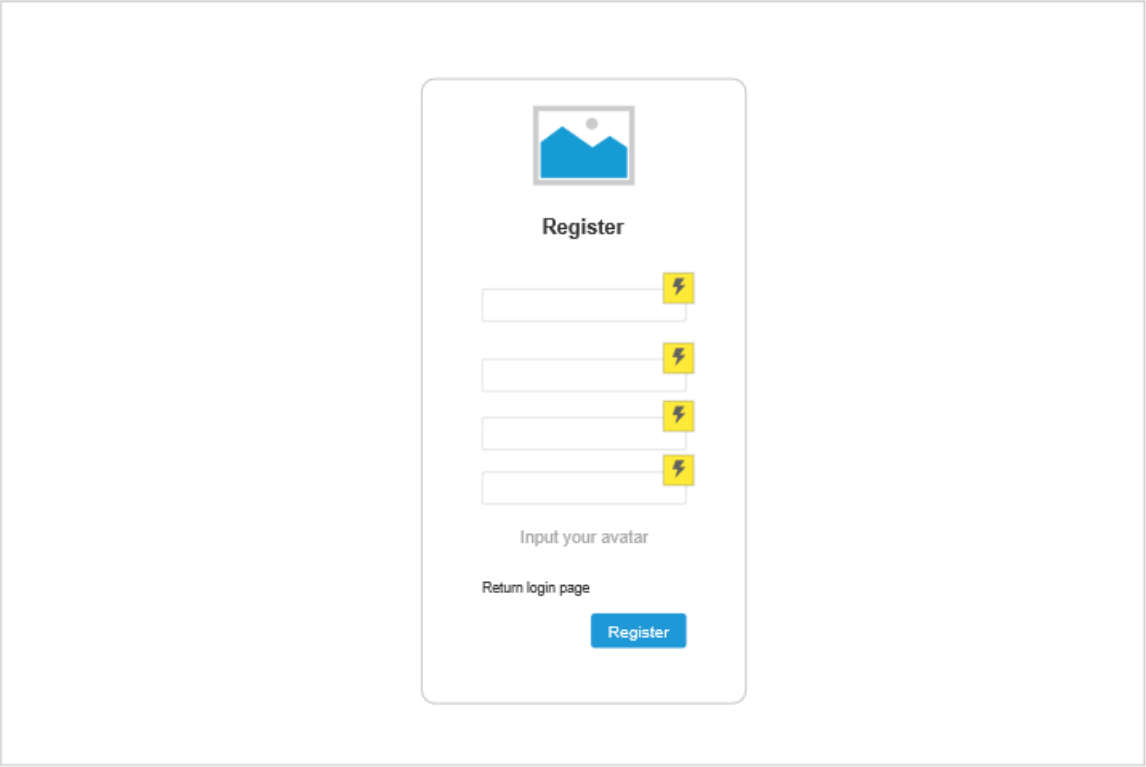

Figure 18: Register page wireframe

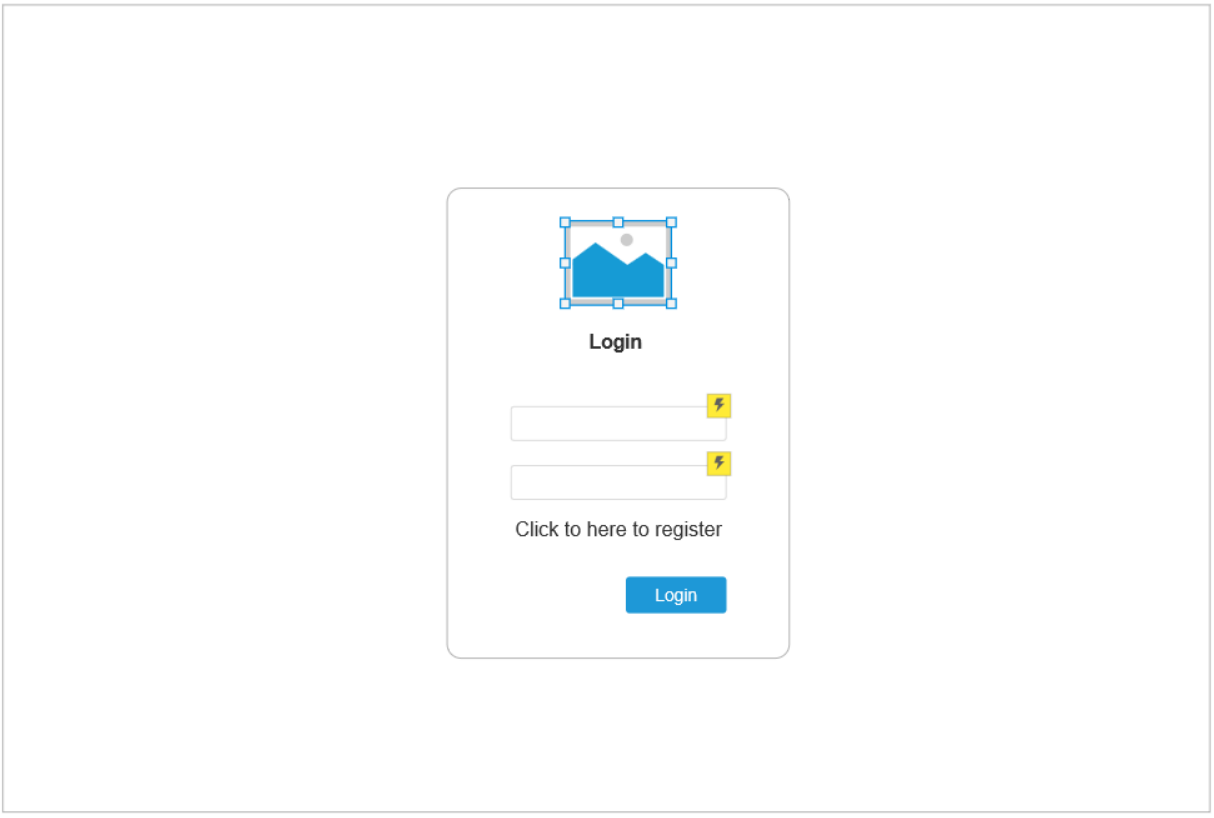

Figure 19: Login page wireframe

Figure 20: Marketing page wireframe

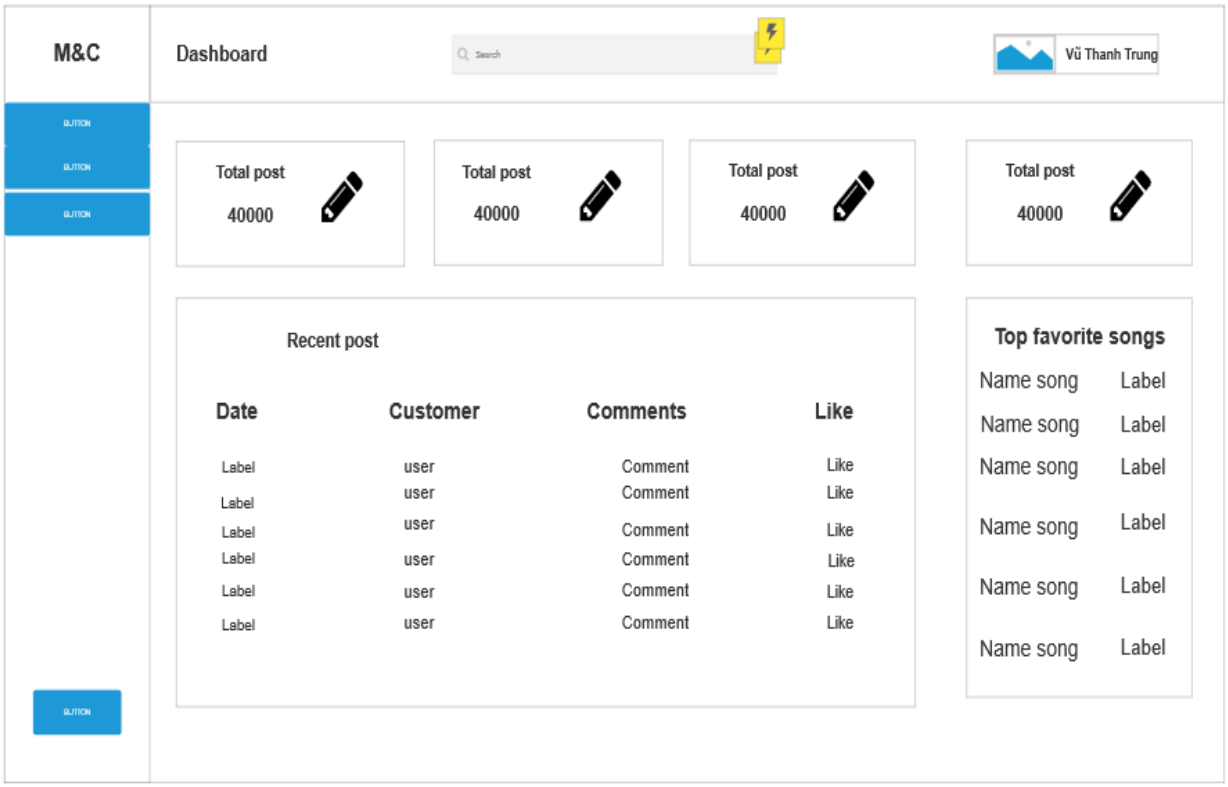

Figure 21: Admin dashboard wireframe

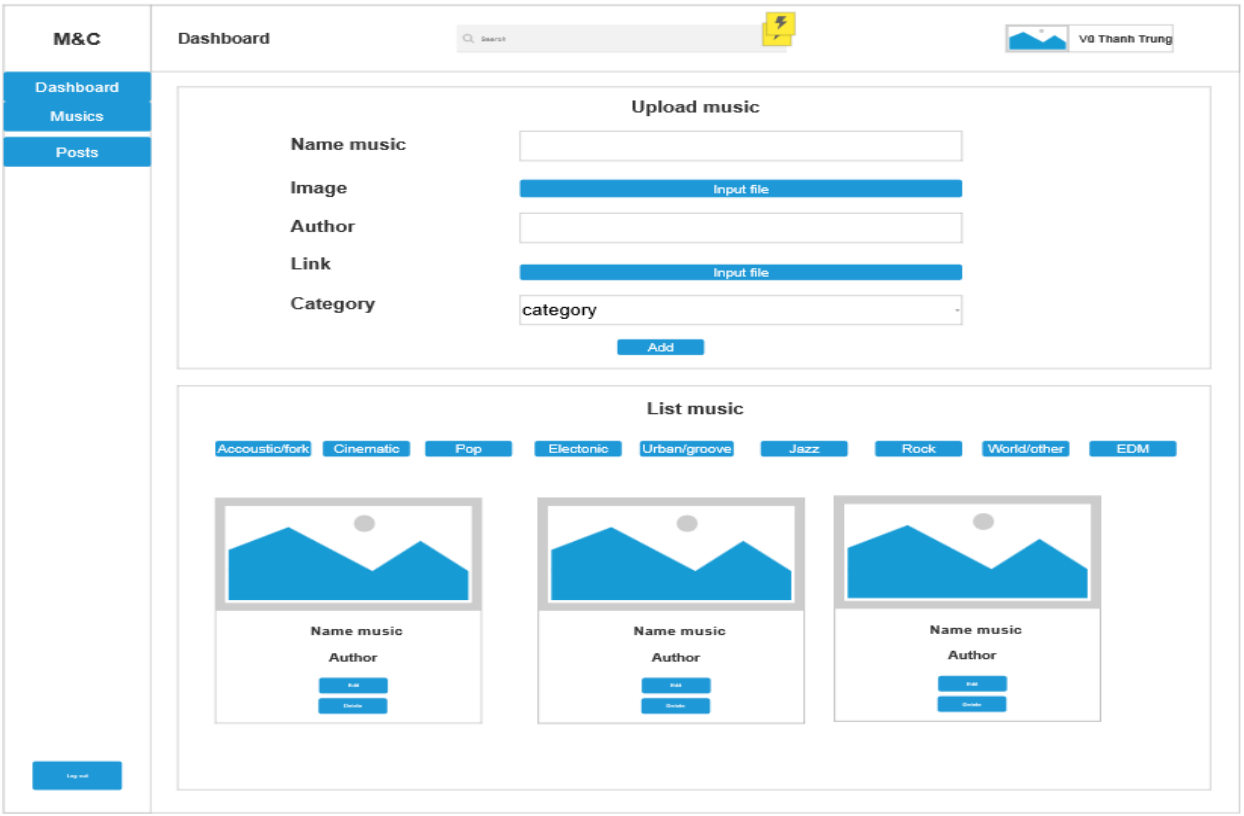

Figure 22: Admin music page wireframe

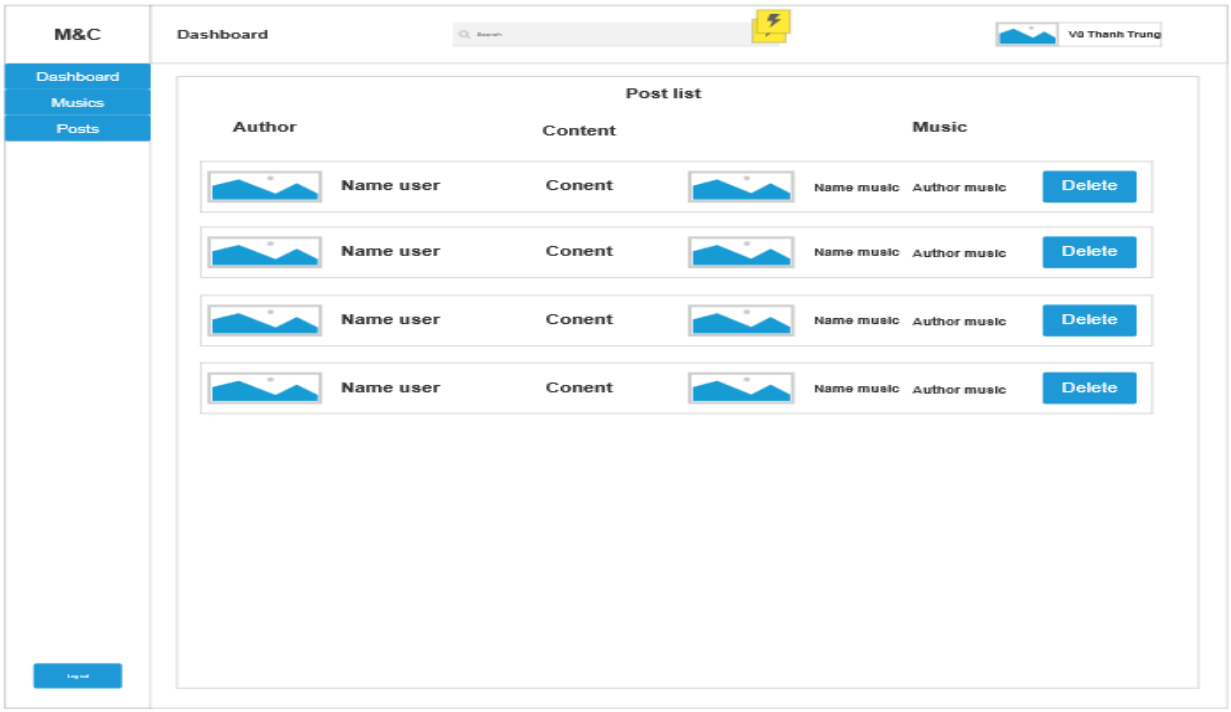

Figure 23: Admin post page wireframe

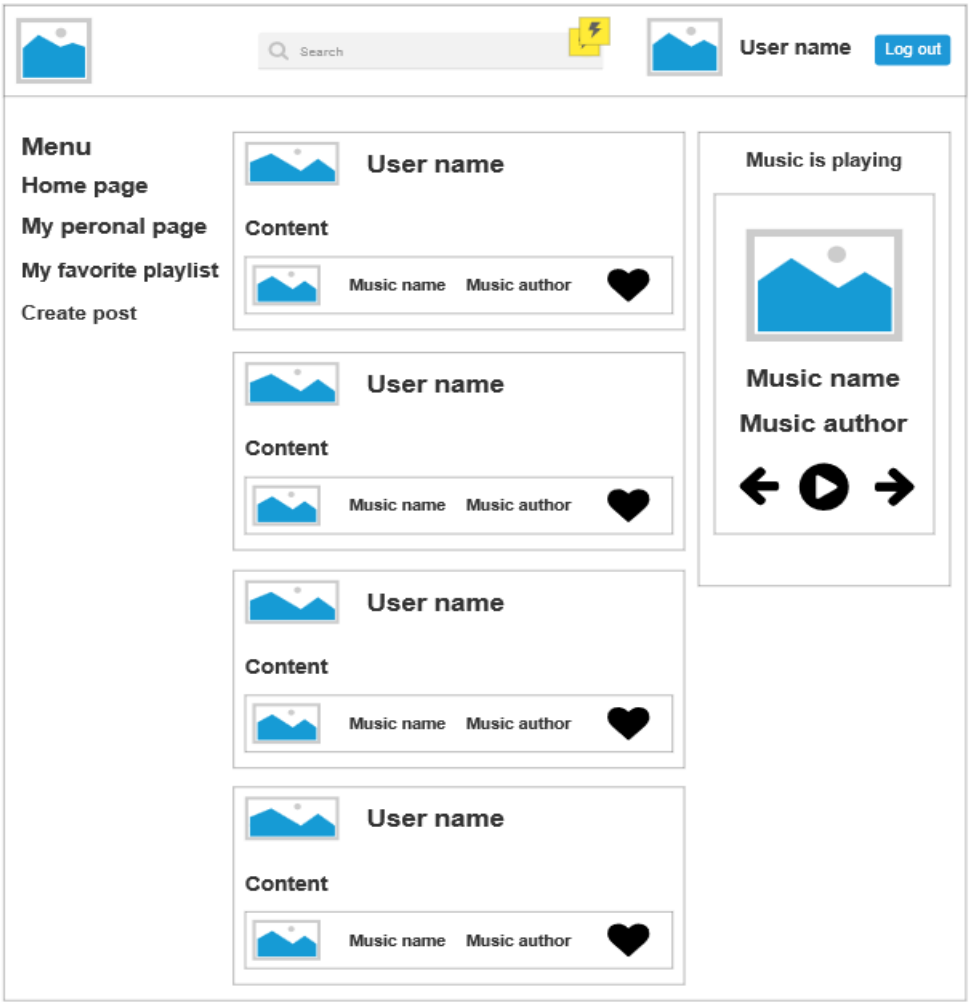

Figure 24: Home page wireframe

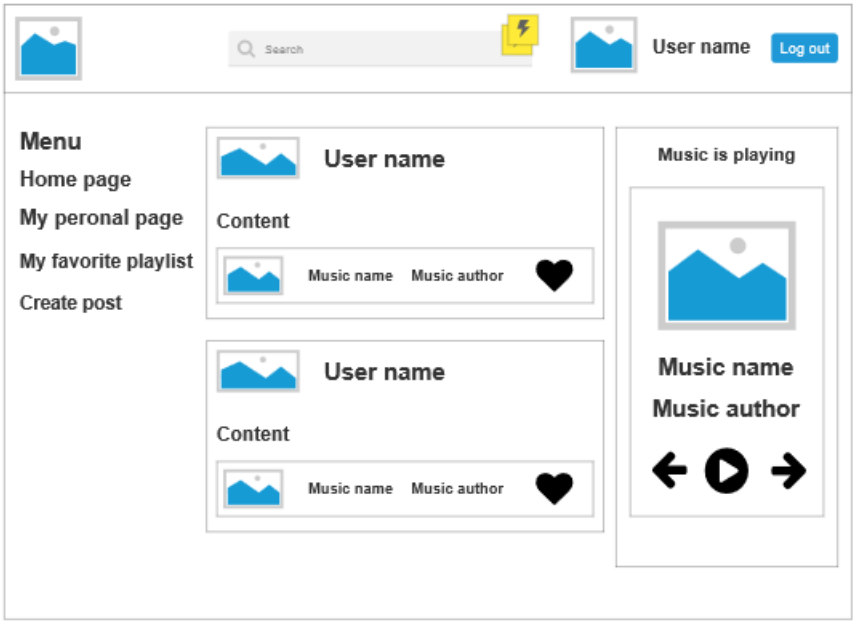

Figure 25: My personal page wireframe

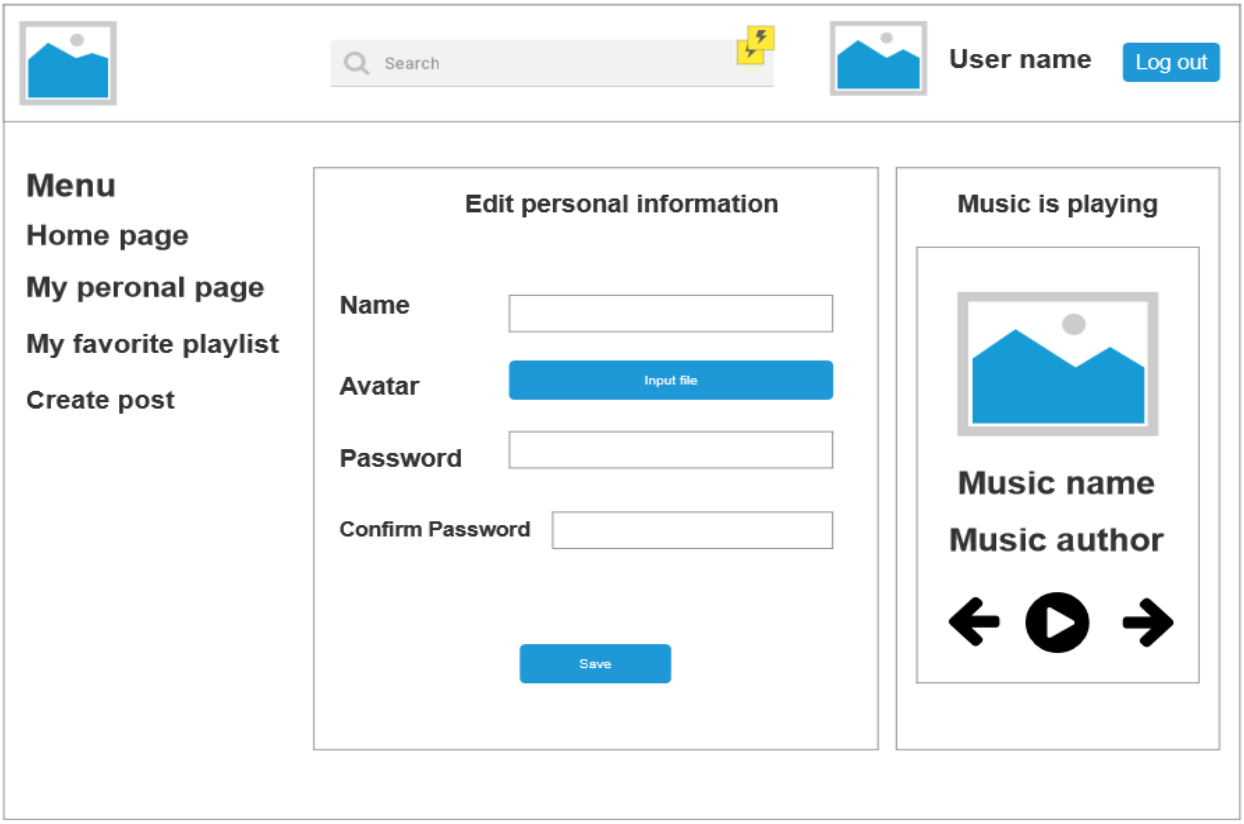

Figure 26: Edit personal page wireframe

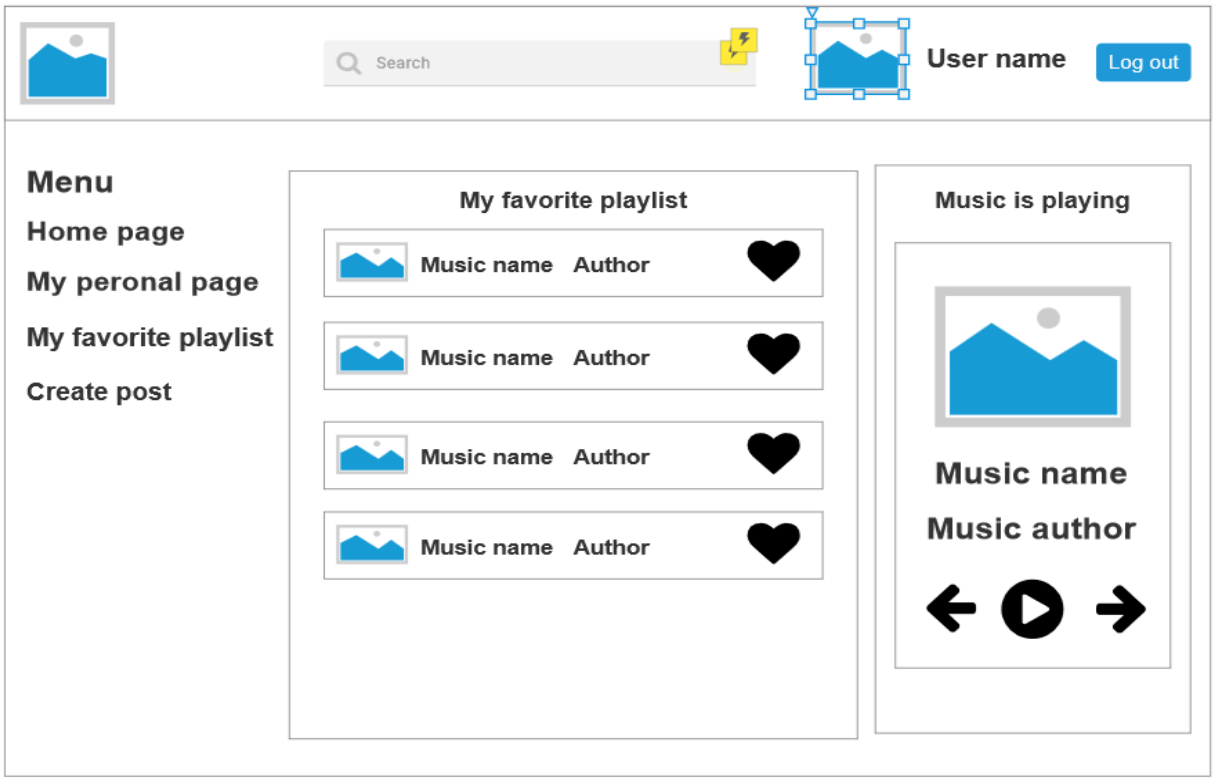

Figure 27: My favorite playlist wireframe

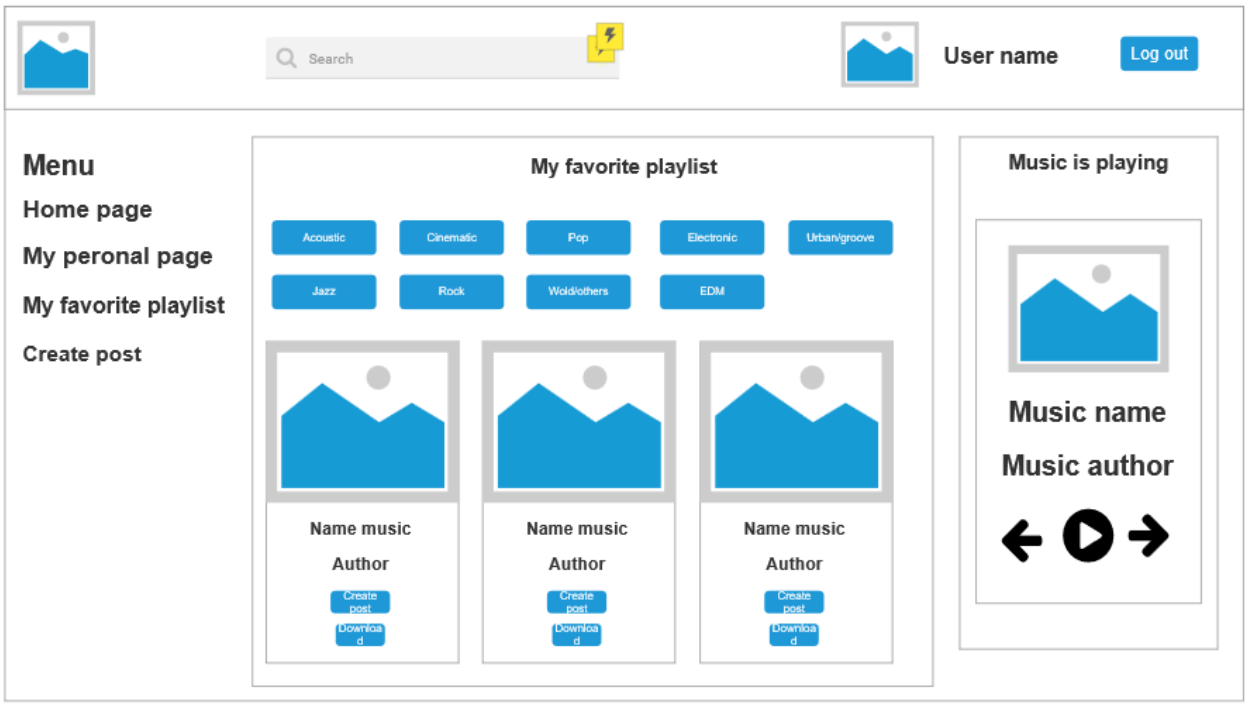

Figure 28: Create post wireframe

6.2. UI design details:

After having the wireframe design for the project, I began to edit and add more details to make the design more complete. This is the most detailed blueprint of my website.

