UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Development of web audio app "Music equalizer"

Author:
Maxym VOLOSKYI

Supervisor:
Antonio PANTSERNO

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2020

Declaration of Authorship

I, Maxym VOLOSKYI, declare that this thesis titled, "Development of web audio app "Music equalizer"" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Development of web audio app "Music equalizer"

by Maxym VOLOSKYI

Abstract

The purpose of this bachelor's thesis is to analyze the users' experience in the creation of the modern music and to build own web application that will help beginners or amateur musicians to learn how to use digital audio workstation. It intends to help users understand music deeply and grow up their skills in the composition of contemporary music.

Demonstration of my work can be found here:
Music Equalizer

Code can be found here:
Github repository

Acknowledgements

I want to thank my supervisor Antonio PANTSERNO for directing and mentoring me during the research and development process, for reviewing the jobs I have done, and all the consultations provided.

Contents

List of Figures

List of Abbreviations

ASP	**A**udio **S**ignal **P**rocessing
DSP	**D**igital **S**ignal **P**rocessing
API	**A**pplication **P**rogramming **I**nterface
REST	**RE**presentational **S**tate **T**ransfer
HTTP	**H**yper**T**ext **T**ransfer **P**rotocol
RAM	**R**andom **A**ccess **M**emory
DAW	**D**igital **A**udio **W**orkstation
DAI	**D**igital **A**udio **I**terface
CLI	**C**ommand **L**ine **I**nterface
HTML	**H**yper**T**ext **M**arkup **L**anguage
XHTML	e**X**tensible **H**yper**T**ext **M**arkup **L**anguage
XML	e**X**tensible **M**arkup **L**anguage
DOM	**D**ocument **O**bject **M**odel
DBMS	**D**data**B**ase **M**anagement **S**ystem
JSON	**J**ava**S**cript **O**bject **N**otation
USB	**U**niversal **S**rial **B**us
PC	**P**ersonal **C**omputer

Thanks God, family and friends that supported me

Chapter 1

Introduction

1.1 Problem

Well-made modern music is a comparison of music instruments and program code. If you want to achieve an exciting sound, you need to connect an instrument to the computer and to impose a selected effect on it. Sound producers and music makers use different professional desktop programs like Ableton Live, FL Studio, or LogicX and lots of various plugins for them to create good music for our ears. More of those programs are quite expensive and complicated for beginners or people who want to try something new.

1.2 Motivation

Composing is one of the crucial musical skills - and it's worth mastering. While musicians may not create a piece that deserves to stand among the next Top 10 Songs of the Year, but they will learn the basics of music composition and enjoy the process itself.

Now, musicians have many chances since they can put blank music pages and pencils aside due to music composition apps. Those apps are the present and the future of music composing.

Nowadays, more and more people realize that it is not enough to have an instrument to play some enjoyable music. I have been playing music for the last eight years, and I could see how the music world has changed. Even for acoustic sessions, people also use music apps, effects and plugins to sound better.

In this day and age, you don't need expensive rack-mounted hardware synthesizers or a powerful PC to create good music. You need to have your instrument and a laptop in front of you. And the motivation for me to design a simple application was a desire to make life easier for beginners or people who don't need lots of complex plugins but want to try some basic effects and discover how their instruments can sound in a new way. The solution we came up with is a web application that has basic effects and will let every user realize how the modern music world works.

1.3 Goals

1. Make imposing sound effects on an instrument sound more comfortable.
2. Create an audio web application that is simple and accessible for users.
3. Let users learn the basics of creating impressive modern sound.

4. Attach recommendations, so the user could be able to discover more about how to use different effects professionally.

5. Let users create and save presets of effects for different instruments.

Chapter 2

Background Information

2.1 Creating music in modern world

Music or musical art is an art of organizing musical sounds, first and foremost, in a timeline (rhythmic), pitch, and timbre scale. Almost any sound with specific acoustic characteristics that correspond to the aesthetics of a particular era and which one might play can be musical. The sources of such sound can be a human voice, musical instruments, sound generators, and more.

Over time, music and approaches to creating it have changed frequently. Over the years, people have been trying to improve the way instruments sound, the rhythm of the songs, and more. Initially, people played only on "acoustic" instruments, later electric instruments appeared, and now also electronic sound generators.

In the modern world, people no longer want to listen to a song if the sound of vocals or instruments is poor or unrhythmical. Nowadays, the human ear has already adapted to good music and longs for it. With the advent of modern technology, what once seemed impossible has become possible. For example, a person who has never been able to sing now performs well. Anyone who has not been able to play the instrument can still record the desired instrument by pressing only the keys on the keyboard or the midi controller. Creating music has become more accessible in recent years.

2.2 Digital Audio Workstation

Digital Audio Workstation is an electronic or computer system designed to record, store, edit, and play digital sound. It provides for the possibility of completing a complete cycle of work on it, from the initial recording to obtaining the finished result.

When most people mention a DAW, they're referring to a recording system - a workstation - that has several components. Computer, Digital Audio Software (the application we are developing), and Digital Audio Interface are a basic set-up to start making music.[12] All of these components are essential and affect the quality of the output. To maximize the quality of this signal, we have analyzed what exactly has to be taken into account while developing our Digital Audio Software to achieve the best results with minimal set-up. The following are the components that will affect the output:

2.2.1 Computer

The heart of your DAW system is your computer, whether Mac or PC. The computer and its subsystems (hard drives, etc.) will determine how many tracks you can record and play at once, how many plug-ins you can apply in real time, how long will the edits take, and more. [11]

2.2.2 Digital Audio Interface

Digital Audio Interface is a device that helps get sound in and out of the computer and software. It can be a built-in audio interface or an external device. Essentially your audio interface is the ′middle man′ in your recording studio that performs digital to analog and analog to digital conversion. The analog signal from your microphones and instruments is converted into a digital audio stream. [7]

2.2.3 Digital Audio Software

If your computer is the heart of your DAW system, then the software you choose is the brain! It is a program for editing audio information in a digital representation (digital sound recording). DAS is the main software component of a digital audio workstation. The functions of digital audio software may vary depending on their purpose. The simplest of them, often freely distributed, have limited audio editing capabilities and the minimum number of supported audio formats. Professional packages can include multi-track recording, support for professional sound cards, video synchronization, an extended set of codecs, a large number of effects, both internal and plug-in - plug-ins

Chapter 3

Sound Effects overview

For the voice or instrument to sound better, sound engineers impose different effects on sound. Many factors influence the sound of the music instrument: microphone, cable, sound card, and even the walls and floors of the room. To produce an excellent sound, you need to understand which aspects you need to bear in mind: noise cancellation, equalizer, or effects (compressor, delay, reverb, etc.) and many others.

However, in most cases, it is the equalizer and effects that help you achieve a presentable sound of an instrument or voice. There are several commonly used effects: compressor, reverb, delay, flanger, distortion.

