Decibel Threshold Event Displayer

Analyze Audio Files and detect Noise Pollution

Dominic Gernert (gernd1)
Darius Degel (deged2)
Lukas von Allmen (vonal3)

January 17, 2025

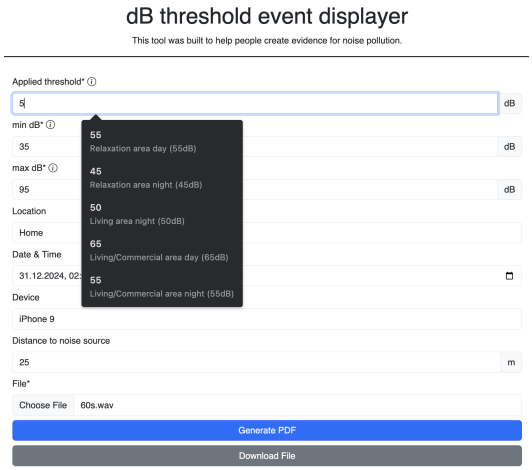
School	**Bern University of Applied Sciences**
Module	**BTI3031 - Project 1 (24/25)**
Tutor	**Dr. Simon Kramer**

Abstract

Noise pollution affects one in seven people in Switzerland, primarily caused by road traffic, railways, and air traffic, along with secondary sources such as construction sites and nightclubs. To address this issue, affected individuals must provide evidence, typically through audio recordings. This project aims to create a free, open-source, and platform-independent application, which helps individuals in documenting noise pollution.

The application processes *.wav files, filters the data by a given threshold and generates a PDF with an according plot which can be used as evidence. For a meaningful interpretation of a *.wav file, the user must provide a minimal and maximal dB(A) provided by an external tool, such as smartphone apps like DecibelX.

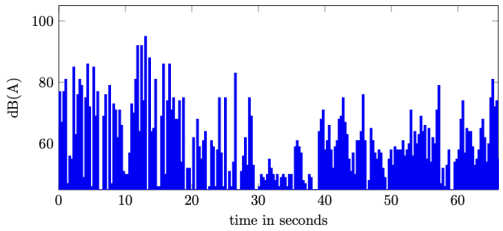
License	**GPL-3.0 licence (FLOSS)**
Application	**dB threshold event displayer**
About Page	**About Decibel Threshold Event Displayer**
Repository	**GitHub**



dB threshold event displayer - result

Recording information

Location:	Home
Date and time:	31.12.2024, 02:59:00
Device:	iPhone 9
Distance to noise source:	25m
Applied threshold:	45db(A)
Minimal dB(A):	35dB(A)
Maximal dB(A):	95db(A)
Duration:	1m 6.1s
Duration over threshold:	1m 0.1s (90.92%)
Peak:	95.00db(A)
Average:	59.86db(A)



Disclaimer:
The accuracy of measurements can vary depending on the distance to the noise source as well as the max/min dB values provided.

More information:
For more information, visit our about page:
`https://decibel-threshold-event-displayer.github.io/#/about`

Generation date: 14.01.2025, 03:02:42

Table of Contents

1 List of Tables

2 List of Figures

3 List of listings

Part I
Product Documentation

4 Introduction

4.1 Initial Situation

According to the Federal Office for the Environment (DE: BAFU) one in seven people in Switzerland is affected by noise pollution [1]. The pollution comes primarily from road traffic, followed by railways and then air traffic. In addition to these noise sources, construction sites, nightclubs and public facilities also produce noise. Because of this, Switzerland has set up upper noise limits that must be respected. However, it is of course the case that these limits are not always adhered to. Affected people must then either accept this or take action against it. For the latter, they must gather evidence to prove their noise disturbance to the police and the courts. This evidence then comes from an audio recording which the affected person made them self.

4.2 Product Goals

To help the people affected by noise pollution, we want to create an application which processes a given sound file (.wav) and analyzes it. It should detect when a specified threshold has been exceeded and then summarizes the result in a PDF document. The document should then contain all necessary information for filing a complaint. As our application will have a wide range of end users, two of our main design goals are to make it as user-friendly as possible and

to make it platform independent, so it can be used with any PC operating system and ideally mobile device.