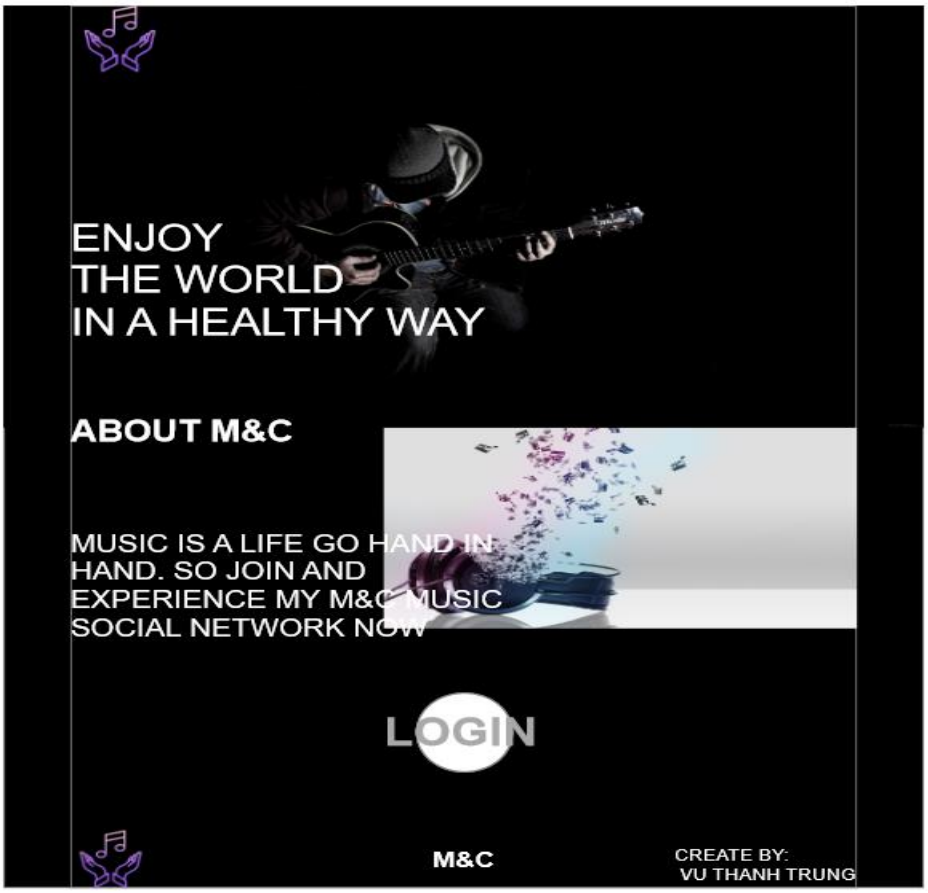

Figure 29 Marketing page design

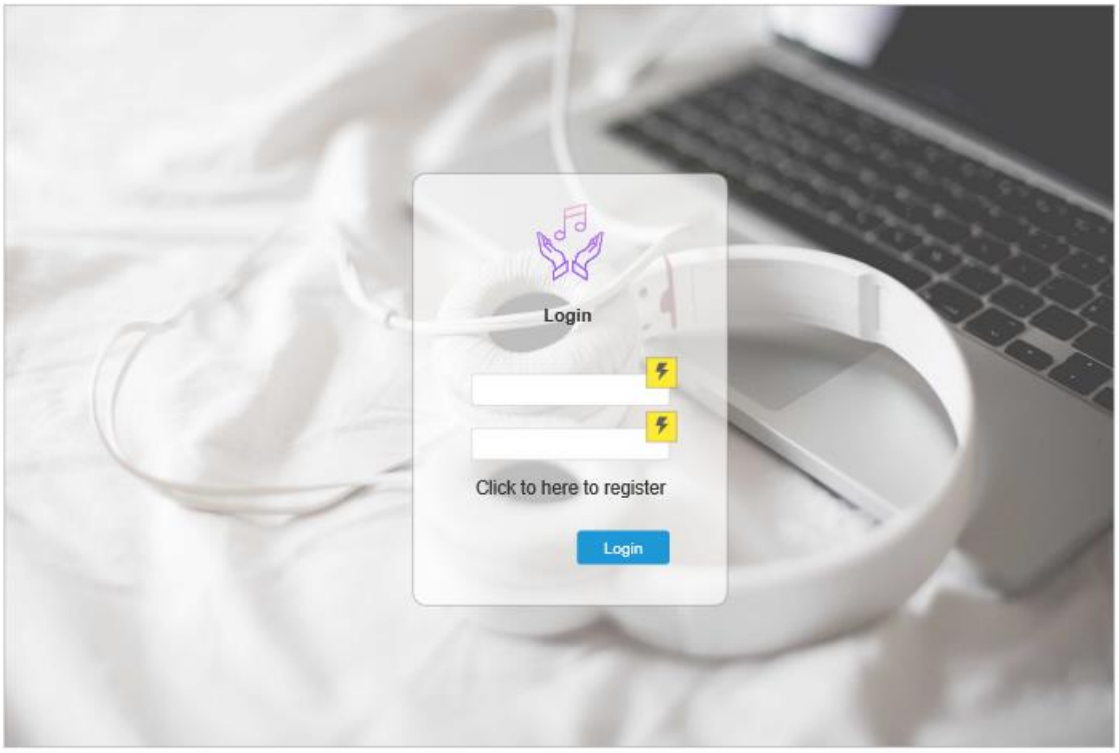

Figure 30: Login page design

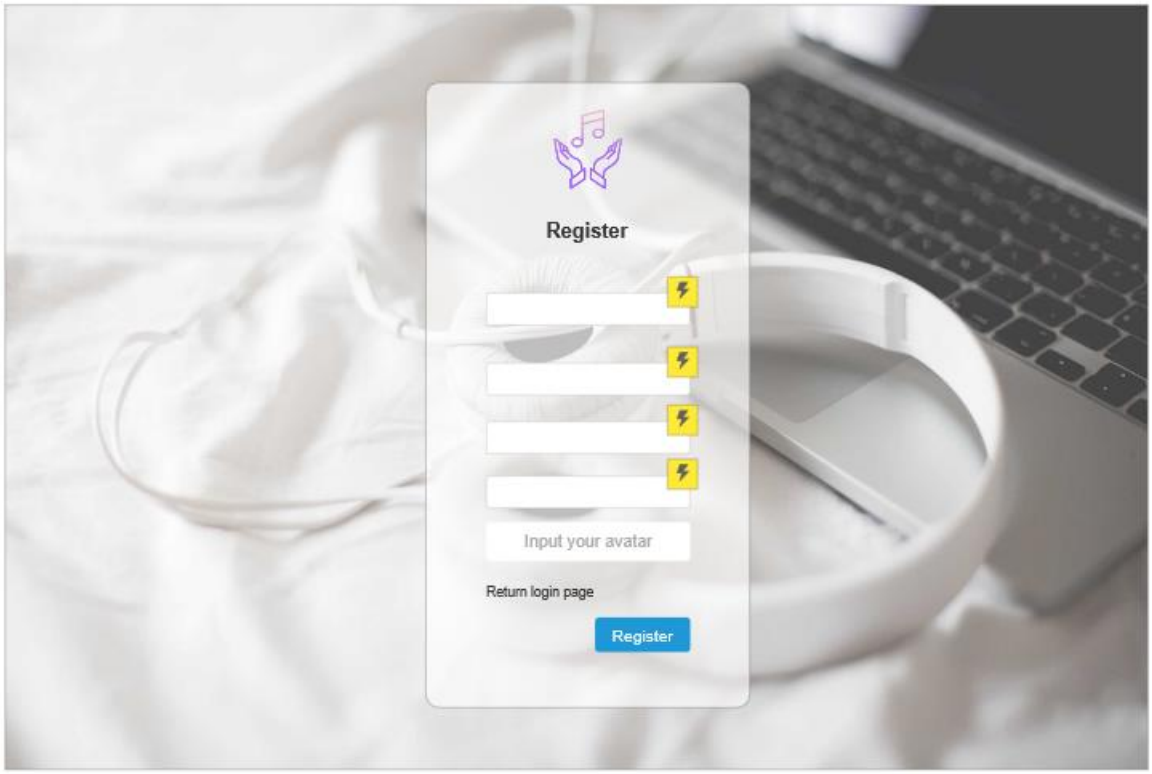

Figure 31: Register page design

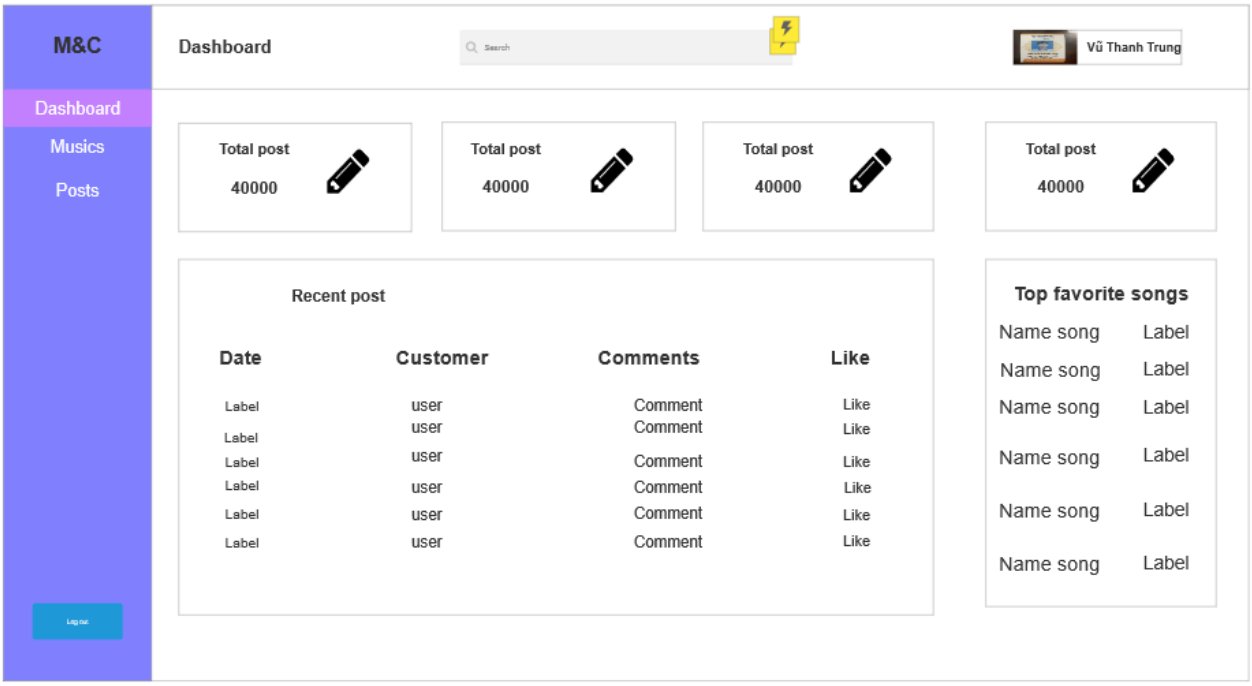

Figure 32: Admin dashboard page design

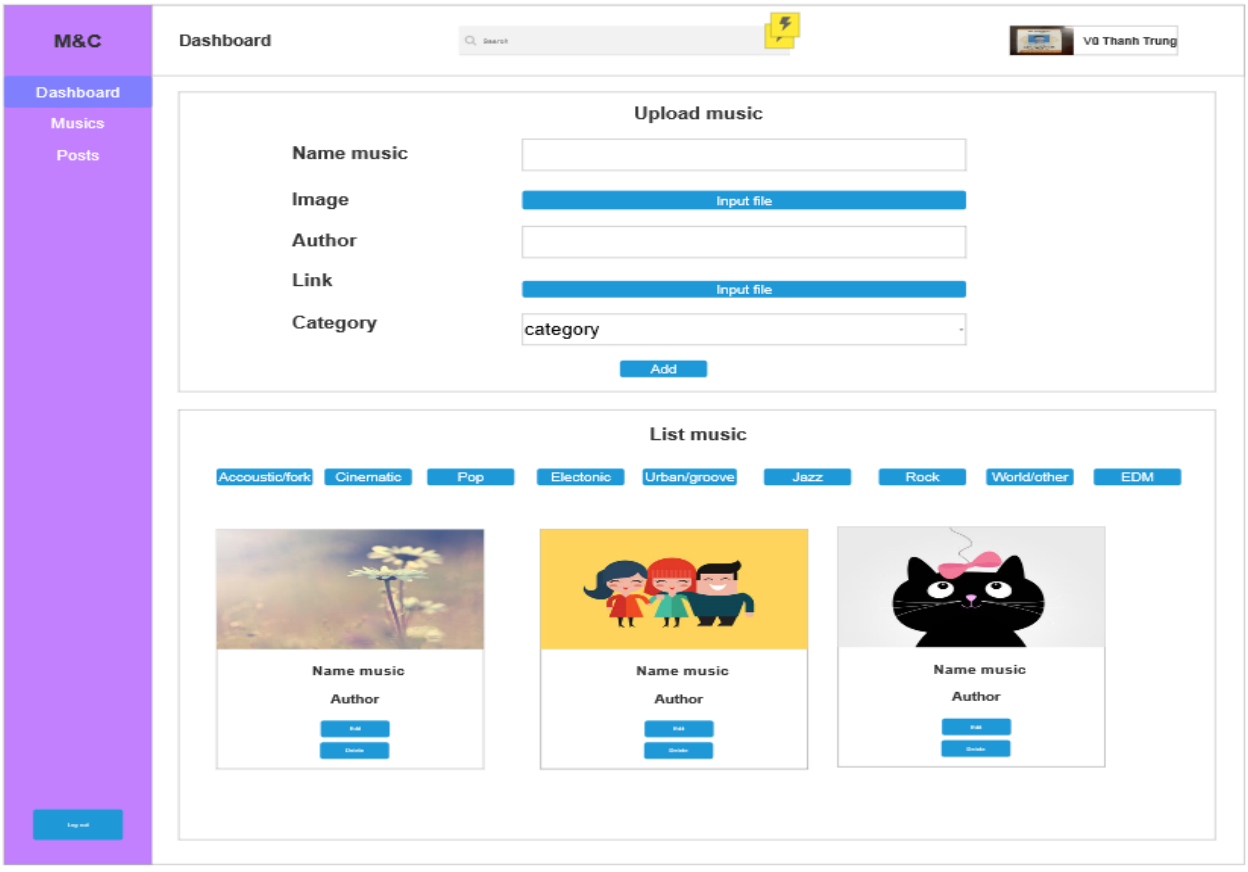

Figure 33: Admin music page design

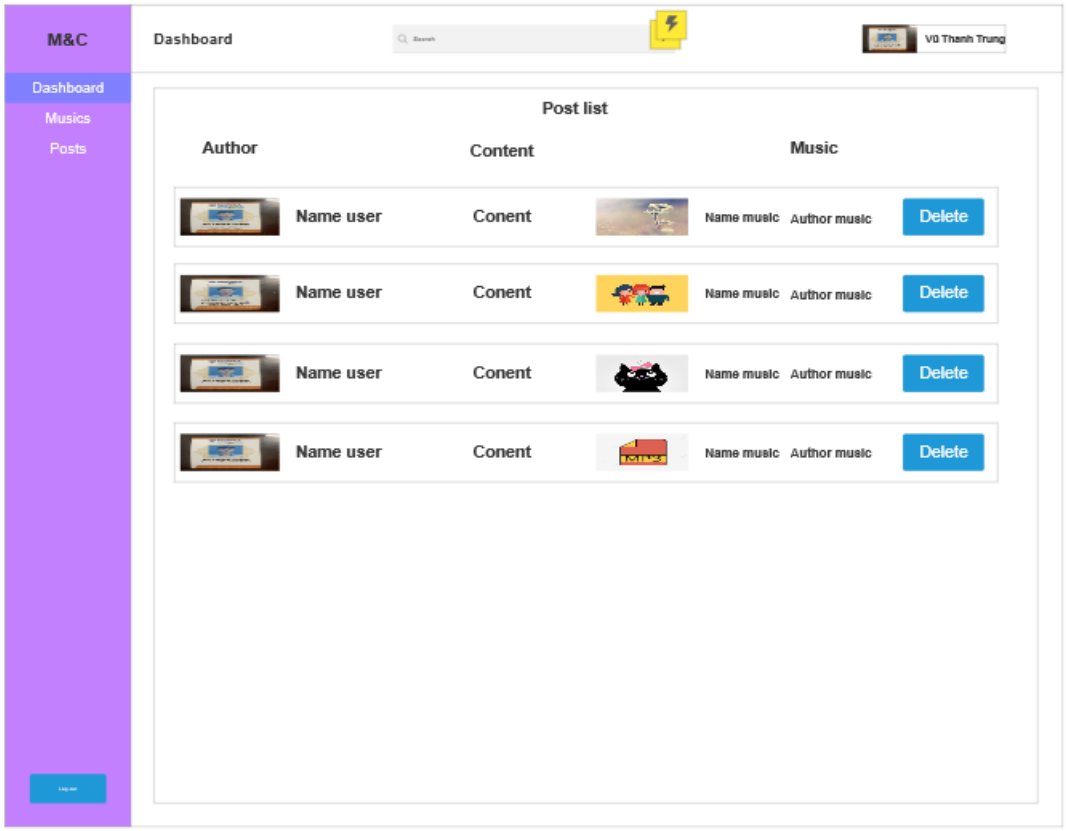

Figure 34: Admin post page design

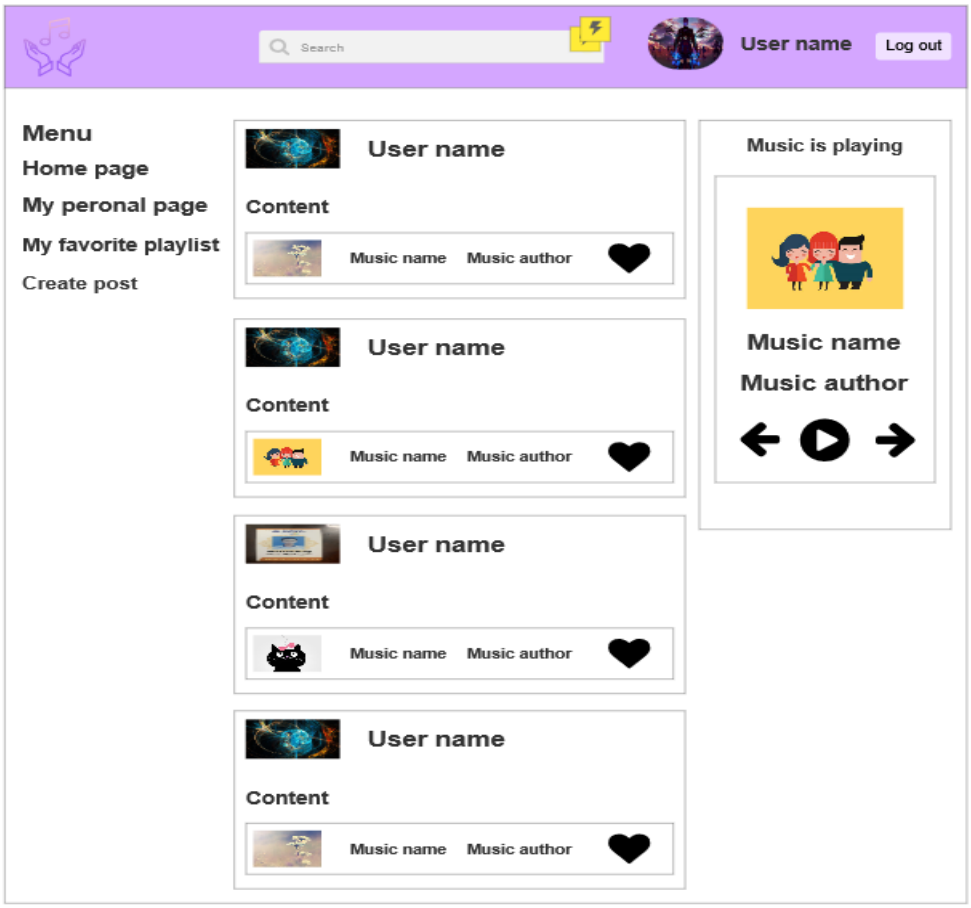

Figure 35: Home page design

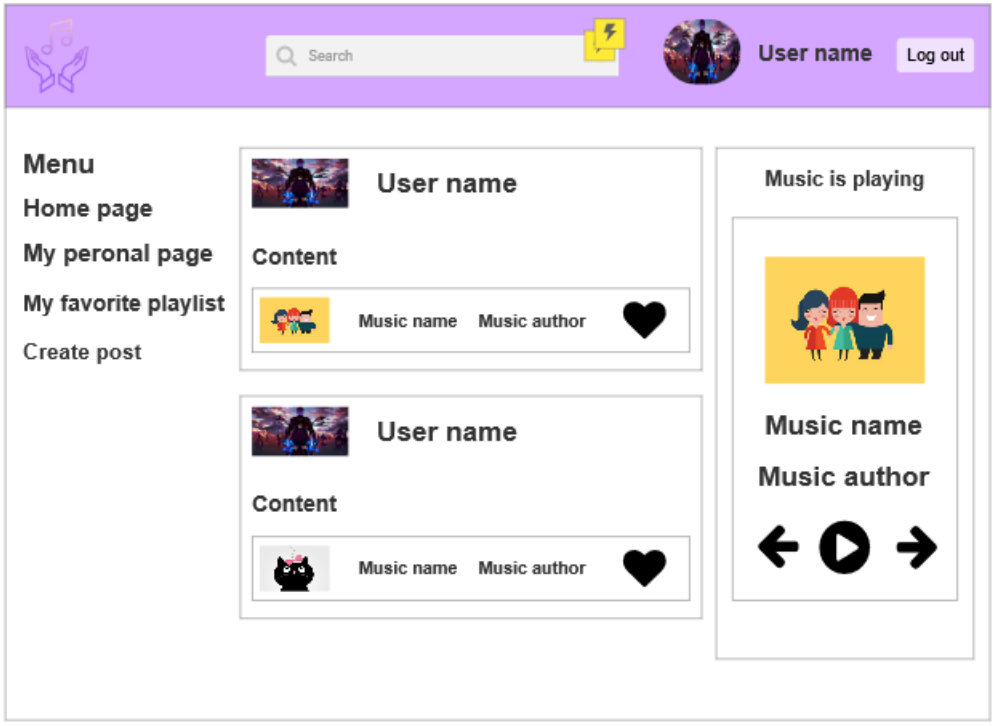

Figure 36: My personal page design

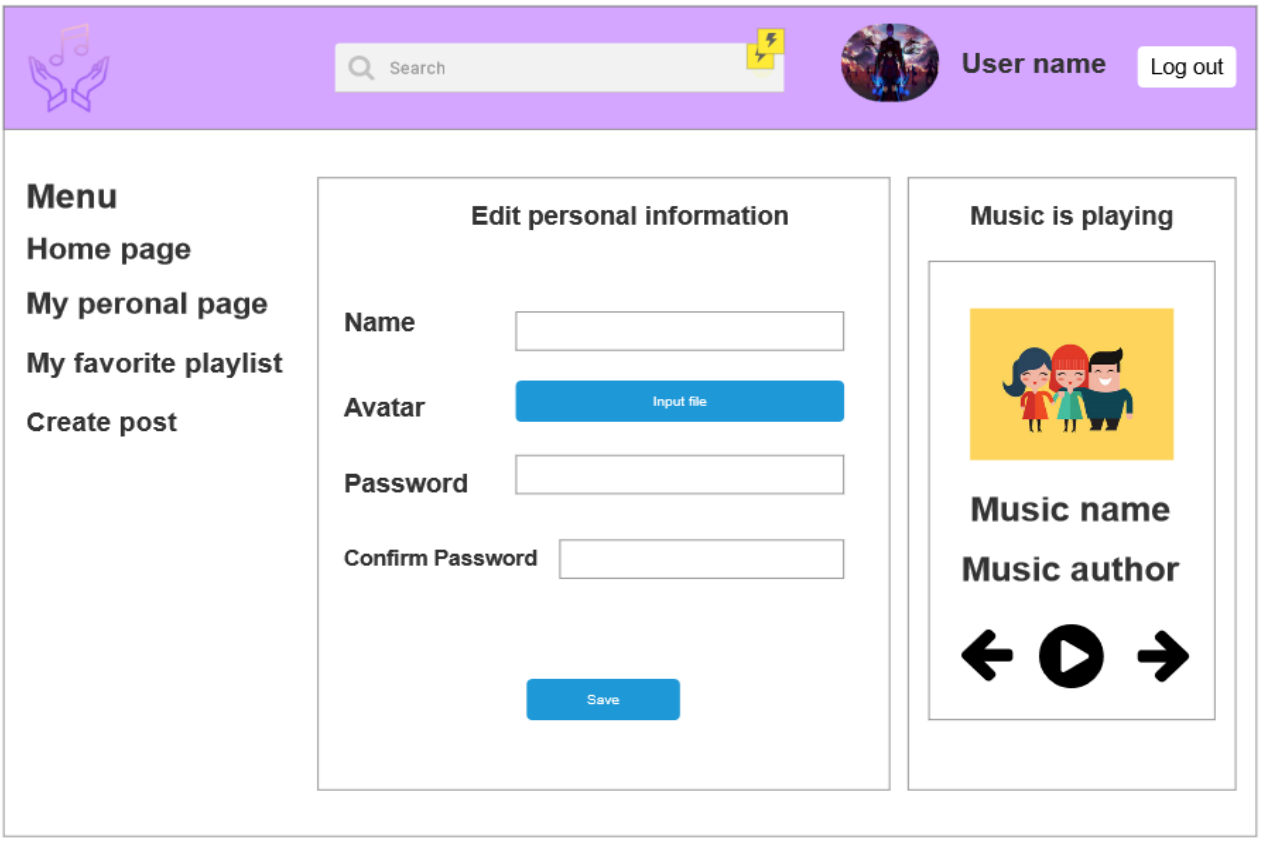

Figure 37: Edit personal page design

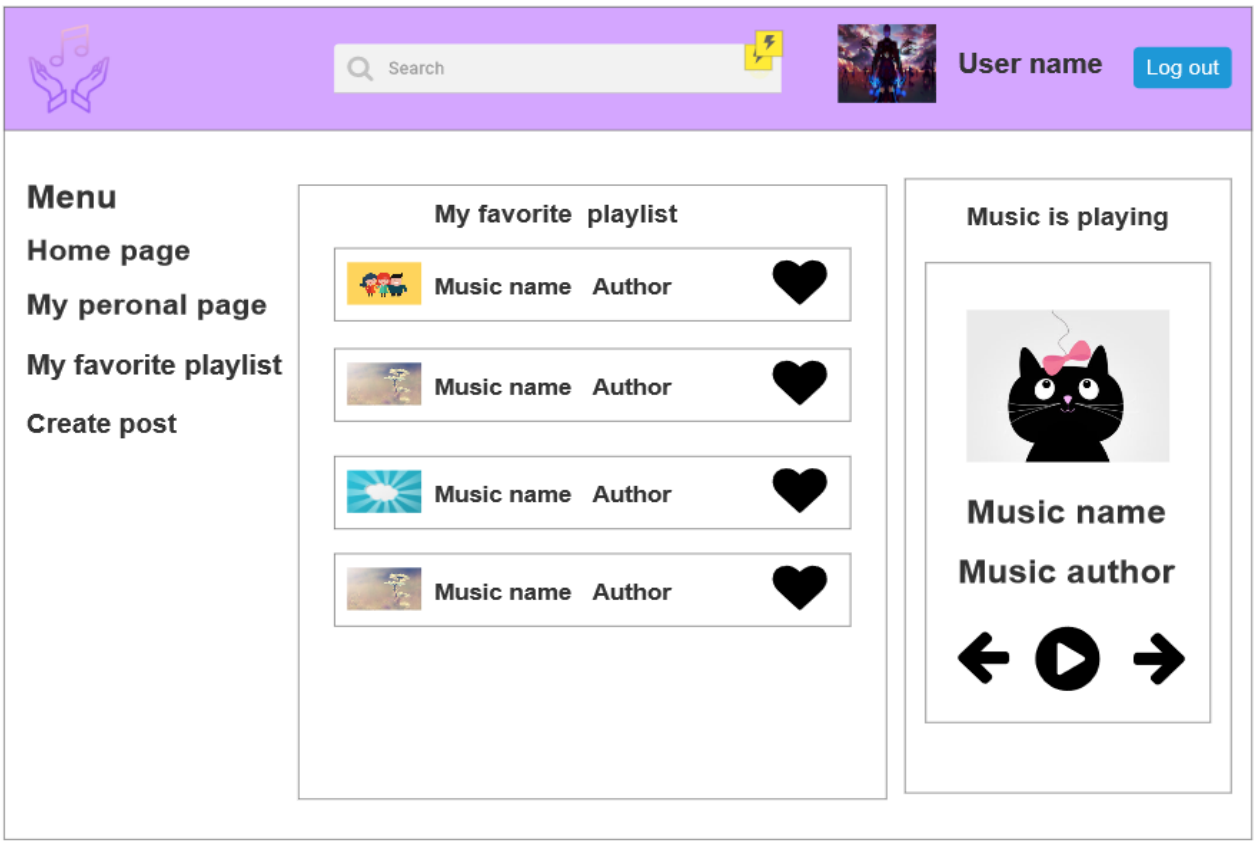