3.1 Compressor

A compressor is an effect that narrows the dynamic range. That is, it continuously determines the level of the input signal, and if it exceeds the level of the specified range, then it weakens it. The compressor reduces the difference between quiet and loud sounds.

The typical controls of a compressor are:

- Threshold – this dictates the level at which the compressor will start to compress. Set high, it will just reduce the peaks. Set low, it will reduce the level of nearly everything. [19]
- Ratio – the amount that the signal above the threshold will be reduced. Typically displayed as X:1, where X is the level above threshold. So, a 2:1 ratio reduces 2db to 1db, 4:1 will reduce a 4db peak to 1 db, etc. If the ratio is infinity:1, this is known as limiting, and completely flattens any peaks. [19]
- Attack – this controls the length of time taken for the effect to reach full compression. Because all sounds are different, to make the compression less obvious and more natural, one can set the effect to transition to compression smoothly.
- Release – this is the time it takes for the signal to return to normality from full compression after it drops below the threshold. Release is tweaked for the same reasons as attack time. [19]

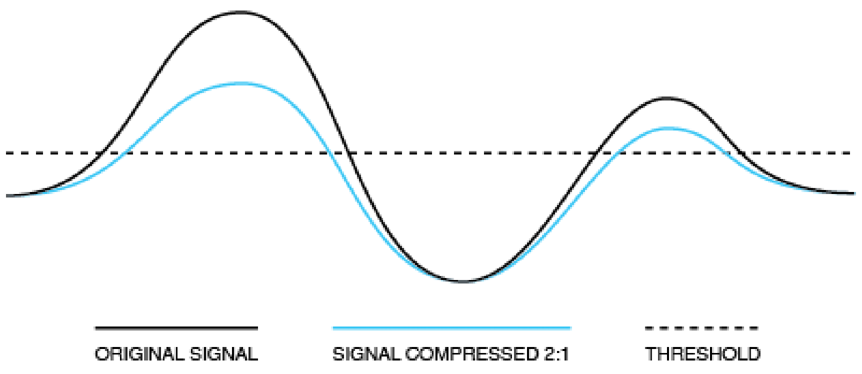


FIGURE 3.1: Compressor.

3.2 Reverb

Reverb is an effect created when any sound is in a confined space, resulting in reflection from the surfaces of the walls, which causes a large number of echoes. Subsequently, the sound slowly dampens due to the absorption of sound waves by walls and air. The effect is most noticeable when the sound source ceases to produce a sound. However, the reflections still sound; their amplitude gradually attenuates until they cease to be audible. The reflection damping time is called the reverberation time. It receives particular attention in the architectural design of large chamber rooms, which must have a specific reverb time to achieve optimum efficiency. Compared to various echoes, the sound of which is located at intervals of 50-100 ms, the reverb has thousands of echoes, located very close. [17]

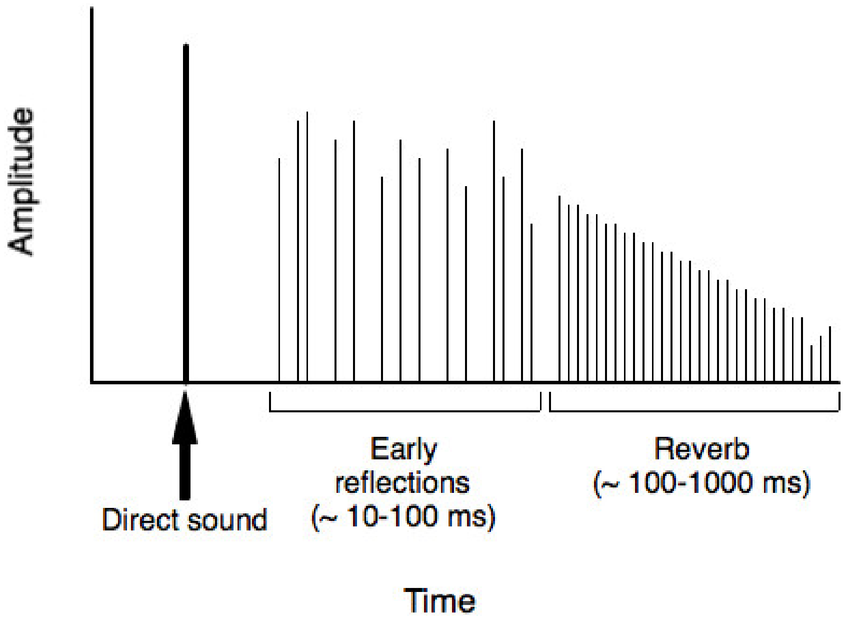


FIGURE 3.2: Reverb.

3.3 Delay

Delay is the effect of delaying sound when one or more copies of it are added to the original signal but delayed in time. The delay time is at least 50-60 ms. This effect allows you to create a rather impressive and unusual sound of an instrument or voice, as well as to select various parameters for the impeccable sound.

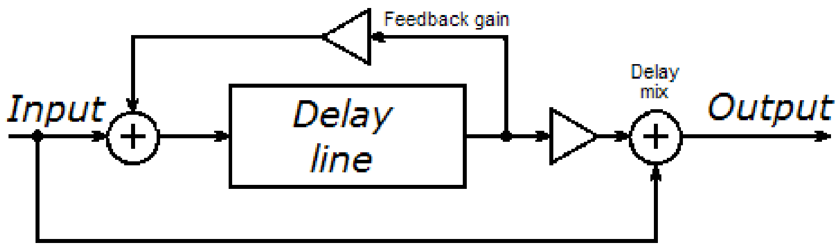


FIGURE 3.3: Delay.

3.4 Flanger

A flanger is a sound effect that occurs while playing the same audio signal when in one of the channels, this signal is produced with some delay (up to 20 ms) relative to the second. This effect is common in music (starting with The Beatles), most often in techno compositions. The effect of a flanger is recognized by its "transitions" from one channel to another, resembling the sound of a flying jet. [15]

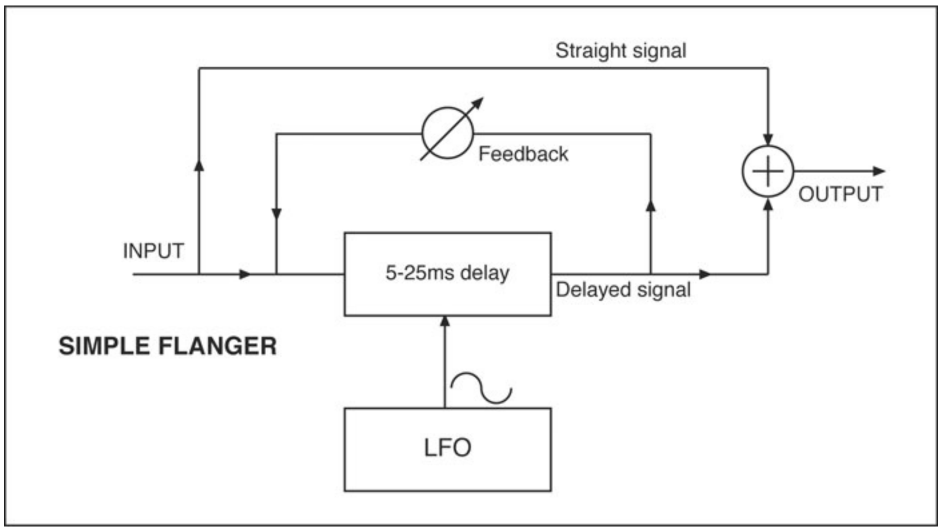


FIGURE 3.4: Flanger.