4.3 Problems with Audio Files

A wave file (.wav) contains samples of the recorded audio, where each sample represents the amplitude at a given moment. Those amplitude values are relative to each other and not absolute. It is therefore impossible to determine the actual dB(A) (loudness relative to the human ear) someone would perceive without any further information [2–4]. Because of this, we require the user to also give information about the minimal and maximal dB(A) measured in the given audio file. To do this, we recommend using a smartphone app like DecibelX for IOS, which allows the user to record the audio and also conveniently inspect the minimal and maximal dB measured in that recording. With those two values, we can then map the relative values from the wave file to its db(A) values.

5 Specification

The following chapter describes the system specification. The specification is derived from the requirements in the project description as well as the discussions with the stakeholder. Assumptions and constraints are described in the following sections and were validated by the defined product owner and the stakeholder.

5.1 System Delimitation

The system delimitation is split into the static system environment (5.1.1) and the dynamic process environment (5.1.2).

5.1.1 System Environment (statics)

The static system environment is split into the three contexts *System*, *System context*, and *Out of scope*. *System* includes the application as well as any software dependencies. The *System context* includes people or objects which have an influence on the application. *Out of scope* describes particularly what could have an influence on the application but has been deliberately excluded by the project team (see5.2.2).

System:

- Frontend (User interaction)
- LaTeX and pgfplots
- WAV file analyser

System context:

- Stakeholder (tutor)
- User
- Noise producers
- Lärmliga
- Recording Device

Out of scope:

- Integration in legal complaints

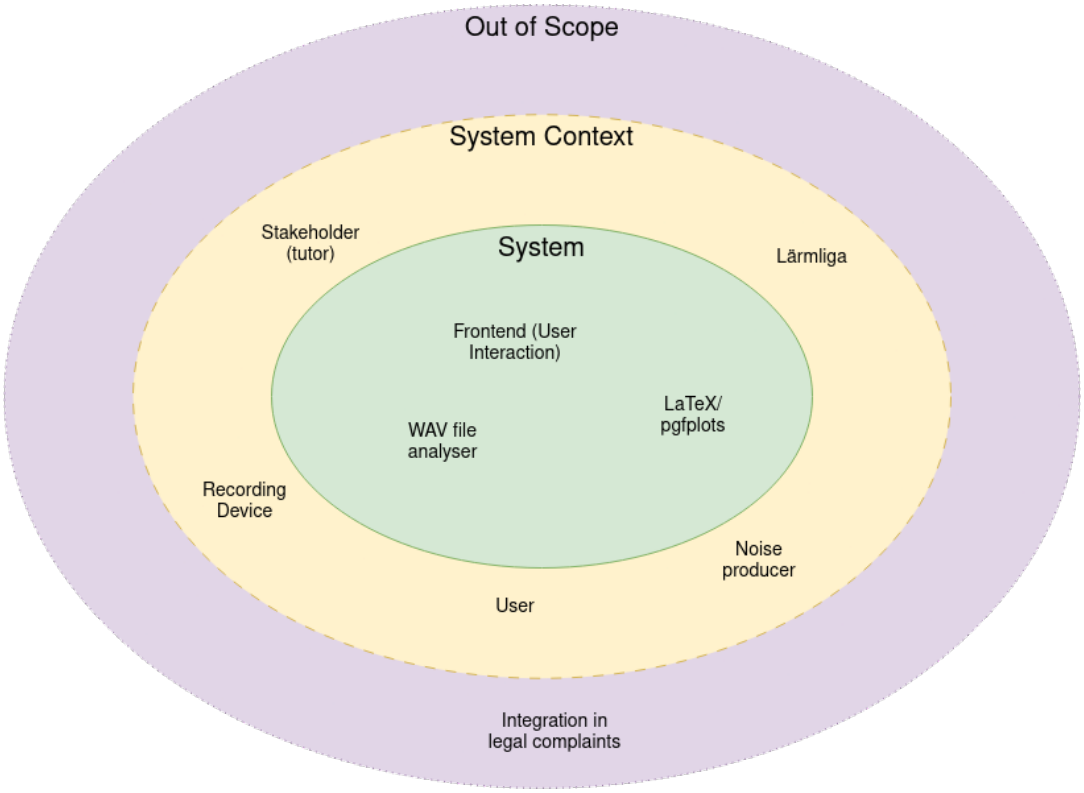


Figure 1: System Environment

5.1.2 Process Environment (dynamics)

As in the previous chapter (5.1.1), the process environment is also split into *System*, *System context*, and *Out of scope*.