Figure 38: My favorite page design

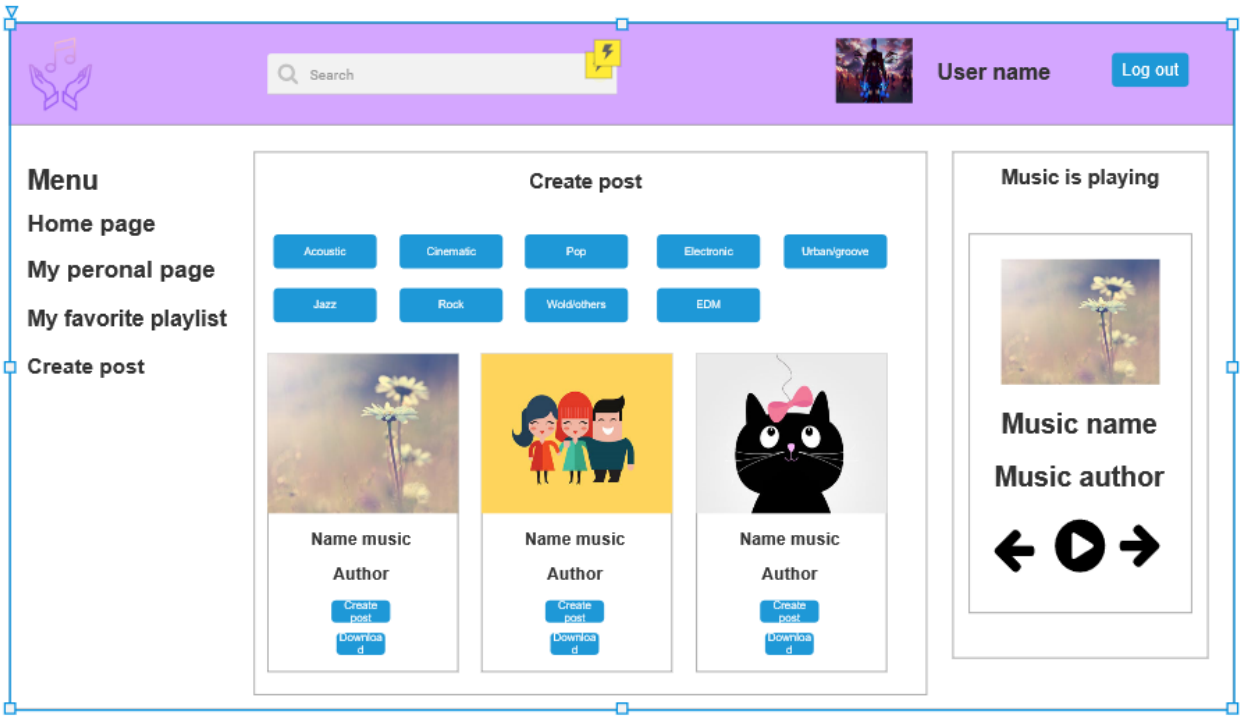

Figure 39: Create post design

CHAPTER 7: Implementation:

7.1. Set up environment:

First, I will install Nodejs to my computer through Nodejs homepage

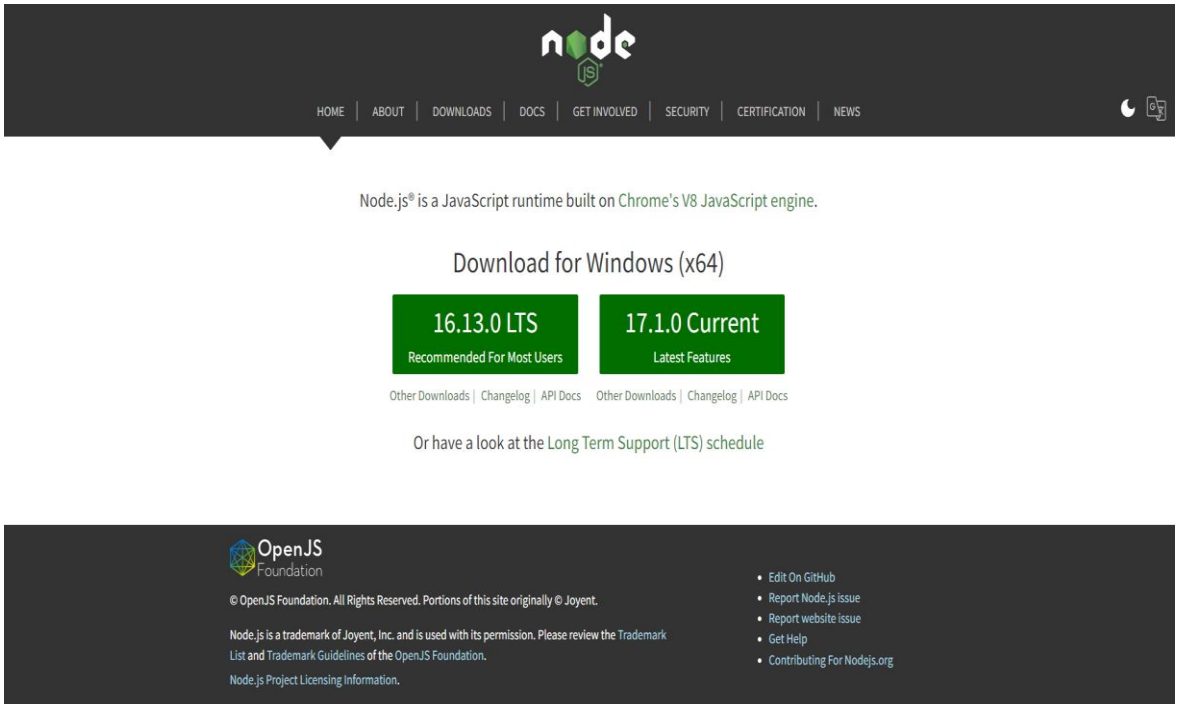

Figure 40: Nodejs install

Then I will use VScode to build the project

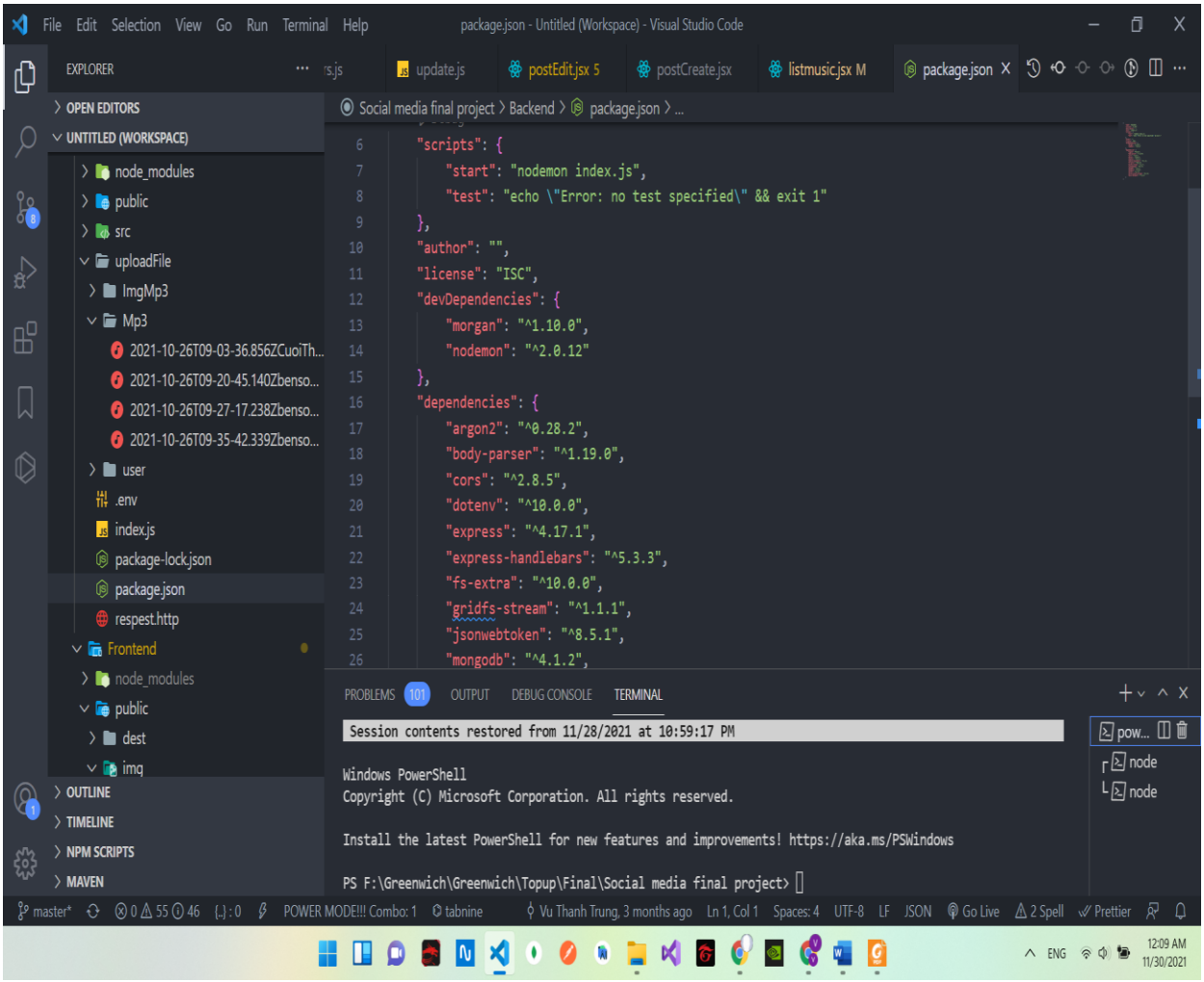

Figure 41: VScode

I will go to the terminal of the folder where I want to save the project then create 2 folders, Frontend, and Backend. The Frontend will represent the entire client code in terms of the interface for the entire project. And the Backend will represent the server to process information for the project.