3.5 Distortion

Distortion is a sound effect achieved by distorting a signal by its “hard” limit in amplitude, or a device producing such an effect. It is most prevalent in hard rock, metal, and punk rock music genres in combination with an electric guitar, as well as in hardcore techno and especially in speedcore and breakcore with a drum machine. Sometimes this term designates a group of the same type of sound effects (overdrive, fuzz, and others) that implement non-linear signal distortion. They are also called "overload" effects, and the corresponding devices are called "distorters."

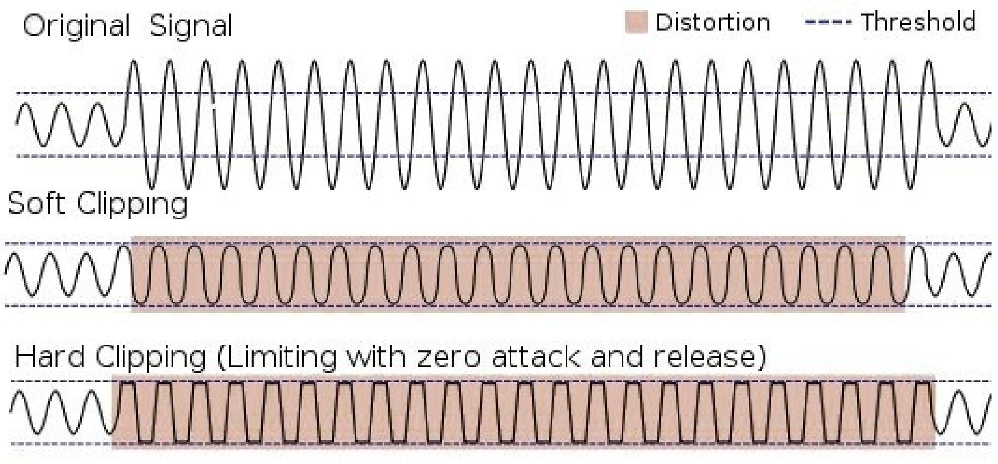


FIGURE 3.5: Distortion.

Chapter 4

Market overview

This age is called a digital era. No matter in which aspect of our lives is concerned with music, music studio computers have entirely revolutionized it. It's not very different in the industry, either. What used to take days and weeks now can be done in a few hours. The output is also very refined, and the ease of use is remarkable. It is not very easy to understand which program is the best to use, since each musician may find it convenient to work with entirely different interfaces. There is no universal pattern to calculate the best of them. Each is unique, but at times it is not very easy for beginners to master. And also - the main problem is that most DAWs exist on macOS or Windows platforms. And not always, a program supported on Windows is user-friendly enough. However, we tried to find common ground for all musicians and to reconcile their experience of using different programs and draw a conclusion from that.

We chose the four that I think are the most comfortable and most common among professionals.

4.1 Ableton Live. [1]

The most convenient and popular of all desktop DAWs is Ableton Live. The 2001 release of the first version of Ableton Live was a breakthrough in the music industry. Musicians appreciated the approach to designing the program interface, equally well suited for both sound recording and live performances. In addition to the traditional DAW, users were given another musical instrument, albeit a soft one. Today, Ableton Live is considered the standard in the music industry. Live is an ideal choice for musicians who want to create new loops and samples in a couple of clicks, control all elements of the project, and experiment with sound and effects.

FIGURE 4.1: Ableton Live 10

4.2 Logic Pro X. [4]

Musicians who choose Logic Pro X get a powerful digital workstation with a clear and not overloaded interface, ready to produce music in a wide variety of styles. Apple's workstation is very popular in the music industry and is present in almost any studio (along with Ableton or others), where there is a Mac. Together with the program, users receive a vast library of content larger than 70 GB. Inside, there are various virtual instruments (both typical and rare), synthesizers, plug-ins, loops and samples, parts of a virtual drummer, and much more. Logic Pro X is the most convenient program for beginners up to date.

FIGURE 4.2: Logic Pro X

4.3 FL Studio. [6]

FL Studio offers a slightly different approach to creating music and the principles of working with tracks - in this context, DAW is more like a software step sequencer than a regular program for music recording. Nevertheless, the possibilities of FL Studio are vast, and the workstation allows you to achieve impressive results. A full version of FL Studio offers a large selection of instruments, synthesizers, effects, and plug-ins aimed at creating electronic music. The sound quality of the individual elements is not the highest, but enough for most work tasks.

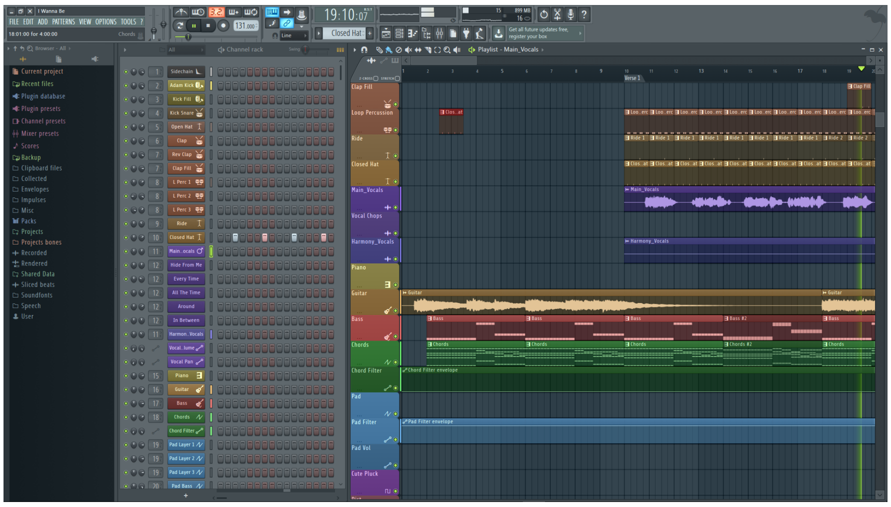


FIGURE 4.3: FL Studio

4.4 GarageBand. [3]

Unexpectedly, I included GarageBand in this list because it is a non-professional program for creating musical compositions. Yes, GarageBand is a simplified version of Logic Pro X, but you should not underestimate it. The sequencer may be lagging behind full-length stations in technical specifications, but the appearance of GarageBand is an important event in the world of amateur music. Now, any owner of an "apple" gadget can create high-quality music, regardless of their skills in working with DAW programs and the knowledge of sound engineering. It is trendy among amateurs and therefore is on our list.