System:

- Selecting (uploading) a WAV file
- WAV file analysis (validation, parsing, conversion to absolute db values, threshold filtering)
- Plotting analysis result (LaTeX / pgfplots) and PDF generation

System context:

- Recording noise
- Producing noise

Out of scope:

- Device calibration (see5.2.2)

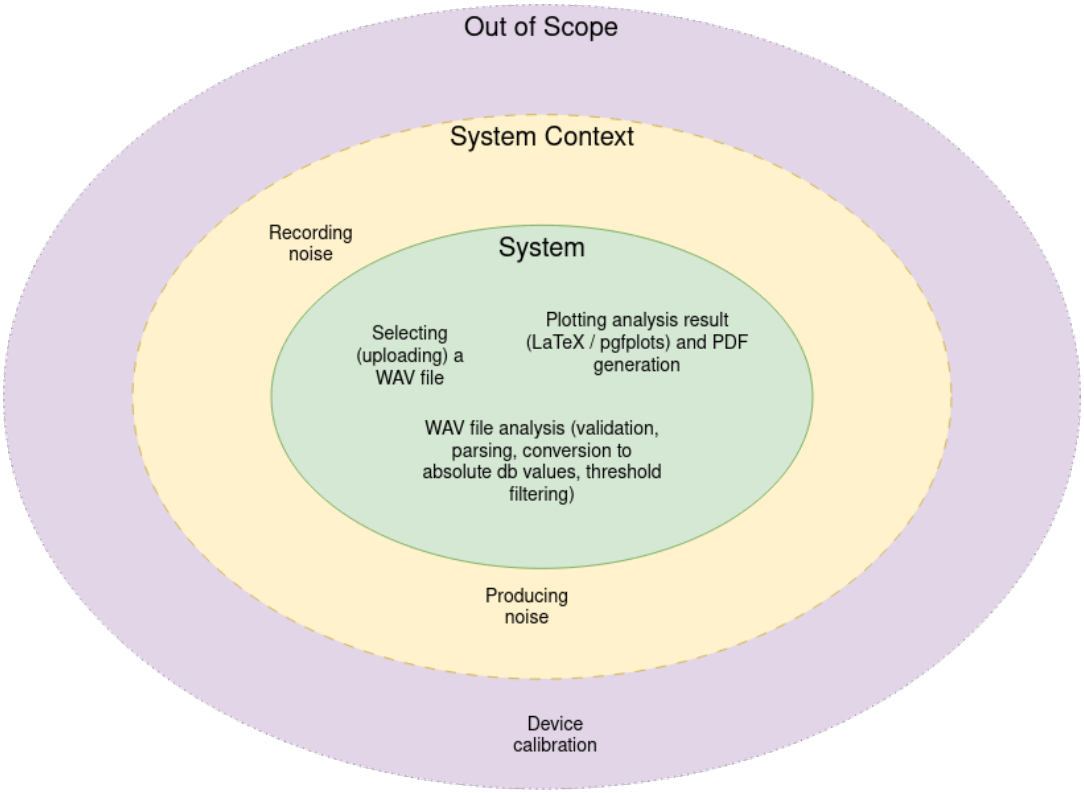


Figure 2: Process Environment

5.2 Requirements

In the following section the requirements are detailed. Also, the project boundaries and preconditions are described.

5.2.1 Functional requirements

The project team identified the following functional requirements:

ID	Requirement	Priority
R1	Allow users to upload a WAV audio file.	MUST
R2	Validate the uploaded file to ensure it is in WAV format, providing an error message if it is not.	MUST
R3	Analyse the uploaded WAV file and calculate the noise level in decibels (dB).	MUST
R4	Allow users to download the plotted noise data as an image or PDF.	MUST
R5	Allow users to input metadata for the audio file, such as location, date, and time.	MUST
R6	Plot the noise level data over time, with the x-axis representing time and the y-axis representing noise level (in dB).	MUST
R7	Generate a PDF report including the plotted noise data and user input metadata.	MUST
R8	Provide clear feedback and error messages.	SHOULD
R9	Intuitive and responsive UI for selecting files, configuring options, and viewing results.	SHOULD
R10	Allow the user to configure custom thresholds for noise levels.	COULD
R11	Allow the user to change the language of the application.	COULD

Table 1: Functional Requirements

5.2.2 Pre-Conditions and Boundaries

Research done by the project team has revealed that reading absolute decibel values from WAV files is more difficult than anticipated. The reason for this is that normal consumer microphones are not calibrated for scientifically accurate measurements [5]. Furthermore, it is trivial for a user to increase the volume of a WAV file using free, easy-to-use software like Audacity [6, 7]. Therefore, the project team has decided on a tentative working hypothesis together with the Stakeholder.