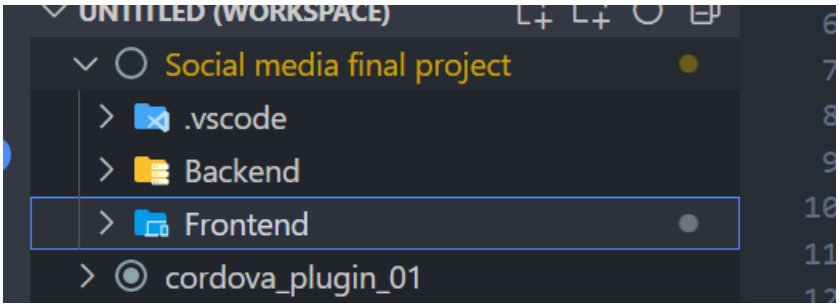

Figure 42: Folder project

Then for the Backend, I will install Expressjs for my project to support Nodejs. Express is a framework built on top of Nodejs. It provides powerful features for web developers and HTTP

and middleware methods to create APIs for client-server communication. After successful installation, my Backend folder structure will look like this

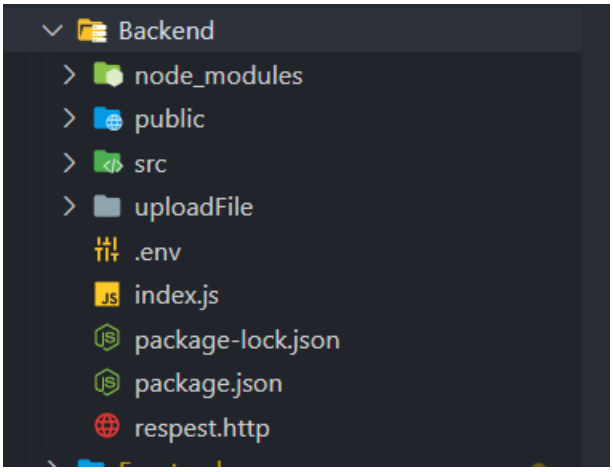

Figure 43: Backend directory structure

Vu Thanh Trung, a month ago | 1 author (Vu Thanh Trung)

```
{
    "name": "backend",        Vu Thanh Trung, 3 months ago • login google
    "version": "1.0.0",
    "description": "",
    "main": "index.js",
    ▷ Debug
    "scripts": {
        "start": "nodemon index.js",
        "test": "echo \"Error: no test specified\" && exit 1"
    },
    "author": "",
    "license": "ISC",
    "devDependencies": {
        "morgan": "^1.10.0",
        "nodemon": "^2.0.12"
    },
    "dependencies": {
        "argon2": "^0.28.2",
        "body-parser": "^1.19.0",
        "cors": "^2.8.5",
        "dotenv": "^10.0.0",
        "express": "^4.17.1",
        "express-handlebars": "^5.3.3",
        "fs-extra": "^10.0.0",
        "gridfs-stream": "^1.1.1",
        "jsonwebtoken": "^8.5.1",
        "mongodb": "^4.1.2",
        "mongoose": "^6.0.4",
        "multer": "^1.4.3",
        "multer-gridfs-storage": "^5.0.2",
        "music-metadata": "^7.11.4"
    }
}
```

Figure 44: Package.json of Backend

In the directory, there will be a file called package.json which is the configuration file of npm, which helps npm understand what it needs to install, information about the application, version,... Here I will see some more libraries that I install to help me develop the website.

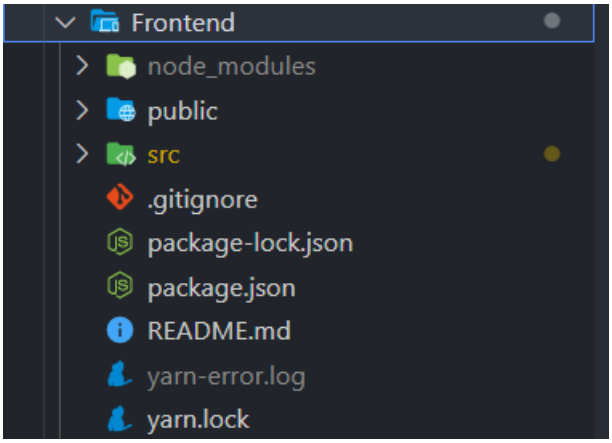

Figure 45: Frontend directory structure

Above is my Frontend file structure after I installed Reactjs library.

Figure 46: Package.json of Frontend

Here is my package.json file as well as the Backend file. I will manage to see what libraries I have installed in this.

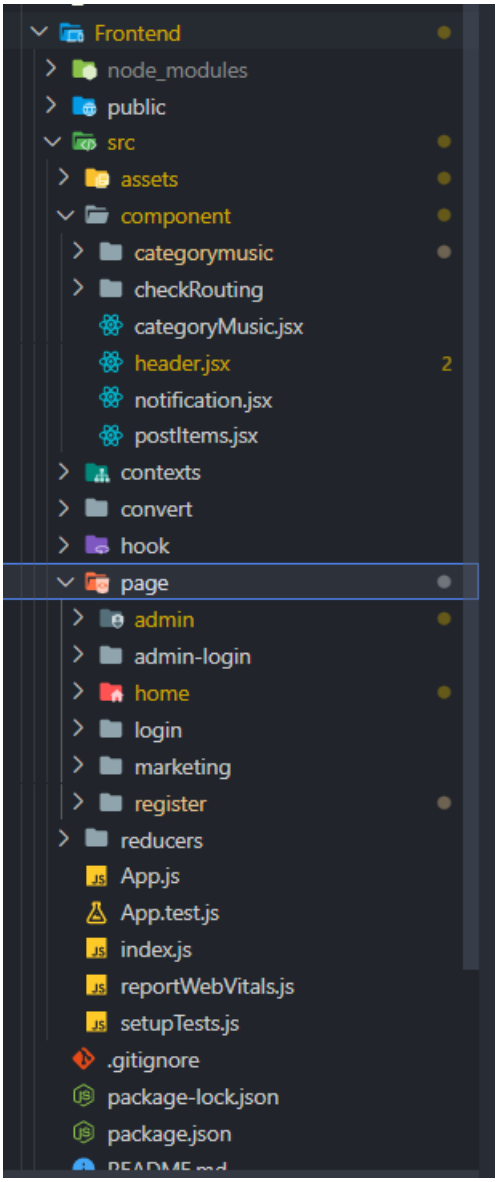

Figure 47: Folder structure after split into component

Reactjs divides the interface into different components for easy code management and maintenance. That's why I divide components for my website according to their main effects.

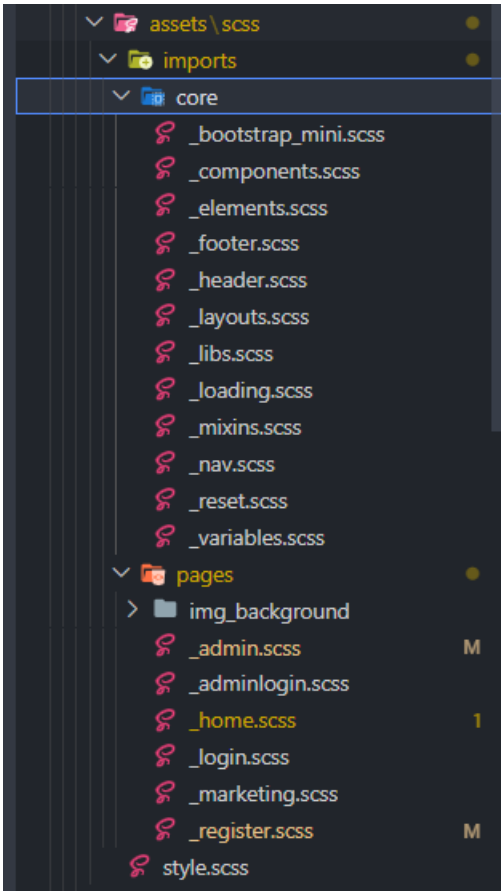

Figure 48: SCSS directory structure

And this is the SCSS technology for the design of the interface that replaces the usual css like before that I will apply to this project. And this is the SCSS technology for the design of the interface that replaces the usual css like before that I will apply to this project. I also divide each paragraph into different pages for easy management and use.

7.2. Database models:

I use mongoose for table initialization for my database. And Shcema in mongoose will help me in this.

```
Vu Thanh Trung, a month ago | 1 author (Vu Thanh Trung)
1  const mongoose = require("mongoose");      Vu Thanh Trung, a month ago • update infomati
2  const Schema = mongoose.Schema;
3
4  const CategorySchema = new Schema({
5      cateName: {
6          type: "string",
7      },
8  });
9
10 module.exports = mongoose.model("categories", CategorySchema);
```

```
Vu Thanh Trung, a day ago • not enough
1  const mongoose = require("mongoose");
2  const Schema = mongoose.Schema;
3
4  // Favorites collection in mongodb atlas
5  const FavoriteSchema = new Schema({
6      user: {
7          type: Schema.Types.ObjectId,
8          ref: "users",
9      },
10     music: {
11         type: Schema.Types.ObjectId,
12         ref: "musics",
13     },
14     favoriteCreationDate: {
15         type: Date,
16         default: Date.now,
17     },
18 });
19
20 module.exports = mongoose.model("favorites", FavoriteSchema);
```

```
Social media final project > Backend > src > resource > models > musics.js > MusicSc
Vu Thanh Trung, a month ago | 1 author (Vu Thanh Trung)
1  const mongoose = require("mongoose");
2  const Schema = mongoose.Schema;
3
4  const MusicSchema = new Schema({
5      musicName: {
6          type: String,
7          required: true,
8      },
9      musicAuthor: {
10         type: String,
11     },
12     musicImg: {
13         type: String,
14         contentType: String,
15     },
16     musicFile: {
17         type: String,
18         contentType: String,
19     },
20     musicCategory: {
21         type: Schema.Types.String,
22         ref: "categories",        Vu Thanh Trung, a month ago • upd
23     },
24     musicLike: {
25         type: Number,
26     },
27     userId: {
28         type: Schema.Types.ObjectId,
29         ref: "users",
30         default: "61600aa78b331c8efe5a3d84",
31     },
32     musicCreationDate: {
33         type: Date,
34         default: Date.now,
35     },
36 });
37
38 module.exports = mongoose.model("musics", MusicSchema);
```

```
Vu Thanh Trung, 3 days ago | 1 author (Vu Thanh Trung)
const mongoose = require("mongoose");        Vu Thanh Trung, 5 days
const Schema = mongoose.Schema;

// post collection in mongodb atlas
const PostSchema = new Schema({
    user: {
        type: Schema.Types.ObjectId,
        ref: "users",
    },
    postContent: {
        type: String,
        required: true,
    },
    music: { type: Schema.Types.ObjectId, ref: "musics" },
    postCreationDate: {
        type: Date,
        default: Date.now,
    },
});

module.exports = mongoose.model("posts", PostSchema);
```

```
Vu Thanh Trung, 2 months ago | 1 author (Vu Thanh Trung)
const mongoose = require("mongoose");        Vu Thanh Trung, 2 month
const Schema = mongoose.Schema;

const RoleSchema = new Schema({
    roleType: String,
    required: true,
});

module.exports = mongoose.model("role", RoleSchema);
```

```
...
const mongoose = require("mongoose");
const Schema = mongoose.Schema;

// user colection in mongoDb atlas
const UserSchema = new Schema({
    userEmail: {
        type: String,
        required: true,
        unique: true,
    },
    userPassword: {
        type: String,
        required: true,
    },
    userName: {
        type: String,
        required: true,
    },

    userAvatar: {
        // data: Buffer,
        type: String,
        contentType: String,
    },

    roleId: {
        type: Schema.Types.ObjectId,
        ref: "role",
        default: "615aba747b19409446d0128f",
    },
    userCreationDate: {
        type: Date,
        default: Date.now,
    },
});

module.exports = mongoose.model("users", UserSchema);
```

7.3. MongoDB:

After initializing and importing data, the data will be saved to MongoDB.

categories	9	42.8 B	385.0 B	1	36.0 KB	
comments	2	118.0 B	236.0 B	1	36.0 KB	
favorites	17	98.0 B	1.6 KB	1	36.0 KB	
musics	4	303.5 B	1.2 KB	1	36.0 KB	
posts	9	126.9 B	1.1 KB	1	36.0 KB	
role	2	41.5 B	83.0 B	1	20.0 KB	
users	3	310.0 B	930.0 B	2	72.0 KB	