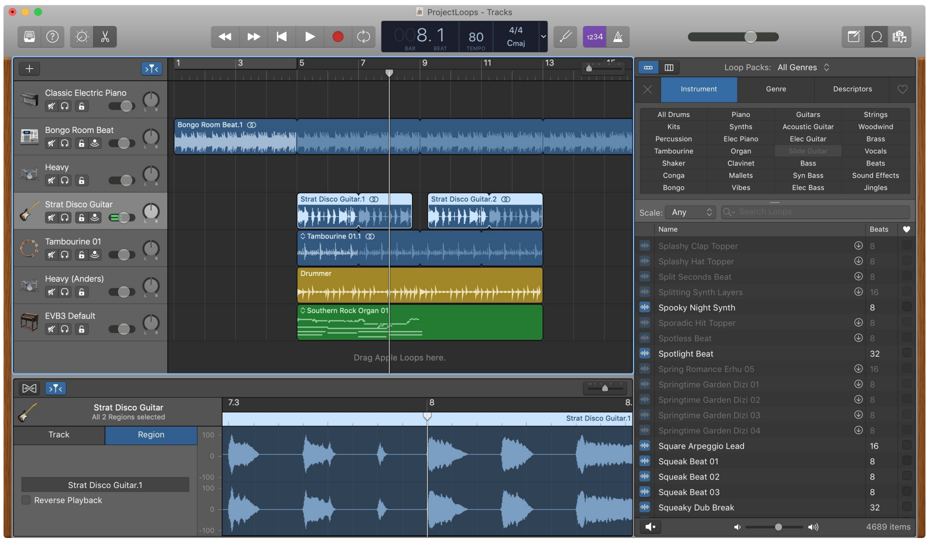


FIGURE 4.4: GarageBand

All four programs are trend-setting, fast, and deliver incredible results. But they all have several problems: it takes a long time to learn how to use the programs, takes much time to install them, and they are not available on all platforms and are not only paid but also a little expensive (excluding GarageBand - it's free).

Chapter 5

Technologies overview

Originally every web application usually is composed of three components: **Web-Service**, **Management Interface** and a **Database**.

5.1 WebService

5.1.1 REST API

There are two most popular types of Web Services:

- SOAP (Simple Object Access Protocol) is an XML-based method for exchanging structured domains, as well as for realizing wiki procedures.
- RESTful web services use open-source HTTP methods for the realization of REST architecture. RESTful web service usually identifies the resource URI (Uniform Resource Identifier), also ensures that the resource is in JSON type and a set of HTTP methods.

We chose REST because it is mostly better for the web API: it provides data in the form of resources (for example user), and not services (for example, getUser) on how SOAP works. Besides, REST inherits HTTP operations, meaning you can make simple API calls using HTTP requests to perform CRUD operations (Create, Read, Update, Detele).

RESTful API Design and Architecture Constraints:

1. **Client-Server.** The system should be divided into clients and servers. Separation of interfaces means that, for instance, there is no connection between the clients and the storage of data, which remains inside each server; therefore, the mobility of the client code improves. Servers are not associated with a user interface or state, which means they can be simpler and more scalable. Servers and clients can be replaced and developed independently until the interface changes.

2. **Stateless.** The server should not store any client information. The request must store all the necessary information for processing the request and, if necessary, customer identification.

3. **Cache.** Each response must be marked if it is cacheable or not to prevent clients from reusing obsolete or incorrect data in response to further requests.

4. **Uniform Interface.** A single interface defines the interface between clients and servers. It simplifies and separates the architecture, which allows each part to develop independently.

5.1.2 Node.js [5]

Modern platforms for creating a web server are PHP and NodeJS. Each of these platforms has its strengths and weaknesses.

The advantage of PHP is that it is a large codebase, a portable solution, and it is designed for WEB (which serves as a reason why we do not consider other languages, such as Java or Python).

But the main disadvantage of PHP is the outdated client-server model. PHP follows the classic client-server model where a page request initiates an application and its connection to a database, their processing, and HTML-rendering. It makes PHP a bit slower in comparison with the Node.js applications, launched at startup, and therefore Node is more suitable for writing real-time applications.

Therefore, NodeJS possesses an advantage in our case, because, for us, it will be a quick server solution. Node.js allows creating applications with non-blocking input/output that can handle several requests at the same time through the use of the JavaScript event queue. Another critical argument is the same language on the front-end and back-end. Having one language on the front-end and back-end is excellent for providing support for your application and for the coordination between the team members. Also, NodeJs is flexible. It does not have strict rules or hard dependencies, which leaves room for creativity when developing applications. Developers themselves can choose architecture and dependencies.

Nowadays, the Node.js platform is one of the most popular platforms for building efficient and scalable REST APIs. It is also suitable for the development of hybrid mobile applications, desktop programs, and even for IoT. Node.js uses an event-driven model and a non-blocking I / O architecture, which makes it lightweight and efficient. It is neither a framework nor a library; it is a JavaScript environment.

5.1.3 Express [10]

We chose Express because it is the most popular Node.js framework. This web framework is well supported and used by many companies in production. Adding Express to your project is easy using the npm package manager.

Express is a minimalistic and flexible web framework for Node.js applications, which provides an extensive feature set for mobile and web applications. It has many HTTP utility methods and intermediate handlers at its disposal, hence you can create a reliable API quickly and easily use it. Express provides a thin layer of the fundamental features of web applications that don't restrain you from working with the long-familiar and loved features of Node.js.

And also because there are many training guides on this framework for a better study.

5.2 Management Interface

5.2.1 JavaSript Framework

Nowadays, there are three of the most popular frameworks - React, Angular, and Vue. Each of them is good in its own right and has strengths and weaknesses. For the project, we chose a platform that is the easiest to learn and implement in the project.

Therefore, our choice immediately fell on Vue or React because Angular is more complicated to learn, and will take much more time. And since the project itself is not very large, we do not need to take Angular as a basis, as it is most useful for an extensive project, but this is not our case.

Angular [8]

The critical point why we did not choose Angular is that the quality of the documentation is quite poor. Frequently, when you open the documentation, you can see the name of the class methods, without any information about their purpose and possible ways of application. Even if the description is present, it is not always possible to understand how to use this or that method without an example.