Pre-Conditions:

- The user must use a properly calibrated microphone or use a phone app which has built-in calibrations like Decibel X [8] [9].
- The user must have a way to export their audio recording as a WAV file.
- The user must have access to a web browser capable of running WebAssembly.
- The user must not edit the exported WAV file in any way except to decrease the length of the recording.

Boundaries:

- The application supports only the analysis of pre-calibrated, unedited WAV files.
- The application supports only single-channel WAV files.
- The application does not allow generating a legal complaint or directly integrating its output in a legal complaint.
- The user is presumed to be using properly calibrated equipment. There is no validation done in the application to ensure proper calibration.
- The user is presumed to be honest and to not have edited the WAV file. There is no validation done in the application to ensure the file has not been edited.

5.2.3 Privacy and Legal requirements

Because we are analyzing audio, privacy is an important point to consider. For analyzing the audio, we will only look at the amplitude of individual samples and take the rms over 300ms. No further analytics will be made. The get the rms, we combine a number of individual samples into a single one, which will then be ploted in the resulting document. Because of the rms we take, it is impossible to reconstruct the original audio from the plot. Additionally, we don't use any library to analyse the audio so we have full control over what's happening.

Furthermore, the application will run solely on the client and the server are only providing the html, css and javascript documents, hence no data is sent back to the server. We also don't safe any information from the user like ip address or the web client he uses. But we will deploy our website with GitHub pages, which means that we don't have any control about the information GitHub saves about the user.

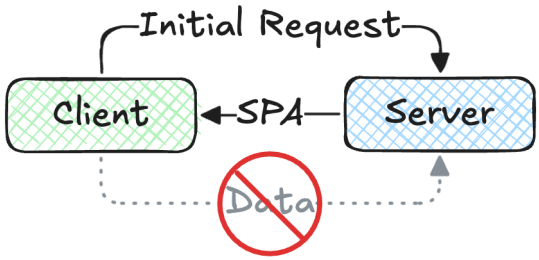


Figure 3: Privacy Illustrated

5.3 Usability

In order to best understand the targeted user base, the project team identified personas (5.3.1) and came up with a storyboard (5.3.2). Using this specification, the project team created prototypes for the user interface as well as the PDF report generated by the application (5.3.4).

5.3.1 Personas

Noise is a problem almost every demographic can be affected by. Nonetheless, the project team has attempted to capture the most common attributes of people likely to use the application.

Name	Daniel
Age	42
Sex	Male
Occupation	Business consultant
Marital Status	Married & has two young children
Lifestyle	Daniel usually has to get up early in the morning because his customers are spread across the entire German-speaking part of Switzerland, and they expect him to meet them in their offices.
Goals	Daniel wants to expand his client base in hopes of possibly quitting at his workplace and founding his own consulting firm.
Frustrations	Daniel needs a lot of sleep which has been difficult ever since his first child was born. Much worse however, is that his new upstairs neighbours are up late every night, playing loud music, stomping across the floor, or arguing loudly. Daniel has already asked them to be more considerate but to no avail.
Expectations	Daniel expects to find a way to prove to the police that his neighbours are consistently breaking the law and interfering with his and his families' lives. He expects this solution to be free, easy to use, and reliable.

Table 2: Persona 1 (Daniel)

Name	Julia
Age	25
Sex	Female
Occupation	Student
Marital Status	Single
Lifestyle	During the day Julia is mostly attending lectures at university, running errands, or going after one of her hobbies. In the evenings she studies, often long into the night.
Goals	Julia wants to land a scholarship at a prestigious university, meaning she has to be top of her class.
Frustrations	Julia's apartment has windows facing a busy street, and she regularly faces difficulties focussing on her studies due to the noise. She is convinced her landlord has neglected to install the noise isolation required by law.
Expectations	Julia is looking for a way to make her landlord install better noise isolation to the apartment complex so she and her neighbours can spend their evenings without being bothered by the noise of passing cars and trucks.

Table 3: Persona 2 (Julia)

5.3.2 Storyboard

Lorem ipsum

5.3.3 Process Model

The project team has created the following process model 4, which is a high level illustration of how the application should work.