Figure 49: Mongodb

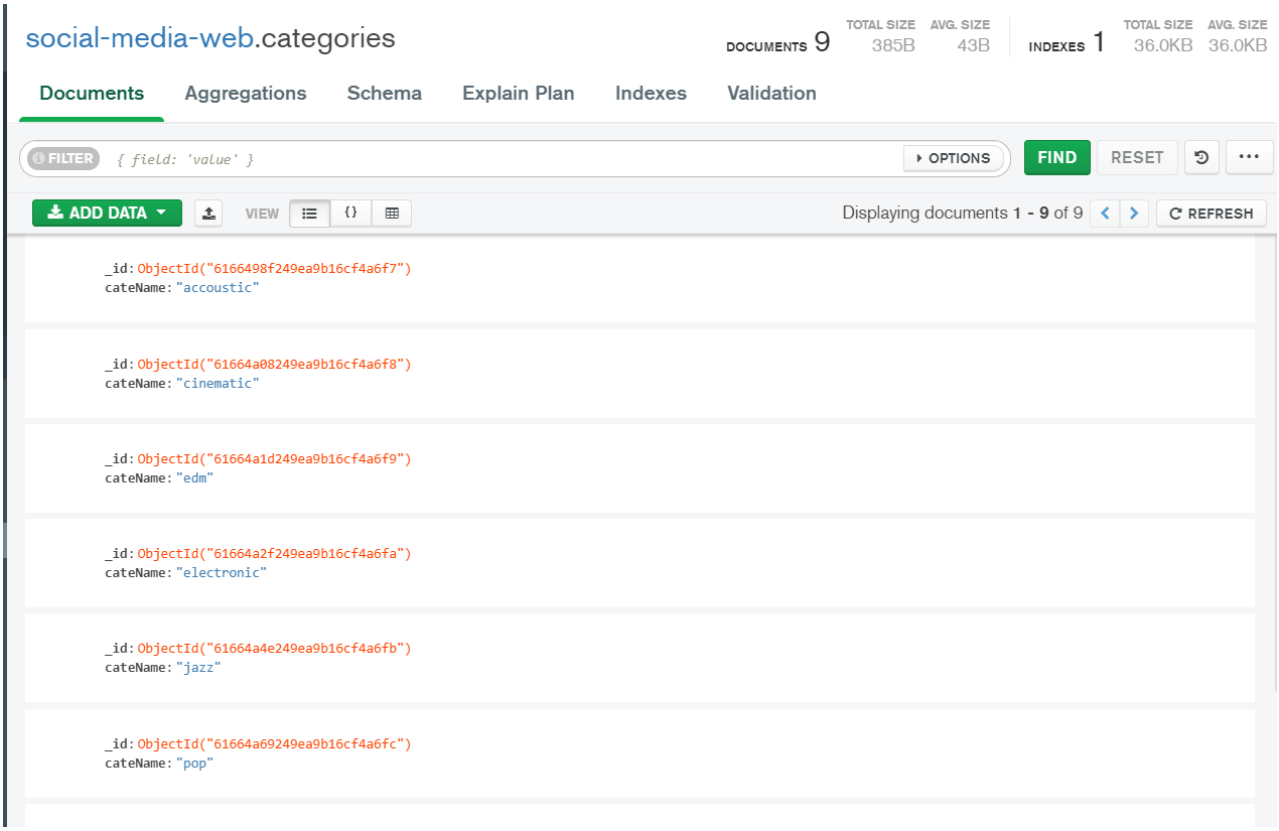

Figure 50: Categories collection

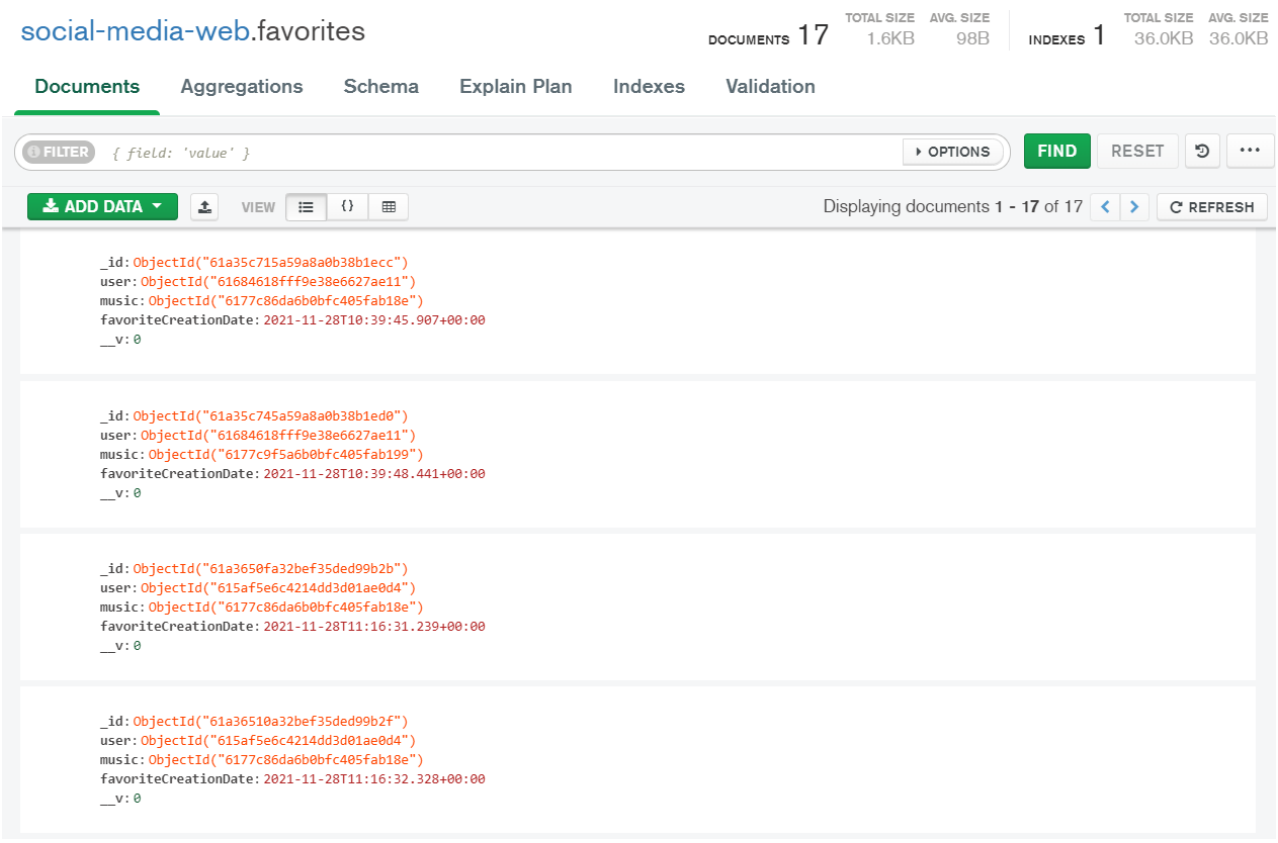

Figure 51: Favorites collection

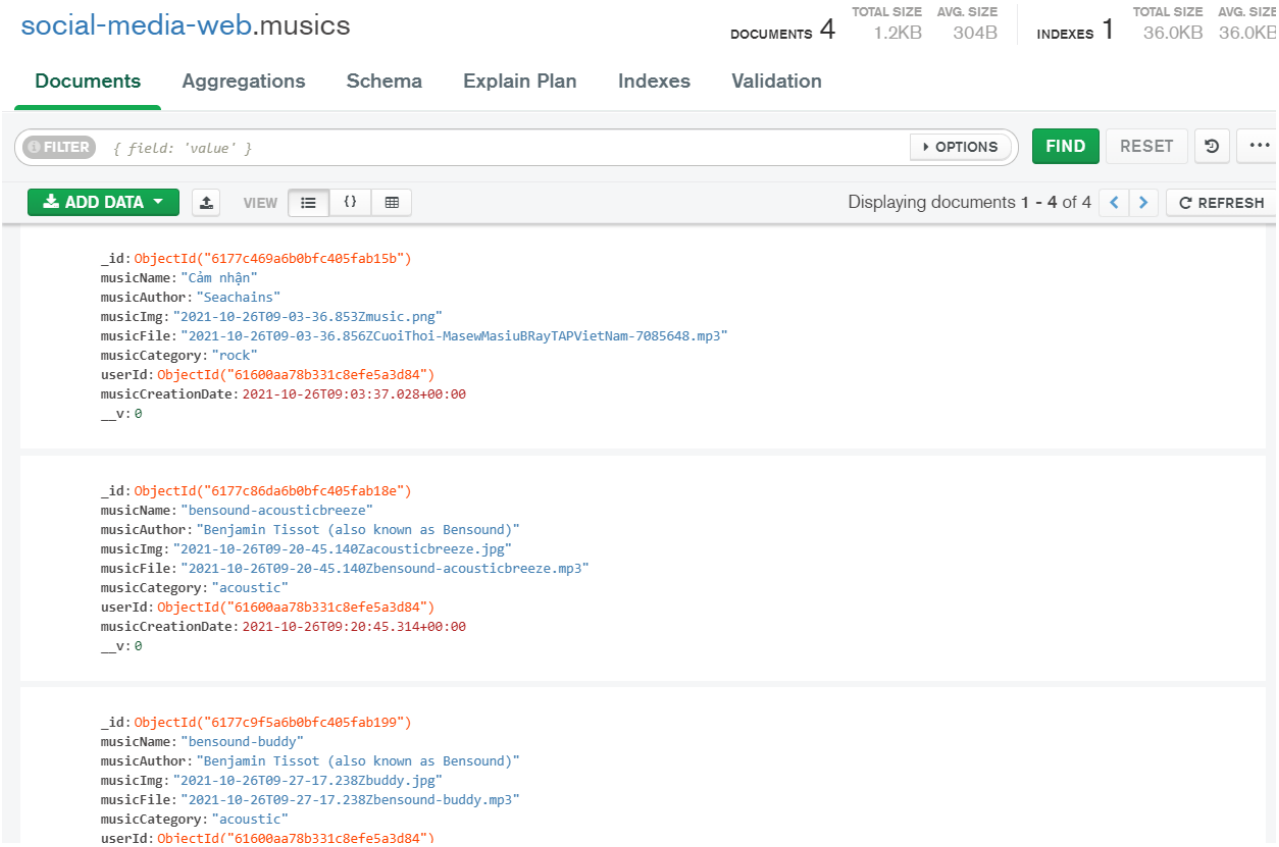

Figure 52: Musics collection

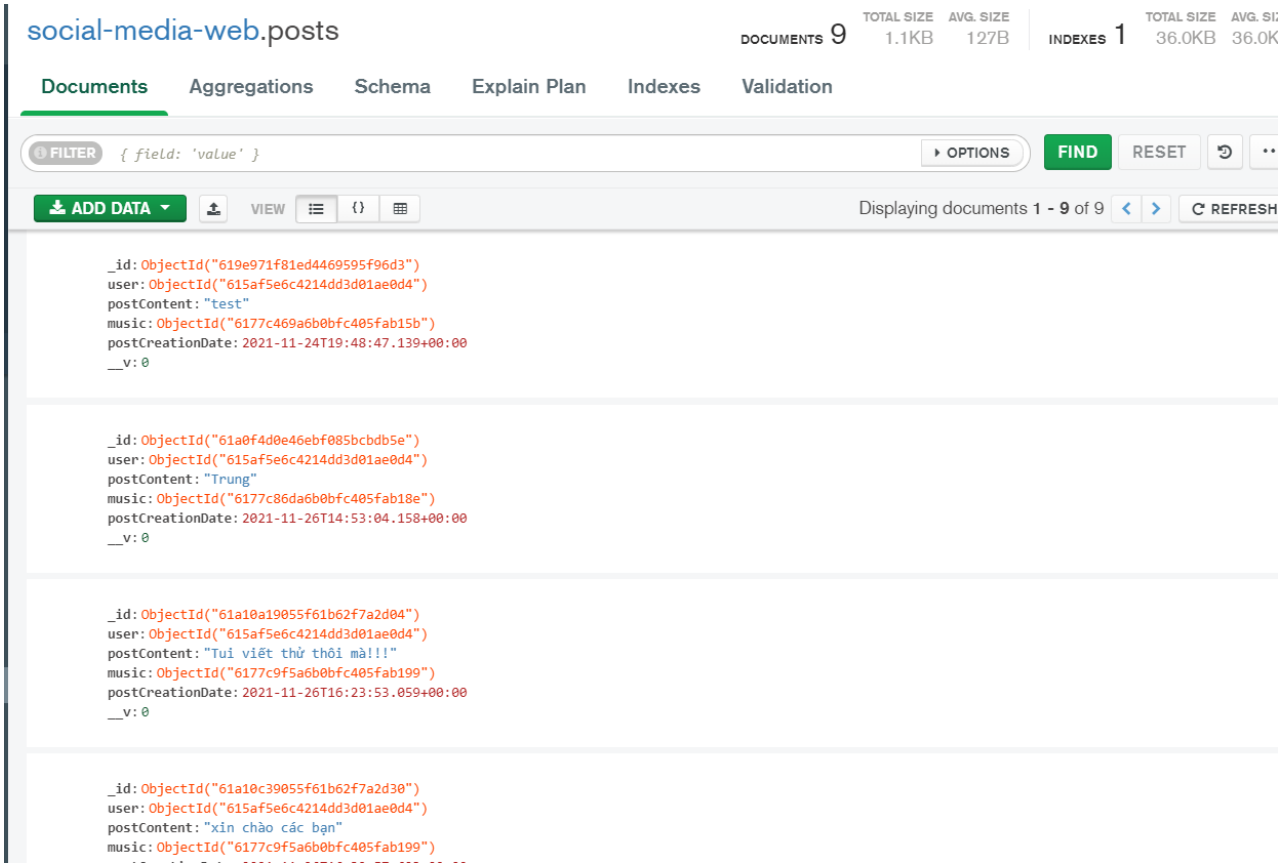

Figure 53: Posts collection

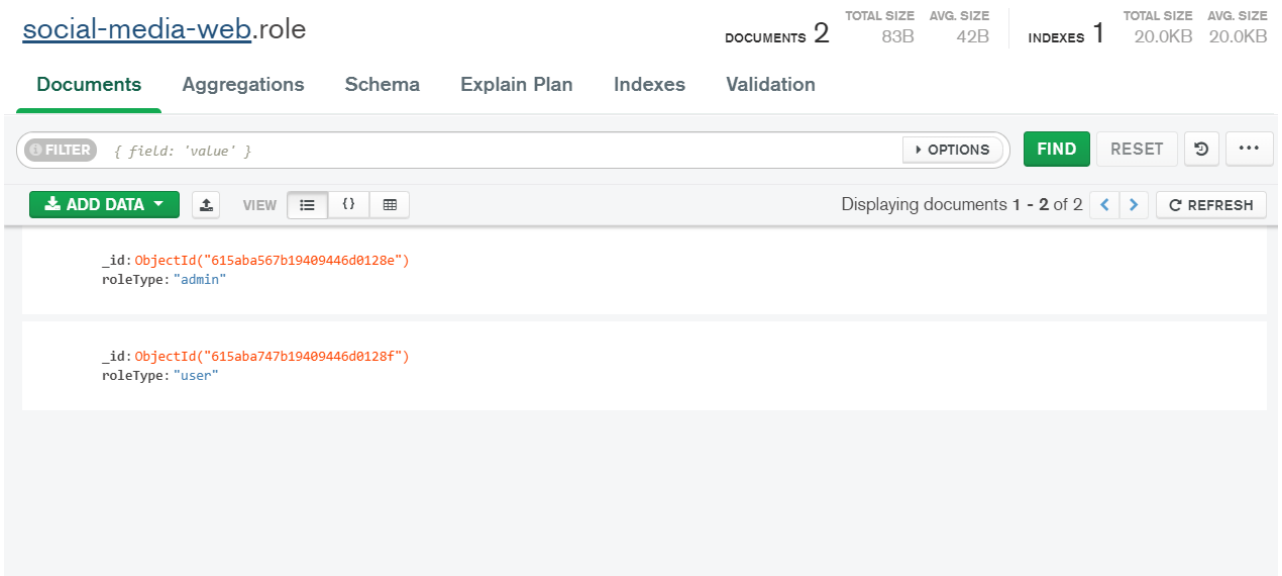

Figure 54: Role collection

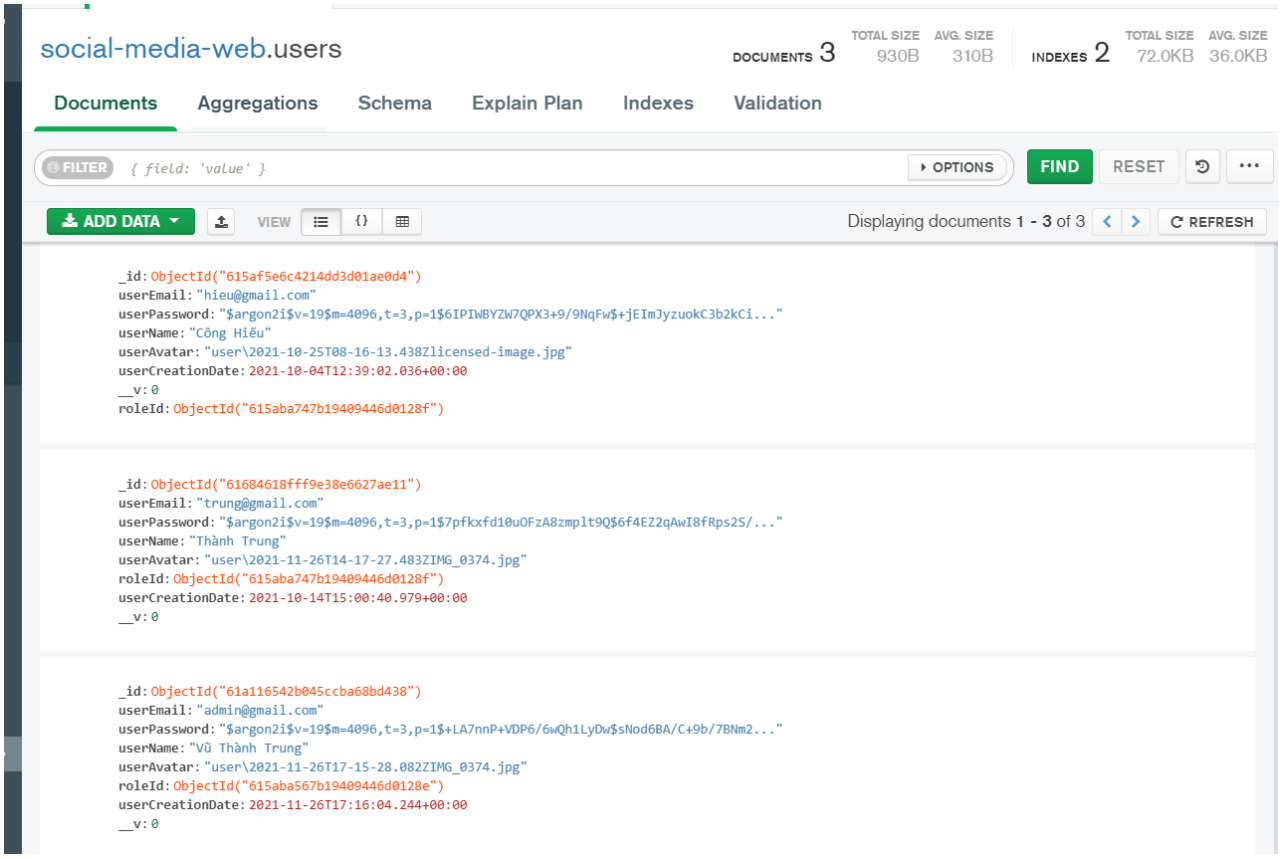

Figure 55: Users collection

7.4. Routing for each page:

I use the routing library React-router-dom to route all the pages with my site.

```
<BrowserRouter>
  <div className="App">
    <Switch>
      <Route exact path="/" component={Marketing} />{" "}
      <Route path="/login" component={Login} />{" "}
      <Route path="/register" component={Register} />{" "}
      <CheckRedirect path="/home" component={Home} />{" "}
      <Route path="/login-admin" component={AdminLogin} />{" "}
      <ProtectAdmin path="/admin" component={Admin} />{" "}
    </Switch>{" "}
  </div>{" "}
</BrowserRouter>{" "}
```

7.5. Marketing page:

Almost every social networking site or e-commerce site has a marketing page to help promote its website. That's why I decided to design an e-commerce website with some eye-catching animations and a nice design to entice users to use my site. When the user scrolls down, some images and text will be parallax to create interest for the user. Here I put a few slogan sentences as well as key information such as the name of my website here so that users have an

overview of my website before using it. When the user scrolls to the bottom of the website, there will be a large login button in the middle, prominent to attract the attention of people there to click login.

Figure 56: Marketing page screenshots

Figure 57: Marketing page 2 screenshots

Figure 58: Marketing page 3 screenshot

This is my marketing interface code after it has been divided into components and some functions to help create a paralx effect for text and images on the page.

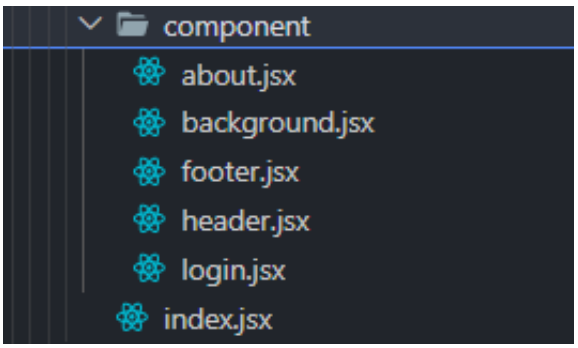

Figure 59: Marketing page component

Figure 60: Marketing page component 2

7.6. Login page screenshots:

After the user clicks the login button outside the marketing page, they will be redirected to the login page. Here, if the user already has an account, they will log in and go to the website. If not, they need to click on the text below "click here to register" to register an account.

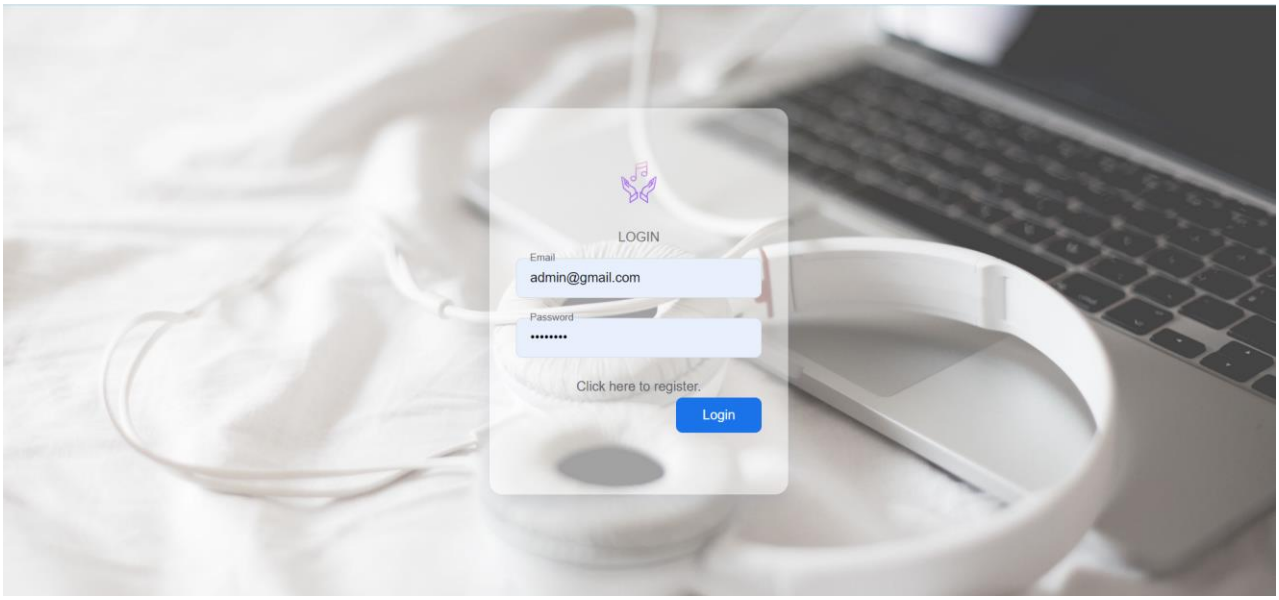

Figure 61: Login page screenshots

```
router.post("/login", uploadlogin.array(), async(req, res) => {
    const { userEmail, userPassword } = req.body;

    // Simple validation      Vu Thanh Trung, 2 months ago • sever: login test
    if (!userEmail || !userPassword)
        return res
            .status(400)
            .json({ success: false, message: "Missing userEmail and/or password" });

    try {
        // All good
        // check role

        // check if this user is registered in the database
        const user = await User.findOne({ userEmail: userEmail });
        if (user.roleId == "615aba567b19409446d0128e") {
            if (!user)
                return res
                    .status(400)
                    .json({ success: false, message: "Incorrect username or password" });

            // Check user's password
            const passwordValid = await argon2.verify(
                user.userPassword,
                userPassword
            );
            if (!passwordValid)
                return res
                    .status(400)
                    .json({ success: false, message: "Incorrect username or password" });

            // Return token
            const accessToken = jwt.sign({ userId: user._id },
                process.env.ACCESS_TOKEN_SECRET
            );

            res.json({
                success: true,
                message: "Admin logged in successfully",
                accessToken,
            });
        }
        if (!user)
            return res
                .status(400)
                .json({ success: false, message: "Incorrect username or password" });
```

Figure 62: Code login in sever

First I will set the API and post method to receive data from the client that the user input. Then I will check if the data has been entered or not. If not, the error message is in the form of json. And will check if this user already has an account in the database through the findOne() function. Then enter the data if the user already exists. After that, it will check whether the input information matches the account information stored in the database or not. With the password for security purposes, I encrypted it, so I have to use the argon2 library to compile the password and then check if the password is correct or not. Before doing this, I will first check the user's roleId is admin or not. If it is an admin, it returns a json message that this is an admin successfully logged in. If it is a user, it will notify you that this is a successfully logged in user

```
// reducer save state for authenticated user
export const authReducer = (state, action) => {
    // First state for login web
    const {
        type,
        payload: { isAuthenticated, user },
    } = action;

    switch (type) {
        case `SET_AUTH`:
            return {
                ...state,
                authLoading: false,
                isAuthenticated,
                user,
            };

        default:
            return state;
    }
};
```

Figure 63: AuthRuducer

```
useEffect(() => loadUser(), []);

// Login
const loginUser = async (userForm) => {
  try {
    // Send req for sever
    const response = await axios.post(`${apiUrl}/auth/login`, userForm);
    // if successful, return accessToken
    if (response.data.success)
      localStorage.setItem(
        LOCAL_STORAGE_TOKEN_NAME,
        response.data.accessToken
      );

    await loadUser();
    // return response user data
    return response.data;
  } catch (error) {
    if (error.response.data) return error.response.data;
    else return { success: false, message: error.message };
  }
};

// Logout
```

Figure 64: Login in AuthContext

```
    userPassword: "",
  });

  const [alertLogin, setAlertLogin] = useState(null);

  // When the data changes, the login changes accordingly
  const { userEmail, userPassword } = login;      Vu Thanh Trung, 2 months ago • client login successful

  // get new data entered by user
  const onchangeLogin = (event) => {
    setLogin({ ...login, [event.target.name]: event.target.value });
  };

  const userLogin = async (event) => {
    //avoid login according to the original data of html
    event.preventDefault();

    try {
      // call the function AuthContext use this page state
      const loginData = await loginUser(login);

      console.log(loginData);
      if (loginData.success) {
        setAlertLogin({ type: "danger", message: loginData.message });
        setTimeout(() => setAlertLogin(null), 5000);
        if (loginData.message == "Admin logged in successfully") {
          history.push("/admin");
        }
        if (loginData.message == "User logged in successfully") {
          history.push("/home");
        }
      } else {
        alert("UserEmail or password incorrect. Please try again");
      }
    } catch (error) {
      console.log(error);
    }
  };

```

Figure 65: Login function at component

The data will be received after the user successfully enters the account from within the user's login form. Then it will be sent to the server to check the data. After checking and authenticating, the user will be able to access the website according to their roleId. After they log in their data will be saved to the browser's localStorage via an accessToken returned from the server. After the user logs out, their localStorage will be deleted to ensure security.

```
};
return (
  <div className="login">
    <div className="l-form">
      <form onSubmit={userLogin} className="form">
        <img src="/img/logo1.svg" alt="" />
        <h2 className="form__title">Login</h2>
        <div className="form__div">
          <input
            type="text"
            className="form__input"
            placeholder=" "
            name="userEmail"
            value={userEmail}
            required
            onChange={onchangeLogin}
          />
          <label htmlFor className="form__label">
            Email
          </label>
        </div>
        <div className="form__div">
          <input
            type="password"
            className="form__input"
            placeholder=" "
            name="userPassword"
            value={userPassword}
            required
            onChange={onchangeLogin}
          />
          <label htmlFor className="form__label">
            Password
          </label>
        </div>
        <Link to="/register" className="register-btn">
          Click here to register.
        </Link>
        <input
          variant="success"
          type="submit"
          className="form__button"
          defaultValue="Sign In"
          value="Login"
        />
      </form>
    </div>
  </div>
);
```

Figure 66: Form login

7.7. Register screenshots:

At the registration page, users can register their own information to register for an account to access the website. My goal is to have a simple and soft design to make it easier for people to look at. If the user enters missing information, the system will notify them so that they can enter all the information needed to log in.