Angular is a very sophisticated framework. On the one hand, using CLI, you can quickly start writing code on it, but, on the other hand, without studying the documentation about how the change detector works, you can write very non-optimal code or even non-working at all.

Vue.js [20]

Vue.js is a framework suitable for the creation of the adaptable user interfaces and complex single-page applications. It has enhanced HTML, which can help optimize the processing of HTML blocks using different components. Vue.js provides a quick transition period from other frameworks to Vue.js due to its similarities with Angular and React in terms of design and architecture. It also helps to develop reasonably large reusable templates that can be developed without spending a considerable amount of time, given the simple structure.

Despite all its advantages, Vue.js still has a relatively small market share compared to React or Angular. This means that the exchange of knowledge within the framework is still developing. Sometimes Vue.js may have problems integrating into huge projects, but there is still no experience in possible solutions. And the biggest drawback is the lack of full English-language documentation, which leads to some difficulties at various stages of development.

React.js [18]

For the project, there will be an expansion of the functionality in the future. We paid more attention to React due to excellent backward compatibility, and because it is the most popular of these three frameworks now.

React is a JavaScript library for creating user interfaces. Its main task is to ensure that what one can see on web pages is displayed on the screen. React considerably facilitates the creation of interfaces by breaking each page into small fragments. We call these fragments components. React simplifies the creation of interactive interfaces. All you have to do is describe how different parts of the interface look in each state of your application and React will effectively update and render only the components you need when your data changes.

The benefit of React is its comprehensibility. React is much easier for understanding due to the simplicity of its syntax. It also has a high level of flexibility and maximum responsiveness. The main advantage of React is that it has a virtual DOM (document object model), which allows you to organize documents in HTML, XHTML, or XML formats into a tree that is best suited for web browsers to analyze various elements of a web application. It also has a 100% open-source JavaScript library that receives many daily updates and improvements according to feedback from developers around the world.

5.2.2 Redux [2]

Redux is an open JavaScript library designed to manage the state of an application. It is most commonly used along with React or Angular to build user interfaces.

It helps to write applications that behave stably / predictably, work in different environments (client/server/native code), and are easily tested.

We chose Redux because it helps us separate the state of the application from React. Redux creates a global store, which is at the top level of your application and transfers status to all other components. Unlike Flux, Redux does not have multiple storage objects. The entire state of the application is inside this storage object, and we could change the view layer to another library with the storage intact.

5.2.3 Web Audio API

Web Audio API is a powerful and multifaceted tool for the manipulation of the sound component on a web page, which allows developers to select sources, add special sound effects (such as panning) to them, visualize them, and much more.

The Web Audio API involves handling audio operations inside an audio context and has been designed to allow modular routing. Audio nodes, which are linked together to form an audio routing graph, perform basic audio operations. Several sources with different types of channel layout are supported even within a single context. This modular design provides the flexibility to create complex audio functions with dynamic effects. [13]

5.2.4 Pizzicato.js [9]

Pizzicato aims to simplify the way you create and manipulate sounds via Web Audio API. This library has 15 available basic effects. We chose Pizzicato.js because it is one of the most user-friendly libraries that simplify working with the Web Audio API. It is very easy-to-use and understandable.

5.3 Database

5.3.1 MongoDB [14]

MongoDB is a cross-platform, document-oriented database that provides high performance and easy scalability. This database is built on the concept of collections and documents. The entire MongoDB system can represent not only one database located on one physical server. MongoDB functionality allows you to host multiple databases on multiple physical servers, and these databases can easily exchange data and maintain integrity.

Namely, document-oriented DBMSs are used for storing JSON-documents in “collections” and making queries on the necessary fields. You can use this database to create applications that will not contain too many links. An excellent example of such an application is an engine for a blog platform or storage of a product catalog. And since we store our data in JSON format, we chose MongoDB for this.

5.3.2 Redis [16]

Redis is an open-source, in-memory data structure store, used as a database, cache, and message broker, usually used for session management. It is a distributed repository of key-value pairs stored in RAM, with the ability to provide long-term storage at the user's request.

Such DBMSs are used when high-speed access to data is required. If you are creating an application for online trading, in which there are categories loaded on each page, then instead of accessing the database at each reading, which is extremely costly, you can store data in the cache. It allows quick read / write operations. Dandala advises using DBMSs that use the cache as a shell for processing frequently requested data, eliminating the need to make frequent queries to the database.

We added Redis to store the refresh token. It can automatically delete entries, and it works fast.

Chapter 6

Proposed Approach

Our main task is to create the most accessible solution for beginners or nonprofessional musicians, which will be most understandable for them and useful for their musical growth or leisure. It is worth noting that the proposed solution should be maximally accessible to anyone and should not require a lot of financial or other efforts. In this chapter, I will consider the unusual tool that I am developing for musicians, the functions it performs, and what the user needs to use it.

We know that our potential client is a beginner or amateur musician who does not perform on big stages. Music is not his main source of income, but it is a hobby that gives him life inspiration. They would like to evolve in this field and compose some new and inspiring music at home, with only an instrument and a laptop in front of them.

6.1 Architecture

The main parts of the architecture are Digital Audio Interface and a laptop. All analog data is going to be converted to digital through Digital Audio Interface and sent to a laptop via USB. After receiving data, the user can modify it in the app and listen to the result. The client-side is written in React with Redux usage. We can perform all manipulations with the sound due to the Pizzicato.js library that simplifies the way you create and manipulate sounds via the Web Audio API. All data after saving is sent to REST API, written in NodeJS. On the server-side, all data is stored in MongoDB and Redis.

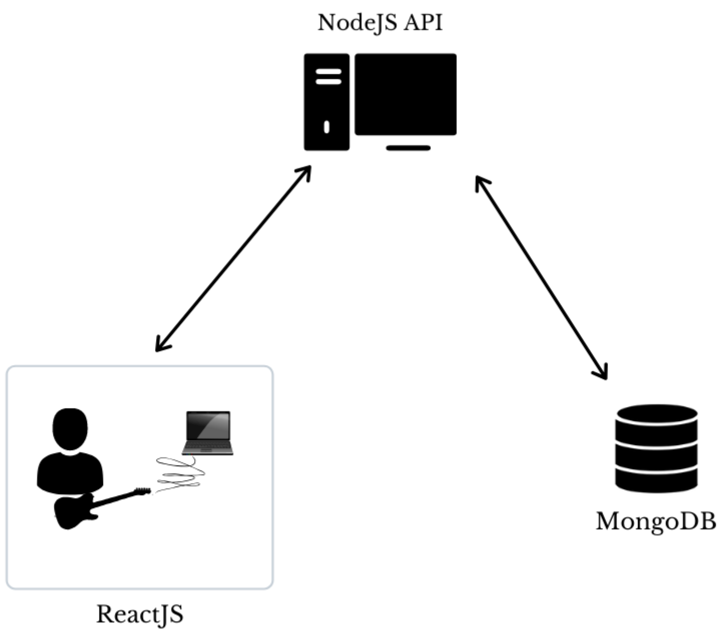


FIGURE 6.1: System Architecture.

6.2 Application

To create an application that anyone can use on any operating system, one should develop it on the web. Then everyone can access and use it. This is a single-page application that is divided vertically into three parts—Header, Equalizer, and Effects.

6.2.1 Authorization

First of all, to use our application, the user needs to register or log in. We provide a simple registration with an email. During registration, the user fills in general data (name, email, and password). (Figure 6.3)

We have a reusable form component here for the Login and Registration. FormComponent includes two components: the fields component and the button. TheRenderFormFields (the fields component) generates as many fields as needed for lo-gin or registration. This allows us to make the code clean, and instead of writing a lot of similar elements, reuse the existing universal components by substituting the desired field value in them.

Below is an example of our FormComponent code:

```
  const FormComponent = props => (
    <div className="form-body" autoComplete="off">
      <RenderFormFields
        fieldsToRender={props.fieldsToRender}
        onInputChange={props.onInputChange}
        userData={props.userData}
```

```
          validationErrors={props.validationErrors}
        />
        <div className="field">
          <Button
            onClick={props.onFormSubmit}
            className="submit"
            value={props.loading ?
                    <img src={Spinner}
                         alt="Authentication spinner"
                    /> :
                    'Submit'
                  }
            type="submit"
            disabled={props.loading ? 'disabled' : null}
          />
        </div>
      </div>
    );
```

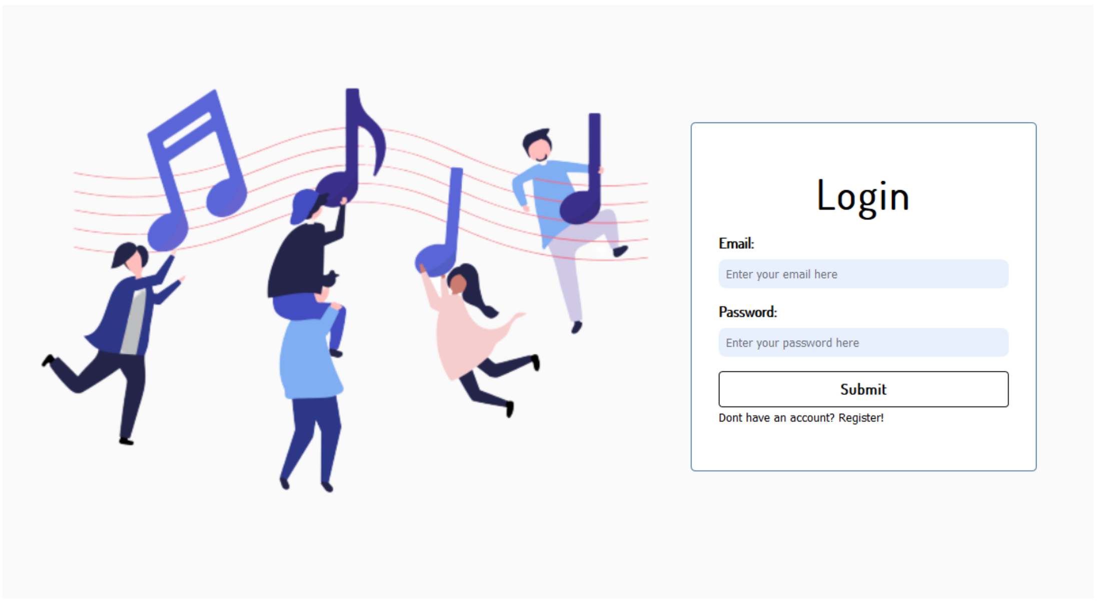


FIGURE 6.2: Authorization screen.

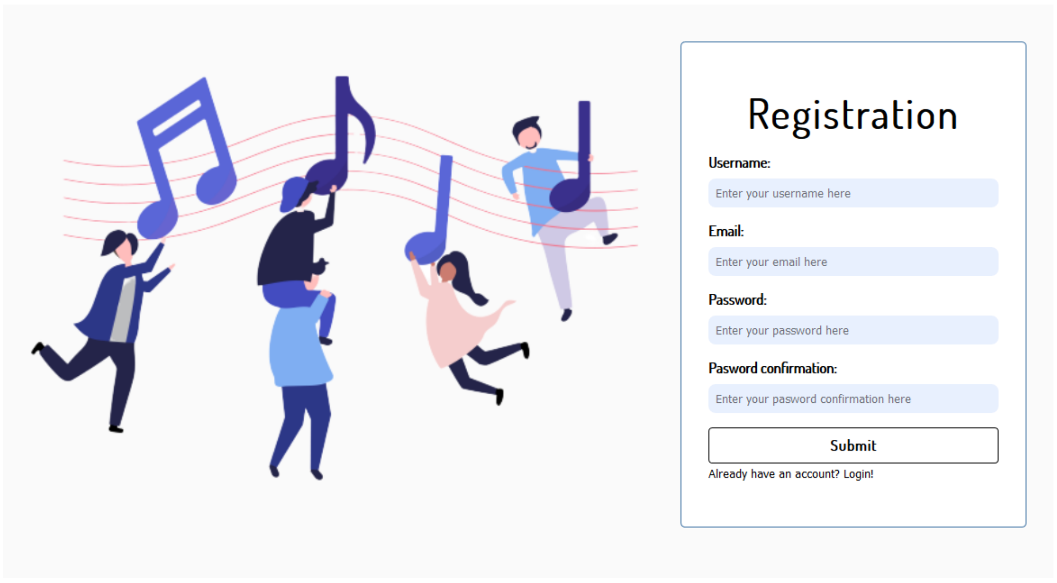


FIGURE 6.3: Registration screen.

6.2.2 Header

Header is the smallest section of the application. It contains the logo of the application, the user's nickname, and the logout button. (Figure 6.4)

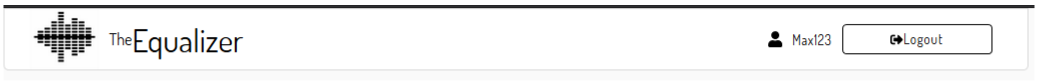


FIGURE 6.4: Header screen.

6.2.3 Equalizer

The middle part of the application contains several important parts. The first of these is the Drag and Drop area (Figure 6.5), through which we can upload the track in addition. I decided to add Drag and Drop because it allows the user to enjoy the upload feature comfortably since it saves him the unnecessary work of searching for a song in the pop-up that pops up when the user presses the upload button. Once started, an equalizer animation replaces the Drag drop area.(Figure 6.6)

Below are the buttons for controlling live audio and downloaded track.

After the buttons, there is a place where its name and duration appears. There are also two sliders: one is a song track, and the other is volume control.