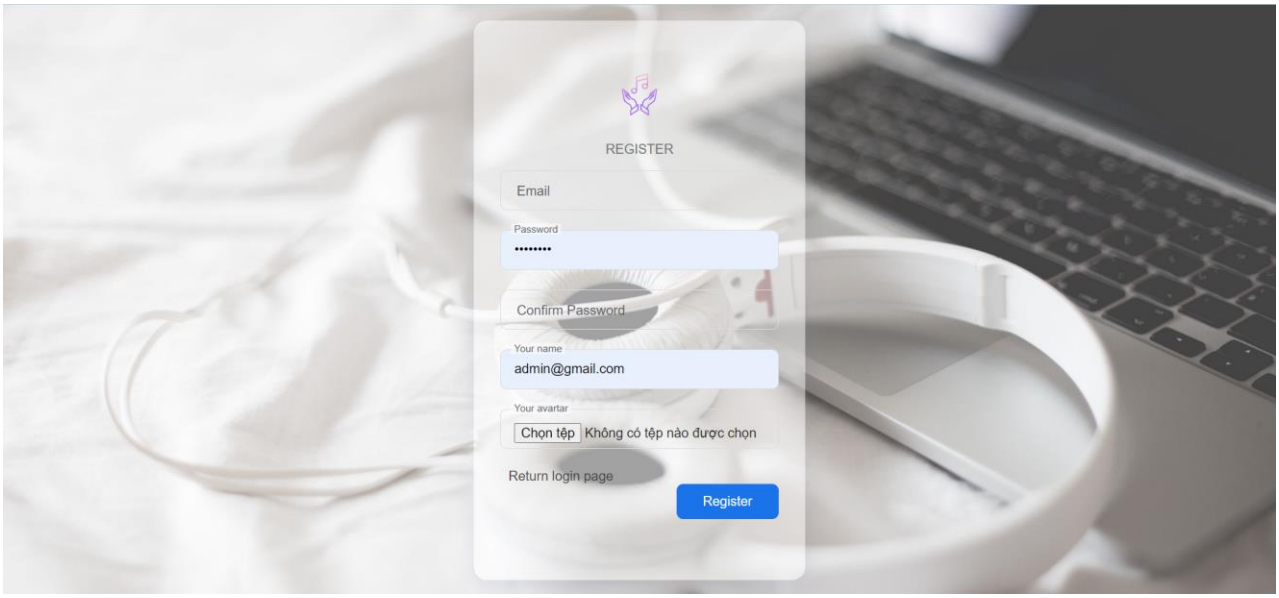

Figure 67: Register page screenshots

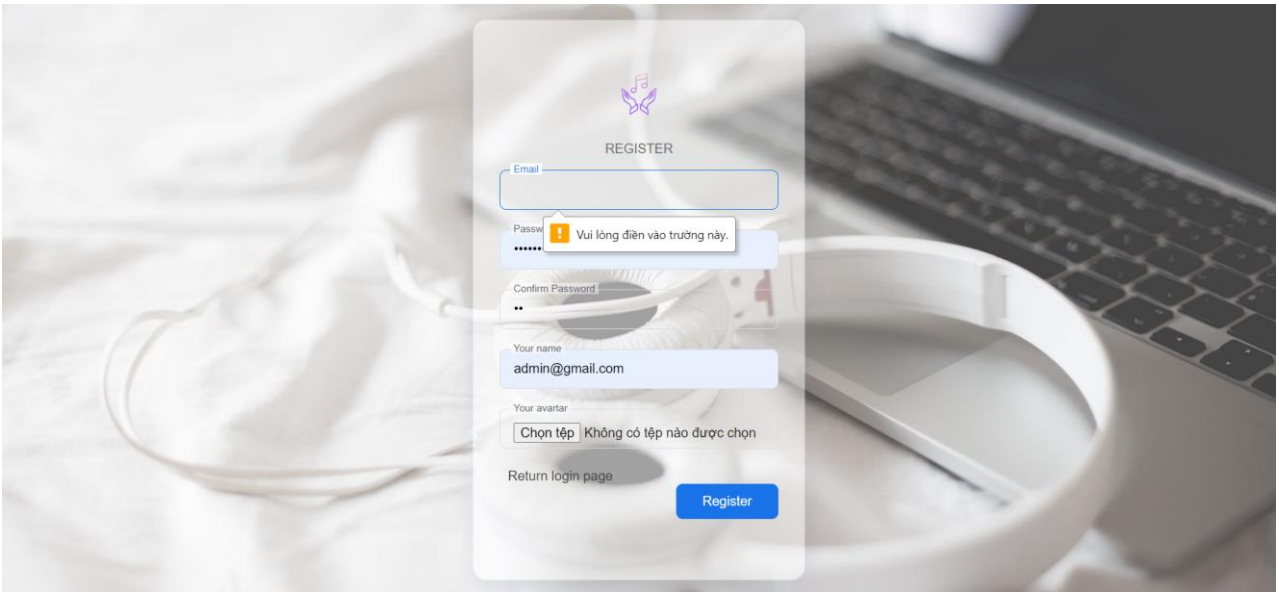

Figure 68: Register page 2 screenshots

This is code for the register section under Backend of I.

```
// @access public
router.post("/register", upload.single("userAvatar"), async(req, res) => {
    const { userEmail, userPassword, userName, userAvatar, roleId } = req.body;

    // Simple validation
    if (!userEmail || !userPassword || !userName)
        return res
            .status(400)
            .json({ success: false, message: "Missing userEmail and/or password" });

    try {
        // Check for existing user
        const user = await User.findOne({ userEmail });

        if (user)
            return res
                .status(400)
                .json({ success: false, message: "Username already taken" });

        // Hash password with argon2
        const hashedPassword = await argon2.hash(userPassword);

        const avatar = req.file.path;

        // Save this user to the database if everything ok
        const newUser = new User({
            userEmail,
            userPassword: hashedPassword,
            userName,
            userAvatar: avatar.replace("uploadFile\\", ""),
            roleId,
        });
        await newUser.save();       Vu Thanh Trung, 2 months ago • sever: login test

        // Return token
        const accessToken = jwt.sign({ userId: newUser._id },
            process.env.ACCESS_TOKEN_SECRET
        );

        const file = req.file;

        res.json({
            success: true,
            message: "User created successfully",
            accessToken,
            file,
        });
    } catch (error) {
        console.log(error);
        res
            .status(500)
            .json({ success: false, message: "Internal server error !!!!" });
    }
});
```

Figure 69: Register at Backend

I use middlewire as multer to help me handle saving information of mp3 and image files. Next is the Frontend code of the register section.

```
import axios from "axios";  18.8K (gzipped: 7K)
import React, { useContext, useState } from "react";  7.2K (gzipped: 3K)
import { useHistory } from "react-router";  10.1K (gzipped: 4K)
import { AuthContext } from "../../contexts/authContext";
import { apiUrl, LOCAL_STORAGE_TOKEN_NAME } from "../../contexts/constants";
import { Link } from "react-router-dom";  14.7K (gzipped: 5.8K)

export default function Register() {
  // Context
  const { loadUser } = useContext(AuthContext);

  // local state
  const [register, setRegister] = useState({
    userEmail: "",
    userPassword: "",
    confirmPassword: "",
    userName: "",
    userAvatar: "",
  });

  //Router
  const history = useHistory();

  // get data as user input
  const onChangeRegisterForm = function (event) {
    setRegister({ ...register, [event.target.name]: event.target.value });
  };

  //get file as user input
  const onChangeFileUserForm = function (event) {
    setRegister({ ...register, userAvatar: event.target.files[0] });
    // console.log(event.target.files[0]);
  };

  // handle submit event in form with axios
  const userRegister = (e) => {
    e.preventDefault();

    // initialize formdata to store values in state and assign those values to name in input
    const formData = new FormData();
    formData.append("userEmail", register.userEmail);
    formData.append("userPassword", register.userPassword);
    formData.append("confirmPassword", register.confirmPassword);
    formData.append("userName", register.userName);
    formData.append("userAvatar", register.userAvatar);

    axios
      .post(`${apiUrl}/auth/register`, formData)
      .then((response) => {
        console.log(response.data);
        if (response.data.success) {
          localStorage.setItem(
            LOCAL_STORAGE_TOKEN_NAME,
            response.data.accessToken
          );
```


Figure 70: Register at Frontend

```
  e.preventDefault();

  // initialize formdata to store values in state and assign those values to name in input
  const formData = new FormData();
  formData.append("userEmail", register.userEmail);
  formData.append("userPassword", register.userPassword);
  formData.append("confirmPassword", register.confirmPassword);
  formData.append("userName", register.userName);
  formData.append("userAvatar", register.userAvatar);

  axios
    .post(`${apiUrl}/auth/register`, formData)
    .then((response) => {
      console.log(response.data);
      if (response.data.success) {
        localStorage.setItem(
          LOCAL_STORAGE_TOKEN_NAME,
          response.data.accessToken
        );
        history.push("/login");
        loadUser();
        return response;
      }
      if (!response.success) {
        alert(response.data.message);
      }
    })
    .catch((error) => {
      if (error.response) return error.response;
      else return { success: false, message: error.message };
    });
};
```

Figure 71: Register at Frontend 2

I will use the form to receive the user's data and then check if the information is enough. As well as check if the password is duplicate or not. If not, notify the user. After checking enough information, it will send data to the sever through API using axios with POST method. The backend after receiving the data will process the data and encrypt the password and then save it to the database.

7.8. Admin dashboard screenshot:

For admins, after entering the account assigned to the admin, they will be redirected to the admin dashboard page. This page will show some key information related to user activity and post, track, or user metrics. This page will be the admin page to monitor the number of users using the website, the number of songs on the page, and the likes of the songs. The left side of the screen shows the page's menu and the admin after logging in can switch to some other pages to perform other functions. If the admin wants to log out, just press the sign out button in the bottom left corner of the screen. After logging out, you will be redirected to the original login page.

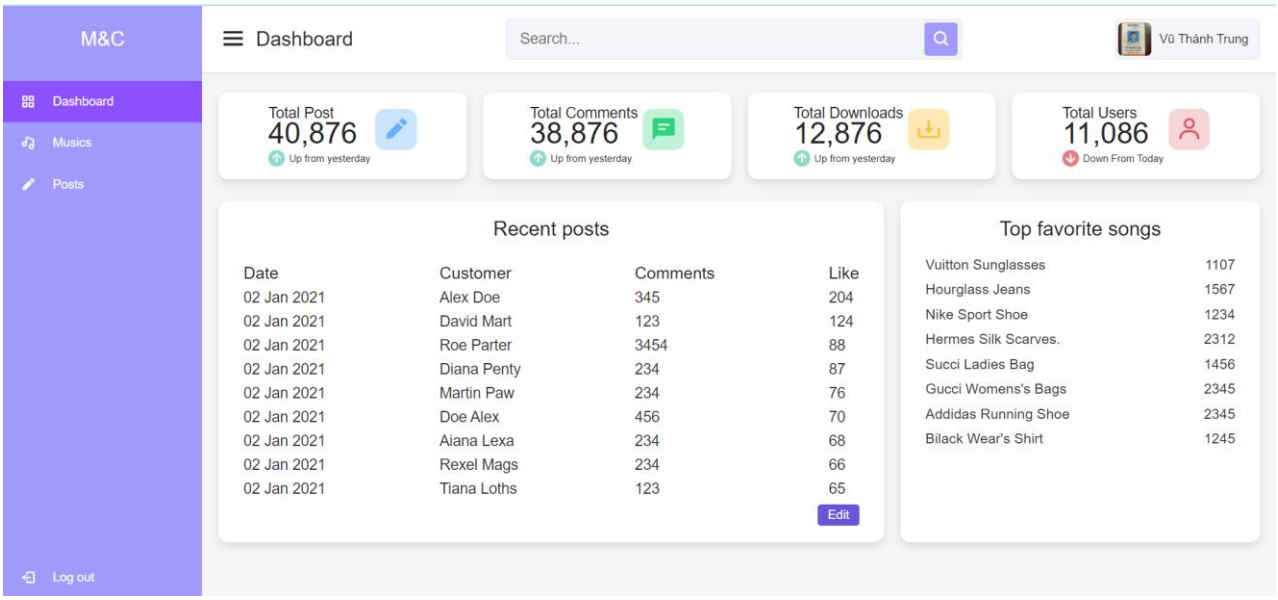

Figure 72: Admin dashboard page screenshots

7.9. Admin music page screenshots:

After the admin clicks the music button in the menu on the left side of the screen, it will be redirected to the music page. The layout of the menu and the information display and search bar above will remain the same to serve when needed. There will be a form to enter song data so that the admin can add new songs to the website. After admin added the song, scrolling down below the shout song will be added to its correct category. Admin can press the button to edit or delete the song if necessary.

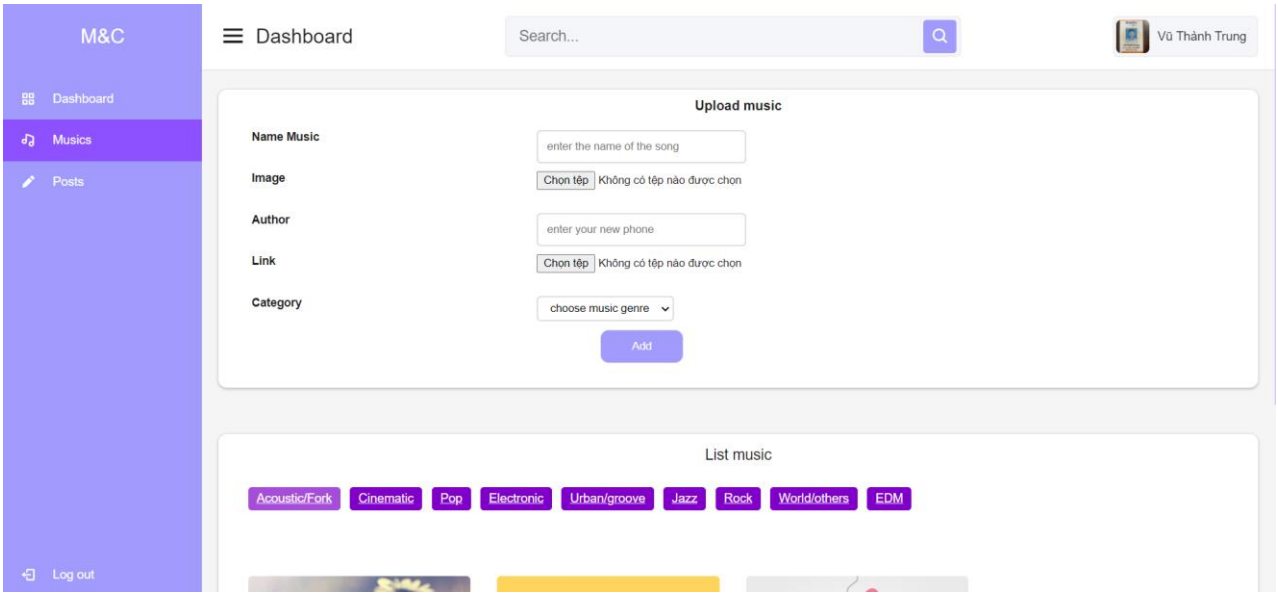

Figure 73: Admin music page screenshots

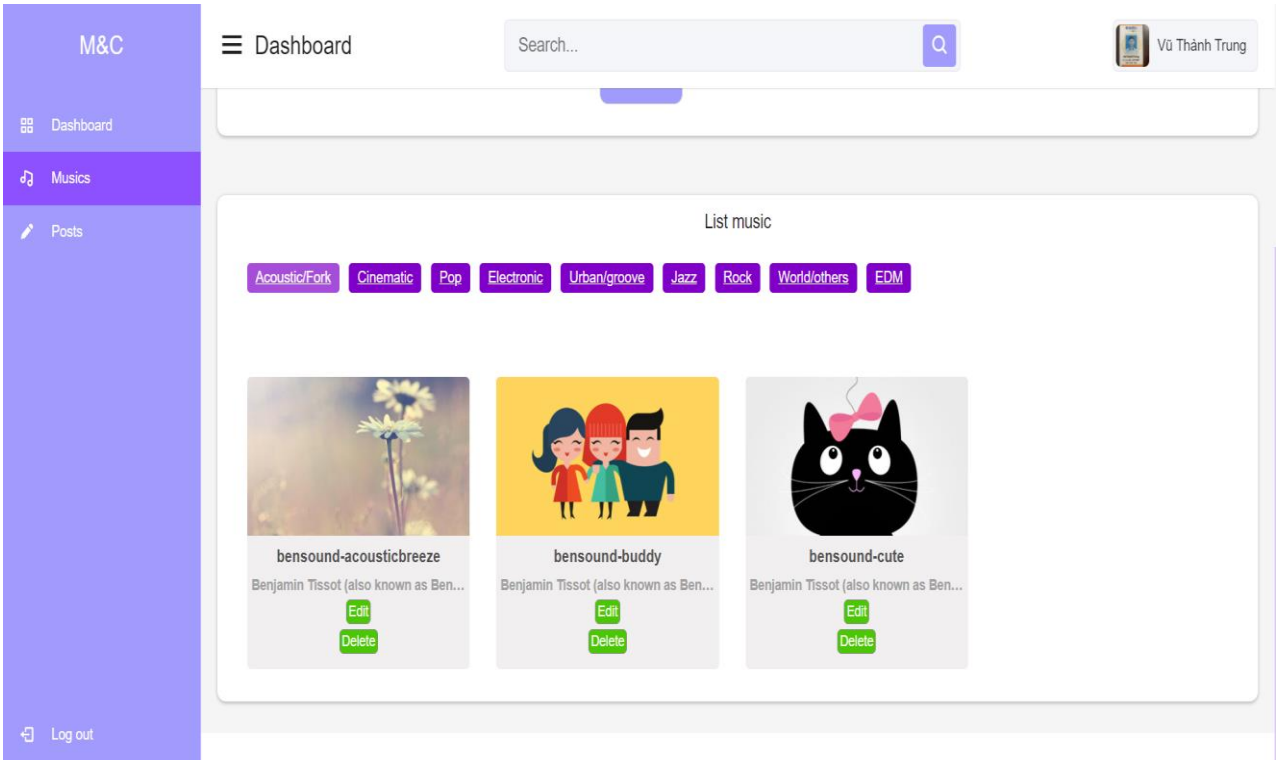

Figure 74: Admin music page 2 screenshots

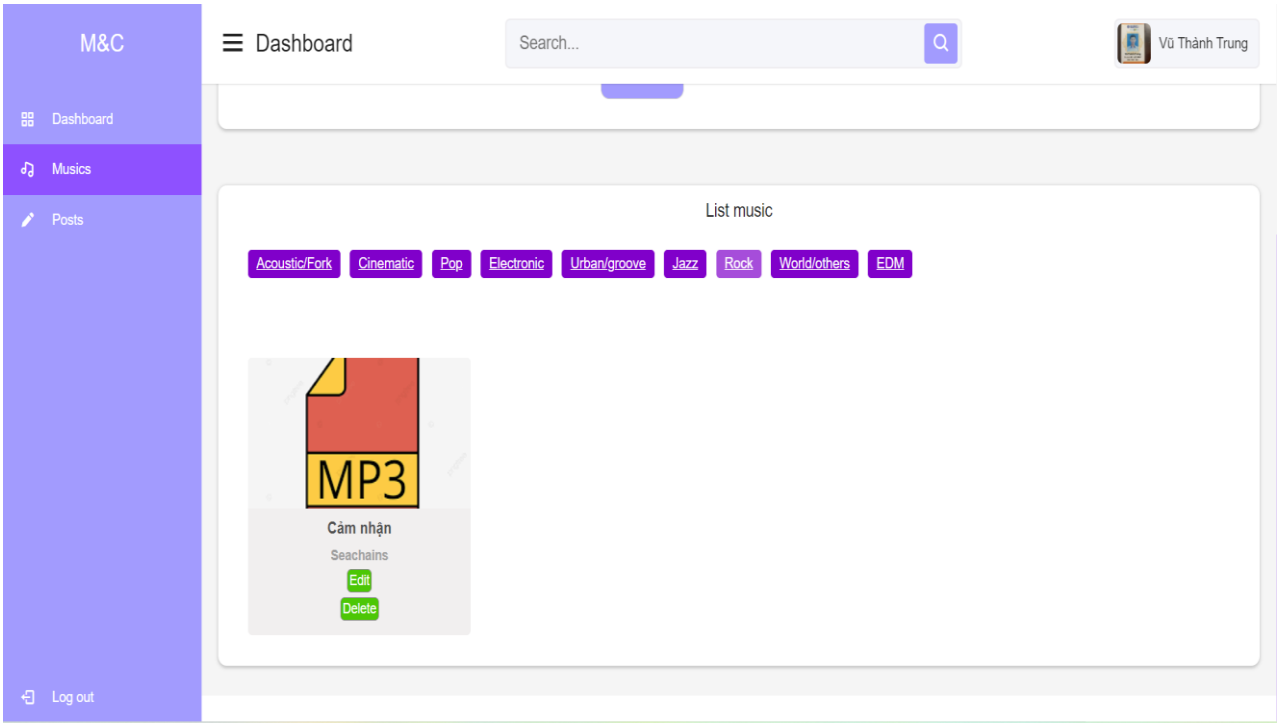

Figure 75: Admin music page 3 screenshots

This is the code under my Backend for adding music to the database as well as getting the music information in the database through the GET and POST methods of the API provided by Express.

```
// @desc Get musics
const router: Router
router.get("/", async(req, res) => {
    try {
        const musics = await Music.find(req);
        res.json({ success: true, musics });
    } catch (error) {
        console.log(error);       Vu Thanh Trung, 3 days ago • create posts backend/frontend do
        res.status(500).json({ success: false, message: "Internal server error" });
    }
});

const middlewareMp3 = uploadMp3Test.fields([{
        name: "musicFile",
        maxCount: 10,
    },
    {
        name: "musicImg",
        maxCount: 10,
    },
]);
```

Figure 76; Get music Backend

```
28
29  router.post("/upload", middlewareMp3, async(req, res) => {
30      const { musicName, musicAuthor, musicImg, musicFile, musicCategory } =
31      req.body;
32
33      // Simple validation
34      if (!musicName || !musicAuthor || !musicCategory)
35          return res
36              .status(400)
37              .json({ success: false, message: "Missing information" });
38
39      try {
40          // const imgMP3 = req.file.path;
41          const imgMp3 = req.files.musicImg.map((element) => element.path);
42          console.log(imgMp3);
43          const fileMp3 = req.files.musicFile.map((element) => {
44              return element.path;
45          });
46          console.log(fileMp3.toString());
47
48          // Save this user to the database if everything ok
49          const newMusic = new Music({
50              musicName,
51              musicAuthor,
52              musicImg: imgMp3.toString().replace("uploadFile\\ImgMp3\\", ""),
53              musicFile: fileMp3.toString().replace("uploadFile\\Mp3\\", ""),
54              musicCategory,
55          });
56
57          await newMusic.save();
58
59          res.json({
60              success: true,
61              message: "Music created successfully",
62          });
63      } catch (error) {
64          console.log(error);
65          res
66              .status(500)
67              .json({ success: false, message: "Internal server error !!!!" });
68      }
69  });
70
71  module.exports = router;
```

Figure 77Post music Backend

Here I have used multer for receiving and saving image and music file data for the database.

```
    // initialize formdata to store values in state and assign those values to name in input
    const formData = new FormData();
    formData.append("musicName", uploadMusic.musicName);
    formData.append("musicAuthor", uploadMusic.musicAuthor);
    formData.append("musicImg", uploadMusic.musicImg);
    formData.append("musicFile", uploadMusic.musicFile);
    formData.append("musicCategory", uploadMusic.musicCategory);

    axios
      .post(`${apiUrl}/music/upload`, formData)
      .then((response) => {
        console.log(response.data);
        if (response.data.success) {
          dispatch({ type: ADD_MUSIC, payload: response.data.post });

          alert(response.data.message);
          return response.data;
        }
        if (!response.success) {
          alert(response.data.message);
        }
      })
      .catch((error) => {
        if (error.response) return error.response;
        else return { success: false, message: error.message };
      });
  };

```

Figure 78: Add a new music at Frontend

After having an API for saving music information from the Backend, I call that API at the Frontend and then send the information received from the user through the Form to send it down to the Backend to save new data to the database.

```
export default function AcousticAdmin() {
  const {
    musicState: { music, musics, musicsLoading },
    getMusics,
  } = useContext(MusicContext);
  // start get all musics
  useEffect(() => getMusics(), []);

  const getCategory = musics.filter((music) => {
    return music.musicCategory == "acoustic";
  });
  console.log(getCategory);

  if (musicsLoading) {
    return (
      <div style={{ width: "100%", height: "100vh", position: "absolute" }}>
        <Ripple
          style={{
            top: "50%",
            left: "50%",
            position: "relative",
            transform: "translate(-50%, -50%)",
          }}
          color="#be97e8"
        />{" "}
        ;{" "}
      </div>
    );
  } else {
    return (
      <div className="category__music acoustic">
        {getCategory.map((music) => (
          <ListMusicAdmin music={music} />
        ))}
      </div>
    );
  }
}
```

Figure 79Render music at Frontend

And here I use the special hook useEffect to render the music information received from the Backend then use filter() to check the array for the correct music to render the correct page. I will pass the data down to the child component so that when rendering will get the data of each song.