The volume control is a slider not only for a song but for the entire application. When we simultaneously turn on the track and live sound, we can separately adjust the volume of the track and the volume of live sound. It allows the user to search for volume balance conveniently.

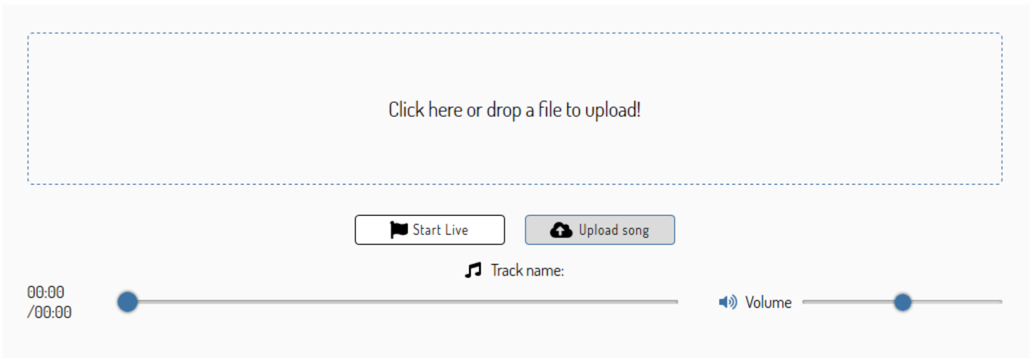


FIGURE 6.5: DragDrop screen.

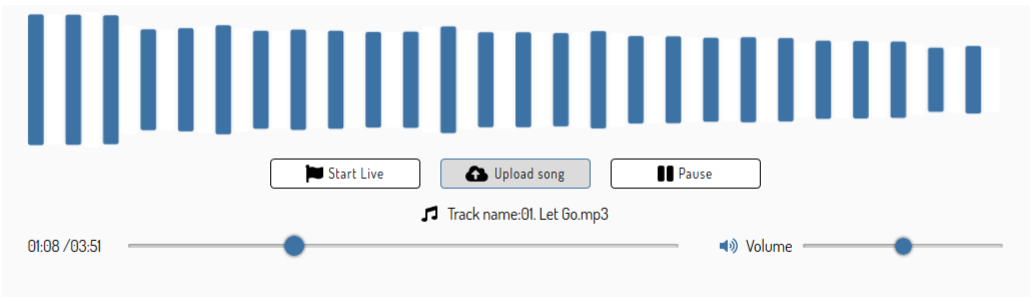


FIGURE 6.6: Equalizer screen.

6.2.4 Effects

One of the most important parts of this app is the effects. First, there are control buttons. Namely, the button for saving presets, a drop-down menu for selecting presets, and the switch of mode which the user currently uses. (Figure 6.7)

Further on the left are vertically arranged checkboxes to add or remove an effect. Here are the thirteen most commonly used effects.(Figure 6.8) On the right, the effects are displayed in blocks. In the effect block, we can see several sliders one can use to adjust.(Figure 6.9)

The block of effects consists of four components: ComponentWithSLiders, AllBlocks, BlockOfSliders, and Slider. The root component of ComponentWithSliders consists of an aside block, where we have effects checkboxes and AllBlocks - component with blocks. AllBlocks is a component to which we pass effects as props and then create several blocks (BlockOfSliders), which should be visible.

Next, we have the BlockOfSliders component. It is the reusable component that is used to visualize the effect name and the specified number of sliders that should be in this effect.

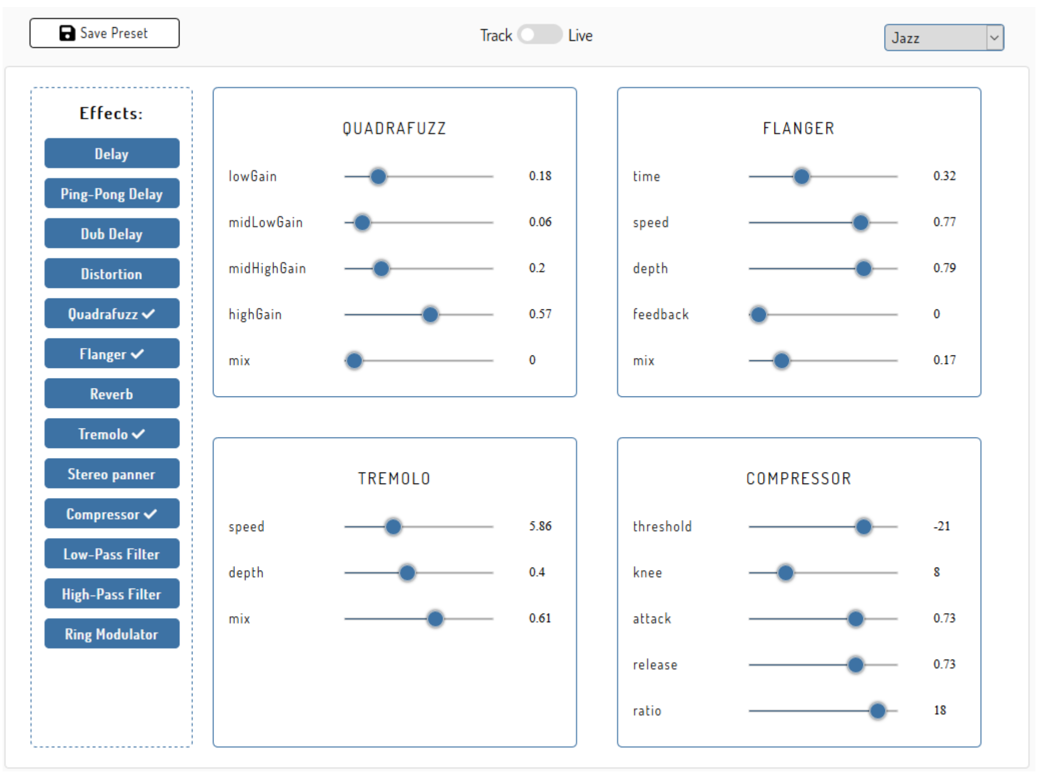
Below is the BlockOfSliders code:

```
  const BlockOfSliders = ({ name, effects }) => (
    <div className="Sliders__block">
      <p className="Sliders__block--title">
        {name}
      </p>
      <div className="Sliders__block--sliders">
        {Object.keys(effects).map(effect => (
          <OneSlider
            blockName={name}
            effectName={effect}
            effectValues={{
                          ...efValues[name][effect],
                          value: effects[effect]
                          }}
            key={effect}
          />
        ))
        }
      </div>
    </div>
  );
```