7.10. Admin post page screenshot:

After the admin presses the post button in the admin menu, the admin will be redirected to the post page to manage the user's post. Here, if the admin finds a post inappropriate or hostile, on a case-by-case basis can delete that user's post or not with the delete button at the end of each post.

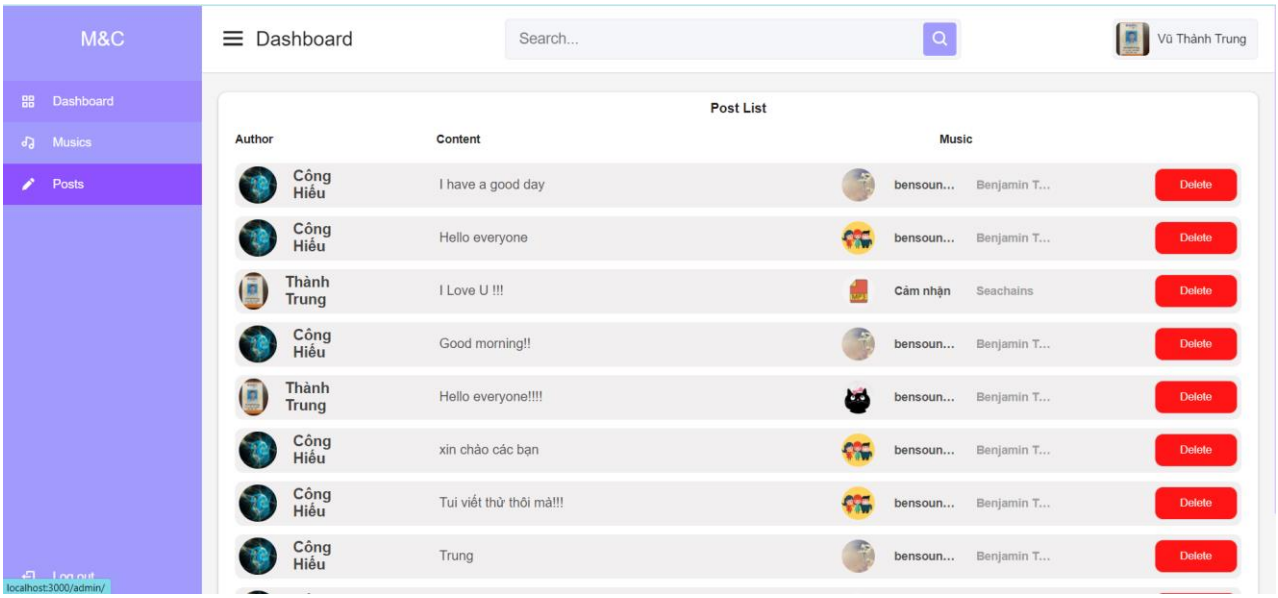

Figure 80: Admin post page screenshots

```
router.get("/datapost", async(req, res) => {
    try {
        const posts = await Post.find(req).populate("music").populate("user");
        res.json({ success: true, posts });
    } catch (error) {
        console.log(error);
        res.status(500).json({ success: false, message: "Internal server error" });
    }
});
module.exports = router;
```

Figure 81 GET post at Backend

```
export default function PostPage() {
  // get global data by useContext
  const {
    postState: { posts, postsLoading },
    getPosts,
  } = useContext(PostContext);

  // start get all posts
  useEffect(() => getPosts(), []);

  if (postsLoading) {
    return (
      <div style={{ width: "100%", height: "100vh", position: "absolute" }}>
        <Ripple
          style={{
            top: "50%",
            left: "50%",
            position: "relative",
            transform: "translate(-50%, -50%)",
          }}
          color="#be97e8"
        />{" "}
        ;{" "}
      </div>
    );
  } else {
    return (
      <div className="post-content menu-bar">
        <div className="overview-boxes">
          <div className="post__items">
            <h3 className="title">Post List</h3>
            <div className="list">
              <h4 className="list__name">Author</h4>
              <h4 className="list__content">Content</h4>
              <h4 className="list__music">Music</h4>
            </div>

            <div className="list__post-admin">
              {posts.map((post) => (
                <PostAdmin post={post} />
              ))}
            </div>
          </div>
        </div>
      </div>
    );
  }
}
```

Figure 82: Render post at Frontend

I also use hook like above and get data through useContext hook. In the userContext of the post I got the post data from the Backend through the API. Then I render a different post for each child post component.

7.11. Home page screenshots:

This page will be the main page of my website once the user is logged in with the correct account they will be redirected here. I design in a minimalist direction so every button that appears on the website carries the most basic function of my website. Once the user has successfully logged in,

the header of the page will display their profile picture and name. On the header bar, there will be a search box for the user. The left side of the page will be the general menu of the page to help users switch back and forth between pages easily and quickly. In the center of the page will be a section displaying user posts, here will display all the posts of all other users on the site. Users will be able to see other people's posts, see what message they want to share or how they feel in life with a song below. The song will be displayed with an image, song title, composer, and a button to add to your favorites if the user finds this song good. On the right will be a box showing the current song that is playing and users can play music, stop and skip songs. The user clicks on a song, this box will update the information of that song on the interface.

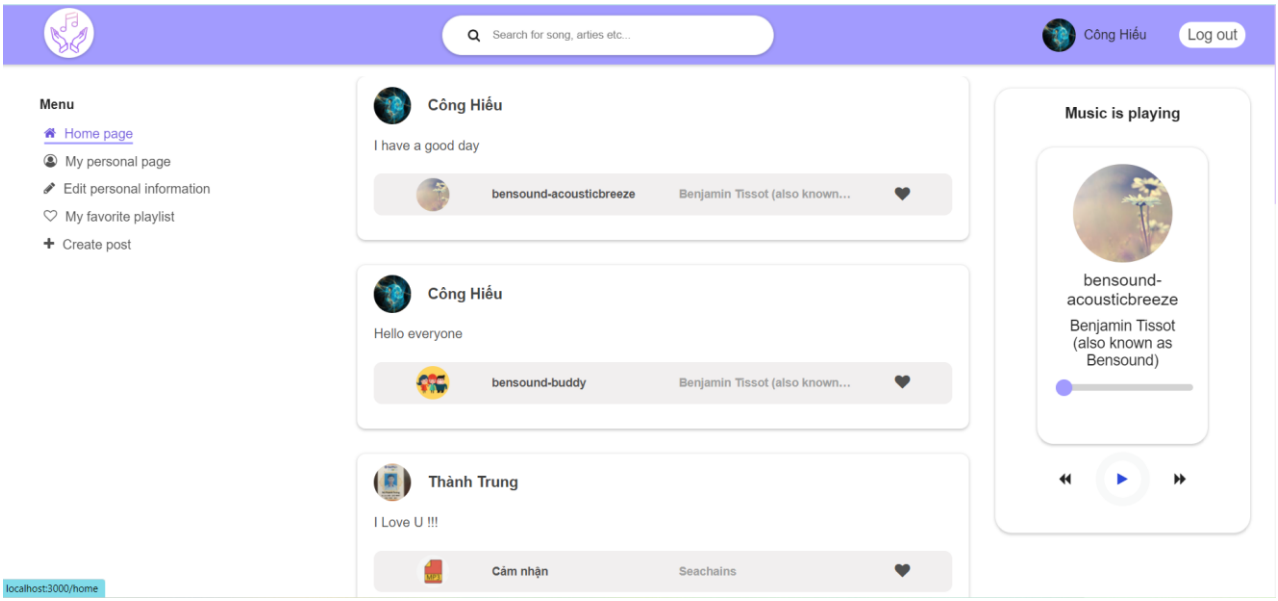

Figure 83: Home page screenshots

```
export default function PostDisplay() {
  // get global data by useContext
  const {
    postState: { posts, postsLoading },
    getPosts,
  } = useContext(PostContext);

  // start get all posts
  useEffect(() => getPosts(), []);

  // // comment function
  // // get global data by useContext
  // const {
  //   commentState: { comments },
  //   getComments,
  // } = useContext(CommentContext);

  // // start get all cmts
  // useEffect(() => getComments(), []);

  if (postsLoading) {
    return (
      <div style={{ width: "100%", height: "100vh", position: "absolute" }}>
        <Ripple
          style={{
            top: "50%",
            left: "50%",
            position: "relative",
            transform: "translate(-50%, -50%)",
          }}
          color="#be97e8"      Vu Thanh Trung, 3 days ago • show post at home pa
        />{" "}
        ;{" "}
      </div>
    );
  } else {
    return (
      <div className="postdisplay post-list post-list-0 ">
        {posts.map((post, i) => (
          <PostItems post={post} key={i} />
        ))}
      </div>
    );
  }
}
```

Figure 84; Render potst at Home page Frontend

Same as render post in admin post page but I just change render place with different interface and same post data. The data will be taken from the same post table in the database so it will be synchronized.

7.12. My personal page screenshot:

After the user presses the "my personal page" button, they will be redirected to a page showing all the posts that the user has ever posted to the site. This page helps users to check their posts. The music player box on the right will always be displayed as a menu because it helps users to listen to music on any page. They can comfortably listen to music while doing other things on the website.

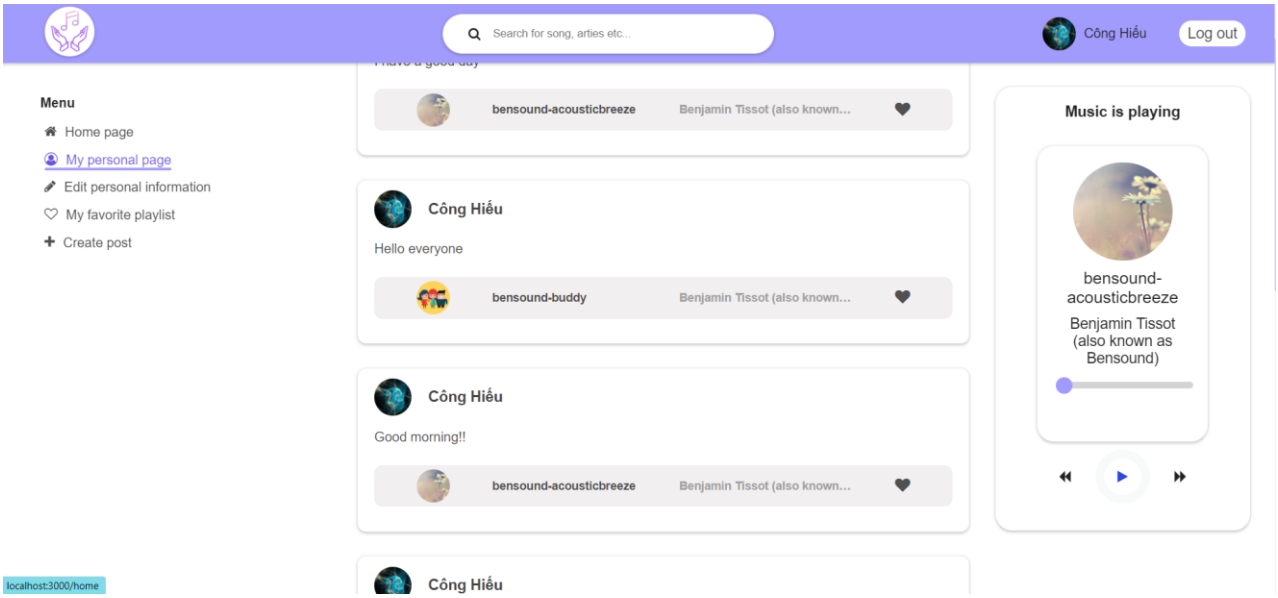

Figure 85: My personal page screenshots

```
export default function PostPerson() {
  // get global data by useContext
  const {
    postState: { posts, postsLoading },
    getPosts,
  } = useContext(PostContext);
  const {
    authState: {
      user: { _id: userId },
    },
  } = useContext(AuthContext);

  // start get all posts
  useEffect(() => getPosts(), []);
  console.log(userId);

  // check id for render post data
  const checkIdpost = posts.filter((post) => {
    return post.user._id === userId;
  });

  if (postsLoading) {
    return (
      <div style={{ width: "100%", height: "100vh", position: "absolute" }}>
        <Ripple
          style={{
            top: "50%",
            left: "50%",
            position: "relative",
            transform: "translate(-50%, -50%)",
          }}
          color="#be97e8"
        />{" "}
        ;{" "}
      </div>
    );
  } else {
    return (
      <div className="postperson post-list post-list-1">
        {checkIdpost.map((post) => (
          <PostItems post={post} />
        ))}
      </div>
    );
  }
}
```

Figure 86; Render post at My personal page Frontend

Same as the render functions above but since this is this user's personal post page, I will filter the post through the userId contained in the post with the current user userId to properly render their post. To avoid confusion with other user posts on this site.

7.13. Edit personal information screenshot:

After the user presses the "edit personal information" button they will be redirected to the page to edit their personal information. The middle of the page will display an information entry form for users if they want to change some personal information. When they successfully change the information here, all the personal data they change will be updated throughout the system.

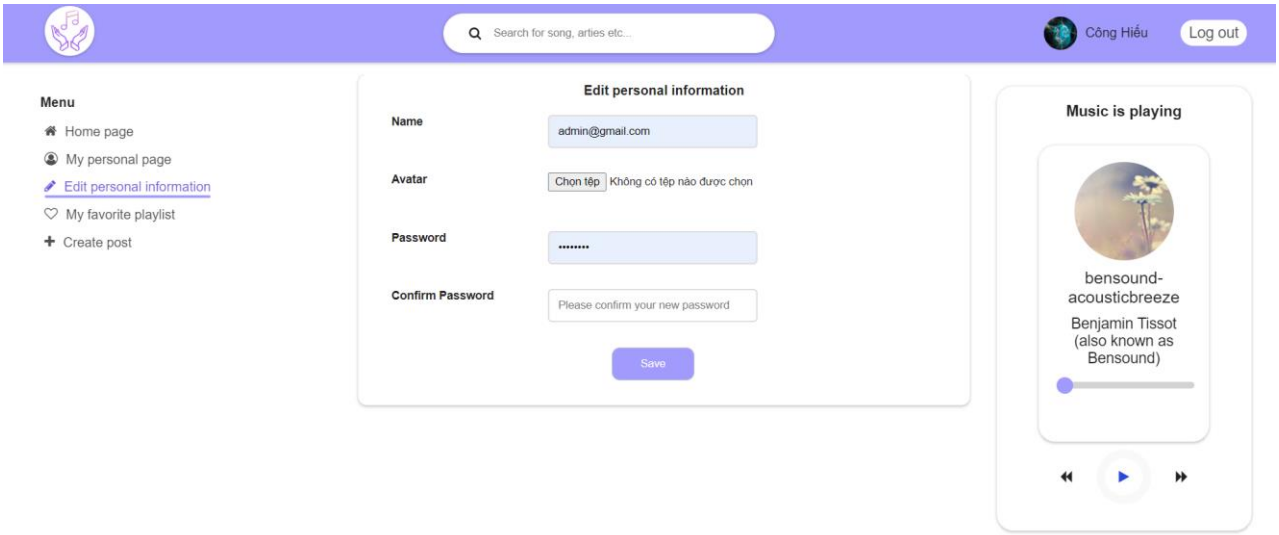

Figure 87: Edit personal information screenshots

If the user changes the password but they confirm the password does not match, they will be notified to check and re-enter the correct information.

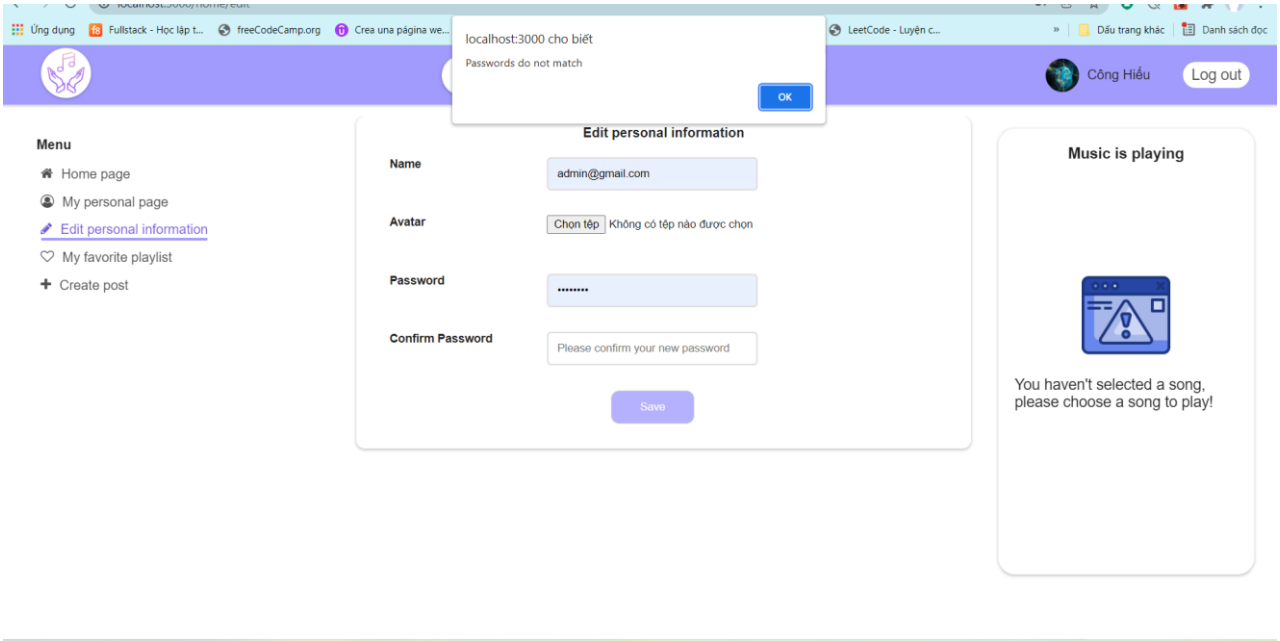

Figure 88: Edit personal information screenshots

```
router.put(
    "/:id",
    upload.single("userAvatar"),      Vu Thanh Trung, a month ago • update infomation's u
    verifyToken,
    async(req, res) => {
        const { userName, userAvatar, userPassword } = req.body;

        try {
            const hashedPassword = await argon2.hash(userPassword);
            const avatar = req.file.path;
            let updatedUser = {
                userName,
                userAvatar: avatar.replace("uploadFile\\", ""),
                // : avatar.replace("uploadFile\\", ""),
                userPassword: hashedPassword,
            };
            // check if this user is in the database or not. If so, it must match the logi
            const postUpdateCondition = { _id: req.userId };

            updatedUser = await User.findOneAndUpdate(
                postUpdateCondition,
                updatedUser, { new: true }
            );

            // User not authorised to update user not found
            if (!updatedUser)
                return res.status(401).json({
                    success: false,
                    message: "User not found or user not authorised",
                });

            res.json({
                success: true,
                message: "Excellent progress!",
                post: updatedUser,
            });
        } catch (error) {
            console.log(error);
            res
                .status(500)
                .json({ success: false, message: "Internal server error123!!!!" });
        }
    }
);

module.exports = router;
```

Figure 89 PUT information at Backend

To implement the update function, I had to check the id of the first user to get the correct user to edit through the id that the user logged in. Then I use Expressjs PUT protocol for updating user information.

```
    event.preventDefault();

    // initialize formdata to store values in state and assign those values to name in inpu
    const formData = new FormData();
    formData.append("userName", updatedUser.userName);
    formData.append("userPassword", updatedUser.userPassword);
    formData.append("confirmPassword", updatedUser.confirmPassword);
    formData.append("userAvatar", updatedUser.userAvatar);

    // get userID in local storage
    const decoded = jwt_decode(localStorage[LOCAL_STORAGE_TOKEN_NAME]);
    if (updatedUser.userPassword == updatedUser.confirmPassword) {
      axios
        .put(`${apiUrl}/update/user/${decoded.userId}`, formData)
        .then((response) => {
          if (updatedUser.userPassword == updatedUser.confirmPassword) {
            console.log(response.data);
            if (response.data.success) {
              dispatch({ type: UPDATE_USER, payload: response.data.user });
              return response.data;       Vu Thanh Trung, a month ago • render category musi
            }
          }
          alert("Passwords do not match");
        })
        .catch((error) => {
          return error.response
            ? error.response
            : { success: false, message: "Server error" };
        });
    } else {
      alert("Passwords do not match");
    }
  };
```

Figure 90: PUT information at Frontend

After getting the data from the form that the user entered and checking that the password that the user wants to update matches, then I will call the API to the Backend and send the data back to the Backend to process the data and update the database.

7.14. My favorite page screenshots:

Users press the "my favorite playlist" button in the menu on the left side of the screen, then the user will be redirected to a page displaying a list of their favorite songs after they have pressed the favorite button in addition to other people's posts. off the homepage. Here users can cancel their favorite song by clicking again on the favorite icon at the end of each song.

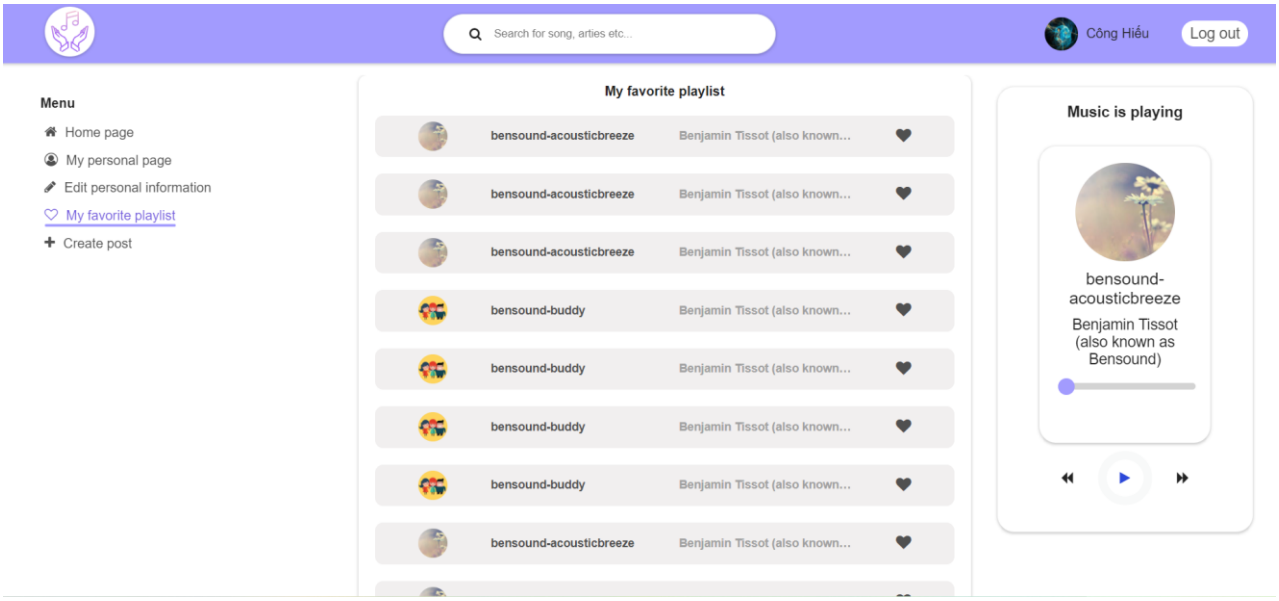

Figure 91: My favorite page screenshots

```
// @access Private
router.post("/", uploadFavorite.array(), async(req, res) => {
    try {
        // get data user by id's user at collection users
        const userData = await User.findById(req.body.user);
        // get data music by id's music at collection musics
        const musicData = await Music.findById(req.body.music);

        const newFavorite = new Favorite({
            user: u const musicData: any
            music: musicData,
        });
        await newFavorite.save();

        res.json({
            success: true,
            message: "Favorite created successfully",
            newFavorite,
        });
    } catch (err) {
        console.log(error);
        res
            .status(500)
            .json({ success: false, message: "Internal server error !!!!" });
    }
});

// @route GET api/favorites
// @desc Get favorites
// @access Private
router.get("/datafavorites", async(req, res) => {
    try {
        const favorites = await Favorite.find(req)
            .populate("music")
            .populate("user");
        res.json({ success: true, favorites });
    } catch (error) {
        console.log(error);
        res.status(500).json({ success: false, message: "Internal server error" });
    }
});

module.exports = router;
```

Figure 92: Favorite function at Backend

This is the code I create an API for importing data of my favorite music into the database and getting it out in the database through 2 protocols, GET and POST.

```
    musicPlayed.style.display = "block";
  };
  const { dispatch } = useContext(FavoriteContext);
  const clickFavorite = (e) => {
    getMusicSelected.bind(this, _id);
    const formData = new FormData();
    formData.append("user", userId);
    formData.append("music", _id);
    console.log(userId, _id);
    axios
      .post(`${apiUrl}/favorites`, formData)
      .then((response) => {
        if (response.data.success) {
          dispatch({ type: ADD_FAVORITE, payload: response.data.favorite });

          return response.data;
        }
        if (!response.success) {
          alert(response.data.message);
        }
      })
      .catch((error) => {
        if (error.response) return error.response;
        else return { success: false, message: error.message };
      });
  };

  return (
```

Figure 93 Favorites function at Frontend

This is the function I handle when the user clicks on the favorite icon of each post to add music to the favorites. The data that will be taken is the id of the music in that post and the id of the user who made that click. Then the data will be sent back to the Backend for use and saved in the database.

```
export default function PostFavorite() {
  // get global data by useContext
  const {
    favoriteState: { favorites, favoritesLoading },
    getFavorites,
  } = useContext(FavoriteContext);
  const {
    authState: {
      user: { _id: userId },
    },
  } = useContext(AuthContext);

  // start get all favorites
  useEffect(() => getFavorites(), []);
  console.log(userId);
  console.log(favorites);

  // check id for render favorites data
  const checkIdFavorite = favorites.filter((favorite) => {
    return favorite.user._id === userId;
  });

  console.log(checkIdFavorite);

  //check data uploaded for render
```

Figure 94: Render favorite at My favorite page Frontend

I would then render the data of the user's favorite tracks into their favorites page. I will also check the correct user id and then properly render their favorite music here.

7.15. Create post page screenshots:

The user can access the article by clicking the create post button in the left menu. On this page, the user can select the song and create a new post here with that song after pressing the create button in each song. Or they can download the music if they like.

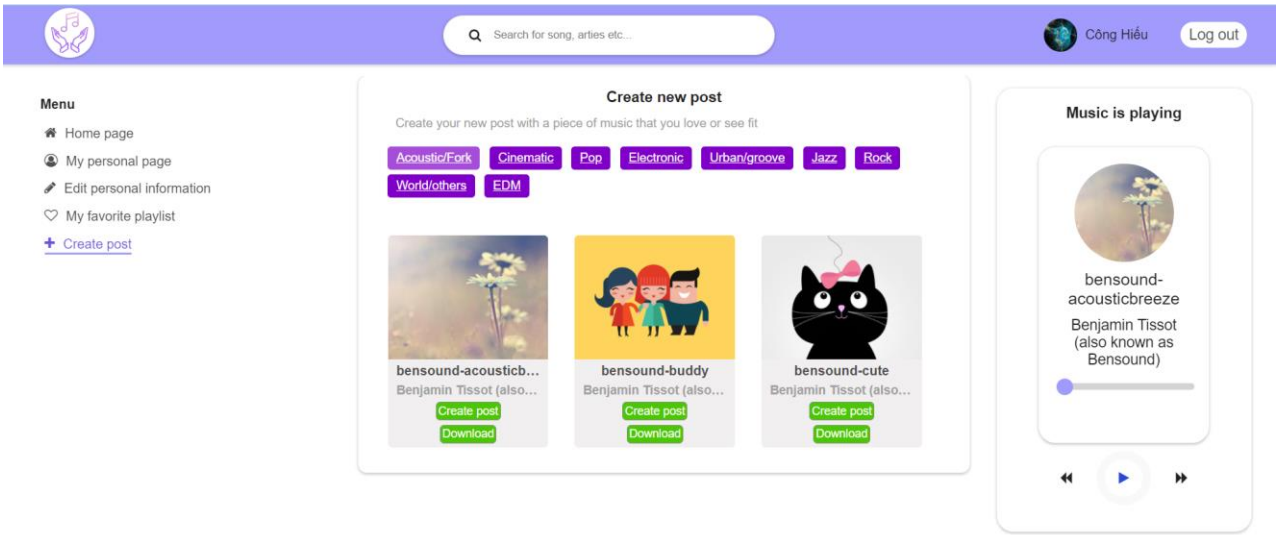

Figure 95: Create post screenshots

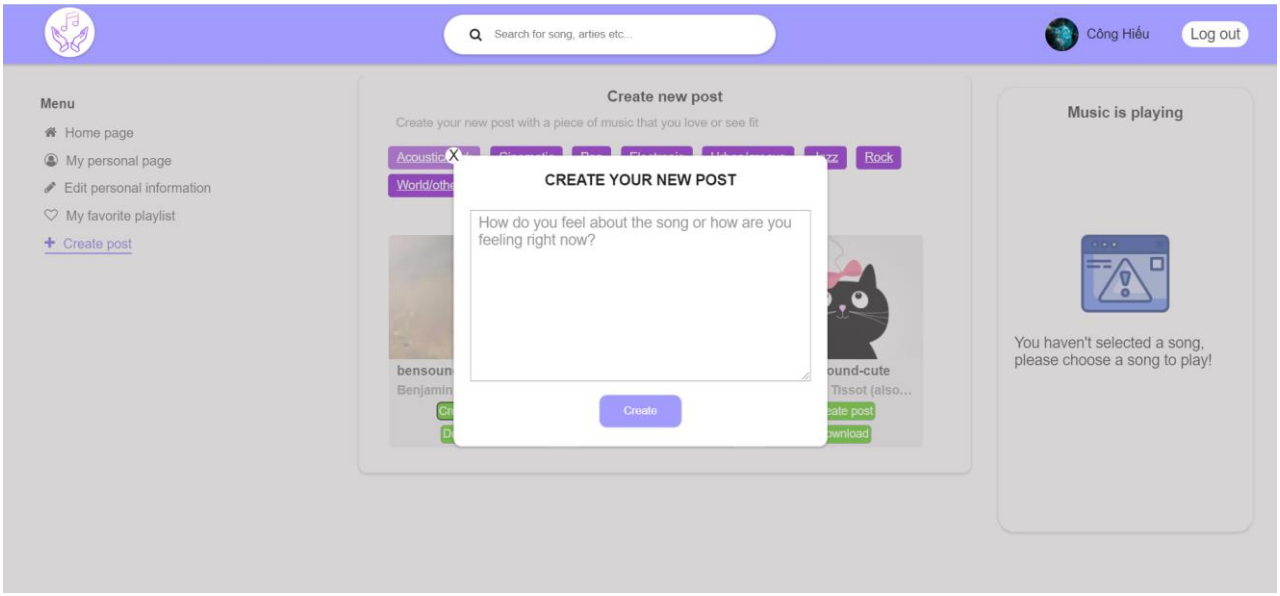

Figure 96: Create post form screenshots

```
export default function Acoustic() {
  const {
    musicState: { music, musics, musicsLoading },
    getMusics,
  } = useContext(MusicContext);        Vu Thanh Trung, a week ago • full all mus
  // start get all musics
  useEffect(() => getMusics(), []);

  const getCategory = musics.filter((music) => {
    return music.musicCategory == "acoustic";
  });

  if (musicsLoading) {
    return (
      <div style={{ width: "100%", height: "100vh", position: "absolute" }}>
        <Ripple
          style={{
            top: "50%",
            left: "50%",
            position: "relative",
            transform: "translate(-50%, -50%)",
          }}
          color="#be97e8"
        />{" "}
        ;{" "}
      </div>
    );
  } else {
    return (
      <div className="category__music acoustic">
        {getCategory.map((music, i) => (
          <ListMusic music={music} key={i} />
        ))}
      </div>
    );
  }
}
```

Figure 97: Render music at Create page Frontend

Here, after getting the API to get data from the Backend, I just need to render the songs that are stored in the database.

```
// get data as user input
const onChangePost = (event) => {
  setinputPost({ ...inputPost, [event.target.name]: event.target.value });
};

const createPost = (e) => {
  e.preventDefault();
  console.log(inputPost.music);
  // initialize formdata to store values in state and assign those values to name in input
  const formData = new FormData();
  formData.append("user", inputPost.user);
  formData.append("postContent", inputPost.postContent);
  formData.append("music", getMusicId);
  console.log(formData);
  axios
    .post(`${apiUrl}/posts`, formData)
    .then((response) => {
      if (response.data.success) {
        dispatch({ type: ADD_POST, payload: response.data.post });

        return response.data;
      }
      if (!response.success) {
        alert(response.data.message);
      }
    })
    .catch((error) => {
      if (error.response) return error.response;
      else return { success: false, message: error.message };
    });
};
return (
```

Figure 98: GET data post at create post page Frontend

Here I will get the data that the user wants to post through a data entry popup. After the user completes the input and presses the create button, the data that the user enters with the id of the song that the user selects and adds the user's id will be sent to the Backend through the API. The backend will save the post data in the database.

CHAPTER 8: EVALUATION

8.1. Test plan:

STT	TEST	DATE
1	Check whether the user is admin or user	7/7/2021
2	Register	10/7/2021
3	Log in	20/7/2021
4	Add new music	25/7/201
5	Create post	29/7/2021

6	Show playlists on admin page and create post page	2/8/2021
7	Play music and stop music	6/8/2021
8	User edits information	15/8/2021
9	Log out	15/9/2021
10	Search	20/9/2021

8.2. Test case:

STT	TEST	PROBLEM	SOLUTION	TEST RESULT	DATE
1	Check whether the user is admin or user	Still not authenticated between admin and user via roleId	I have added condition to check user roleId value then categorize	PASS	7/7/2021
2	Register	Can't get data from client	I have rechecked the API and revised the API	PASS	10/7/2021
3	Log in	Enter the information but nothing happens after that	I added the redirect function to make the page redirect to another page after the login is successful	PASS	20/7/2021
4	Add new music	Do not receive data as mp3 files	I used multer for sever to help read and save data as mp3 files	PASS	25/7/201
5	Create post	I can't send data from client to server	I have checked and fixed the error in the API's protocol	PASS	29/7/2021

6	Show playlists on admin page and create post page	I can't display the right music for each different music page	Set the condition to check the type of music stored in the database then render the correct song with the type of music	PASS	2/8/2021
7	Play music and stop music	Some problems still arise when pressing the play button or stopping the music	I still can't find a solution	FAIL	6/8/2021
8	User edits information	Entering data but still can't change the information	I checked the id of the user input and output fixed it	PASS	15/8/2021
9	Log out	Still can't work due to missing a piece of code	I was missing remove the accessToken when the logout button was pressed	PASS	15/9/2021
10	Search	I can't find information	I'm still trying to fix the problem	FAIL	20/9/2021

8.3. Overall Evaluation:

The final overview reviews the functions that work and do not work in my website.

- Function can work:
  - Login and register
  - Add, edit, delete, view music
  - Add, delete, view posts
  - Play, stop, download music
  - Add music to favorites.
  - Edit user information

- Log out

- The function is not working:
  - Search
  - Cut music
  - Notify
  - Switch music
  - Comment

CHAPTER 9: CONCLUTION

9.1. Product achievement:

After completing the project, I personally feel very grateful to Mr. Thanh as well as the lecturers and others who helped me to successfully complete the project with the original goals. To start building the project I had to research and find out a lot of information about the entertainment social networking site. I had to consult and research other social networking sites available on the market today to get ideas for the design of the website. I have also been able to apply the knowledge learned from previous subjects into lessons such as designing user interfaces so that it is convenient for users. As well as designing diagrams such as UML, ERD, Sequence, Activity for your project.

Through this project, I also learned about two new and trending frameworks, Reactjs and Nodejs. I learned how to design web interfaces with components and their communication. With Nodejs I learned more about how to create a complete API for web protocols. And how to connect the two frameworks together. Learn how to get data from the database onto the web. Finally, I know a new database to store data that is quite nice and convenient, MongoDB. Learn how to manipulate data MongoDB and Nodejs data through Expressjs.

And finally, I think that through this project, my knowledge has been expanded a lot. There are valuable experiences to be drawn during the completion of the project. The difficulties that I face when managing projects alone and organizing work, dividing time to meet deadlines. Because my level is not high and I am still learning, my entertainment social networking website still has many shortcomings that I hope you and your teachers will ignore.

9.2. Development orientation in the future:

In the future I will continue to develop more for this project because I see that the project still has many shortcomings. Since this is the first project I have completed by myself, the project is still incomplete. I will update and develop more functions such as search, post comments, cut music, add new songs, improve website interface, add more animation effects, add user management functions more for admin.

Reference

1) hotro@cooftech.com, P. (2019). *Top 10 ứng dụng được tải nhiều nhất năm 2020*. [online] Top 10 ứng dụng được tải nhiều nhất năm 2020. Available at: https://cftsoft.com/top-10-ung-dung-duoc-tai-nhieu-nhat-nam-2020-201225122103917.html [Accessed 29 May 2021].
2) Statista (2020). *Facebook users worldwide 2020*. [online] Statista. Available at: https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/#:~:text=How%20many%20users%20does%20Facebook.
3) Social Media Marketing & Management Dashboard. (2021). *23 Important TikTok Stats Marketers Need to Know in 2021*. [online] Available at: https://blog.hootsuite.com/tiktok-stats/#:~:text=TikTok%20user%20statistics.
4) Backlinko. (2020). *How Many People Use TikTok? 80+ TikTok Statistics (2021)*. [online] Available at: https://backlinko.com/tiktok-users.
5) News, V. (n.d.). *Bắt nhịp xu hướng giải trí của giới trẻ*. [online] VietNamNet. Available at: https://ictnews.vietnamnet.vn/cuoc-song-so/bat-nhip-xu-huong-giai-tri-cua-gioi-tre-262005.html [Accessed 29 May 2021].
6) Vinayak SP (2019). *Article headline*. [online] ProBlogBooster - For Bloggers, By ProBloggers. Available at: https://www.problogbooster.com/2011/09/advantages-disadvantages-of-facebook-drawbacks-benefits-of-social-media.html.
7) LuatVietnam (2021). *Đăng video xấu, độc hại lên mạng xã hội bị xử lý thế nào?* [online] LuatVietnam. Available at: https://luatvietnam.vn/tin-phap-luat/dang-video-co-noi-dung-xau-len-mang-xa-hoi-bi-xu-ly-the-nao-230-29367-article.html [Accessed 29 May 2021].
8) Nam Kỳ Lân Blog - Thông tin bất động sản, kinh doanh đỉnh cao. (2020). *Tik Tok là gì? Lịch sử, ưu nhược điểm, cách đăng ký tài khoản - Nam Kỳ Lân*. [online] Available at: https://namkylan.com/kien-thuc/kinh-doanh/tik-tok-la-gi/ [Accessed 29 May 2021].
9) Nghĩa, T. (2020). *Spoon Radio là gì? Ưu và nhược điểm của mạng xã hội âm thanh này*. [online] AgencyVN. Available at: https://agencyvn.com/spoon-radio-la-gi [Accessed 29 May 2021].
10) Báo Công an nhân dân điện tử. (n.d.). *Ngăn chặn "rác" độc trên không gian mạng: Quyết liệt xử lý*. [online] Available at: http://cand.com.vn/Van-de-hom-nay-thoi-su/Ngan-chan-rac-doc-tren-khong-gian-mang-Quyet-liet-xu-ly-619823/ [Accessed 29 May 2021].

11) NordicCoder. (2018). *Vì sao ReactJS đang ngày một phổ biến?* [online] Available at: https://nordiccoder.com/blog/vi-sao-reactjs-dang-ngay-mot-pho-bien/.
12) CodeHub. (n.d.). *Vì sao React.js đang ngày một phổ biến?* [online] Available at: https://www.codehub.com.vn/Vi-sao-React-js-dang-ngay-mot-pho-bien [Accessed 29 May 2021].
13) techmaster.vn. (n.d.). *10 ưu điểm vượt trội của NodeJS so với các ngôn ngữ backend khác*. [online] Available at: https://techmaster.vn/posts/36450/10-uu-diem-vuot-troi-cua-nodejs-so-voi-cac-ngon-ngu-backend-khac [Accessed 29 May 2021].

14) (Drahošová, M. and Balco, P., 2021. *The analysis of advantages and disadvantages of use of social media in European Union*.

15) Etactics | Revenue Cycle Software. 2021. 40+ Frightening Social Media and Mental Health Statistics — Etactics. [ONLINE] Available at: https://etactics.com/blog/social-media-and-mental-health-statistics. [Accessed 29 November 2021].

16) Mitch de Klein . 2021. 8 reasons why music is important to us — Mitch de Klein . [ONLINE] Available at: https://www.mitchdeklein.com/blog/201688-reasons-why-music-is-important-to-us. [Accessed 29 November 2021].

17) GeeksforGeeks. 2021. What is the difference between CSS and SCSS ? - GeeksforGeeks. [ONLINE] Available at: https://www.geeksforgeeks.org/what-is-the-difference-between-css-and-scss/. [Accessed 29 November 2021].