And the last reusable component is OneSlider. We reuse this component for each slider in BlockOfSliders. Inside this component, we have the setEffectsValue function, which changes the value of the slider in the Redux store for this effect. The code for this component is listed below.

```
    class OneSlider extends Component {
      constructor(props, context) {
        super(props, context);
        const {
          value, maxValue, minValue, step,
        } = props.effectValues;

        this.state = {
          sliderValue: value,
          maxValue,
          minValue,
          step,
        };
      }

      componentDidUpdate(prevProps) {
        const { value } = this.props.effectValues;
        if (value !== prevProps.effectValues.value) {
          this.setState(
            {sliderValue:parseFloat(value.toFixed(2))}
            );
        }
      }

      setEffectsValue = (sliderValue) => {
        const { blocksData, blockName, effectName } = this.props;
        blocksData.forEach(({ name, effects }) => {
          if (name === blockName) {
            effects[effectName] = parseFloat(sliderValue.toFixed(2));
            createEffect[name][effectName] = effects[effectName];
          }
        });
        this.setState(
            {sliderValue:parseFloat(sliderValue.toFixed(2))}
            );
      };

      render() {
        const {
```

```
      sliderValue, maxValue, minValue, step,
    } = this.state;

    const { effectName } = this.props;
    return (
      <div>
        <p className="Slider--label">{effectName}</p>
        <Slider
          className="Sliders--slider"
          min={minValue}
          max={maxValue}
          value={sliderValue}
          step={step}
          orientation="horizontal"
          onChange={this.setEffectsValue}
        />
        <span className="Slider--value">{sliderValue}</span>
      </div>
    );
  }
}
```



QUADRAFUZZ
lowGain 0.18
midLowGain 0.06
midHighGain 0.2
highGain 0.57
mix 0

FLANGER
time 0.32
speed 0.77
depth 0.79
feedback 0
mix 0.17

TREMOLO
speed 5.86
depth 0.4
mix 0.61

COMPRESSOR
threshold -21
knee 8
attack 0.73
release 0.73
ratio 18

FIGURE 6.7: Effects screen.

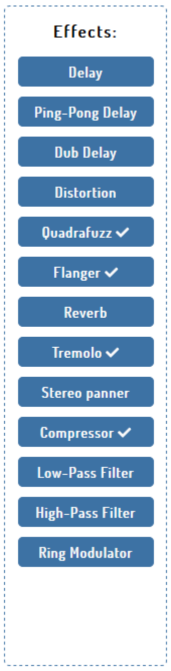
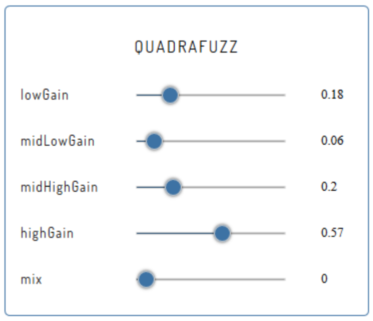


FIGURE 6.8: List of effects screen


QUADRAFUZZ

lowGain	0.18
midLowGain	0.06
midHighGain	0.2
highGain	0.57
mix	0

FIGURE 6.9: Quadrzfuzz screen.

6.3 Backend

REST API, written in NodeJS, provides access to endpoints through HTTP methods. All data is stored in MongoDB, and we use Redis to store the user's token. User endpoint allows us to get user profile information with access based on authorization token, add or remove user's presets. Gathered preset data is stored as JSON.

Chapter 7

Summary

I am sure about the accomplishment of the set goal - the creation of a web audio application that makes imposing sound effects on an instrument sound more accessible and more comfortable. The current application is user-friendly and helps learn the basics of creating an impressive modern sound. During the development of this app, we tried to take into account all the factors that could affect the original sound and tried to maximize its quality and tailor it to the user's needs so that they could use this application for their purposes.

7.0.1 Future work

There are many opportunities to improve this project, which will give more facilities to the user. Here, I will look briefly at some of them.

- **Theory and documentation page.** First of all, I want to add a helper page where users could find lots of nice and helpful materials for their growth in this area. There will be many resources divided into different levels: from beginner to professional. Also, there are going to be lots of hints and recommendations from people who are successful in music production.

- **Recording.** The second main feature I would like to implement is the opportunity to record the current live melody that the user plays.

- **Mobile App.** Now we have only a web application that works only in a browser on a laptop or PC. But there is a goal also to adapt it for the mobile version or to create a mobile application and enable users to use it with a mobile phone.

There is plenty of work to do, even though the main goal is achieved!

Bibliography

[1] Ableton. *Digital Audio Workstation*. 2016. URL: www.ableton.com.

[2] Dan Abramov. *JavaScript library*. Software available from redux.js.org. 2015. URL: https://redux.js.org.

[3] Apple. *Digital Audio Workstation*. 2004. URL: www.apple.com/mac/garageband/.

[4] Apple. *MIDI sequencer and Digital Audio Workstation*. 1993. URL: https://www.apple.com/ru/logic-pro/.

[5] Ryan Dahl. *Runtime environment*. Software available from nodejs.org. 2009. URL: https://nodejs.org.

[6] Didier Dambrin. *Digital Audio Workstation*. 1997. URL: https://www.image-line.com/flstudio/.

[7] Geoffrey Francis. *Home Recording for Beginners*. Course Technology, 2009.

[8] Google. *JavaScript framework*. Software available from angular.io. 2016. URL: https://angular.io.

[9] Alejandro M Guillén. *Web Audio Javascript library*. Software available from alemangui.github.io/pizzicato. 2016. URL: https://alemangui.github.io/pizzicato/.

[10] TJ Holowaychuk. *Web framework*. Software available from expressjs.com. 2010. URL: https://expressjs.com.

[11] *Home Recording : Everybody can record!* 2013. URL: https://rmlstudiomy.wordpress.com/2018/03/13/home-recording-everybody-can-record/.

[12] *Insider's View of the Modern DAW*. URL: https://www.sweetwater.com/feature/daw/daw_defined.php.

[13] Indrek Lasn. *All You Need to Know About the Web Audio API*. URL: https://medium.com/better-programming/all-you-need-to-know-about-the-web-audio-api-3df170559378.

[14] MongoDB. *Document-oriented database*. Software available from mongodb.com. 2009. URL: https://www.mongodb.com.

[15] Matt Piper. *What is flanger?* 2018. URL: https://www.strymon.net/flanger/.

[16] Salvatore Sanfilippo. *Data structure store, key-value database*. Software available from redis.io. 2009. URL: https://redis.io.

[17] Ric Viers. *Tte Sound Effects Bible*. Michael Wiese Productions, 2008.

[18] Jordan Walke. *JavaScript library*. Software available from reactjs.org. 2013. URL: https://reactjs.org/.

[19] *What Is A Compressor?* URL: https://www.dawsons.co.uk/blog/what-is-a-compressor.

[20] Evan You. *JavaScript framework*. Software available from vuejs.org. 2014. URL: https://vuejs.org.