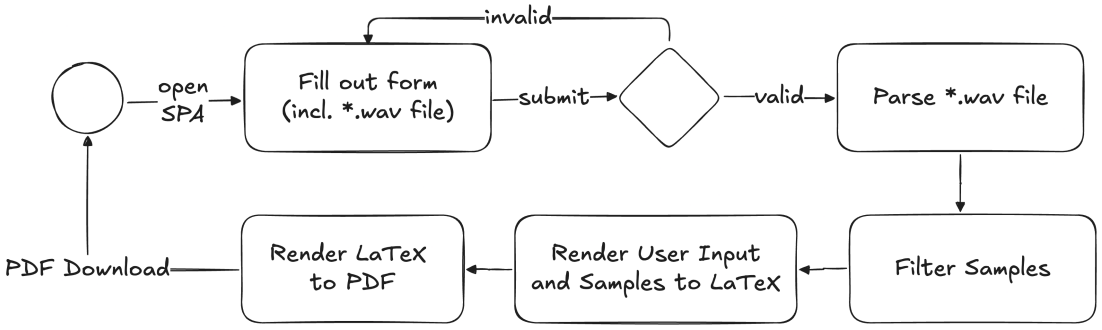


Figure 4: Process Model

5.3.4 UX-Prototyping

Based on our the previous specification, especially the functional requirements 5.2.1 and the process model 4, the project team identified two main UX components of the application and created their respective UX-Prototypes:

- UX Prototype – Website 5, 6
- UX Prototype – PDF Report 7

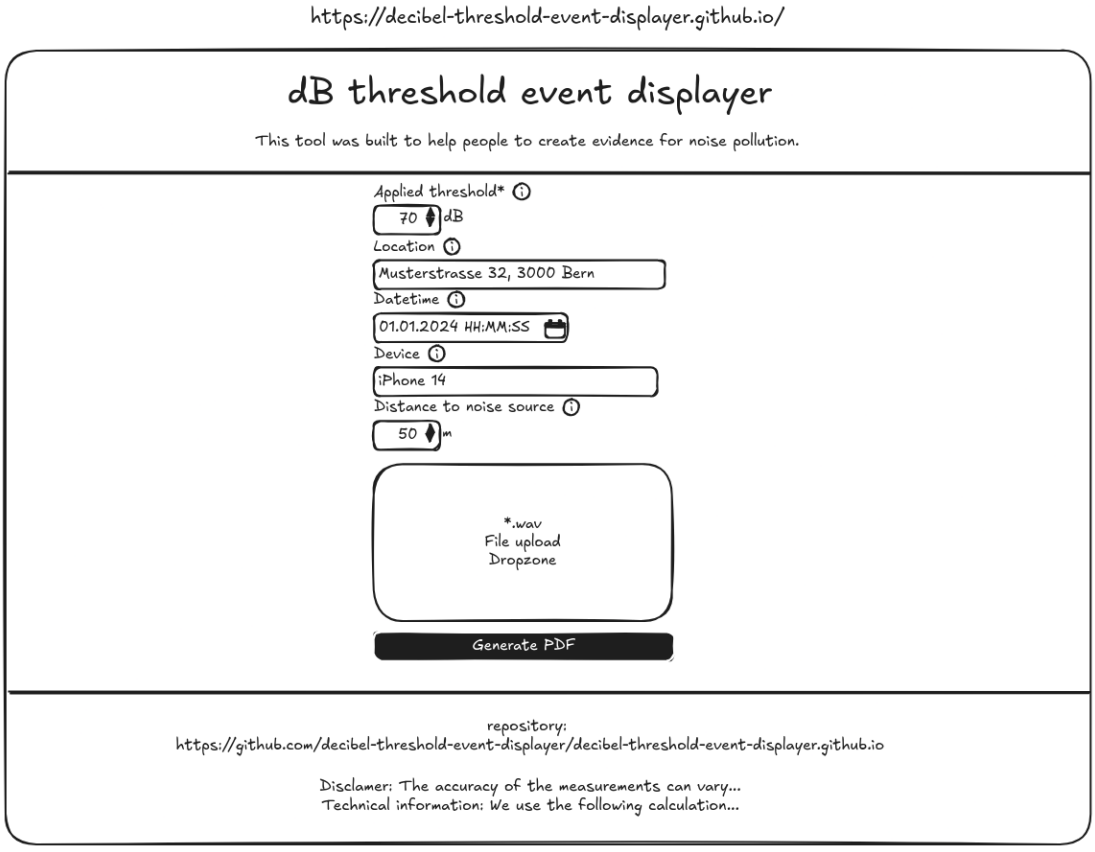


Figure 5: UX Prototype – Website

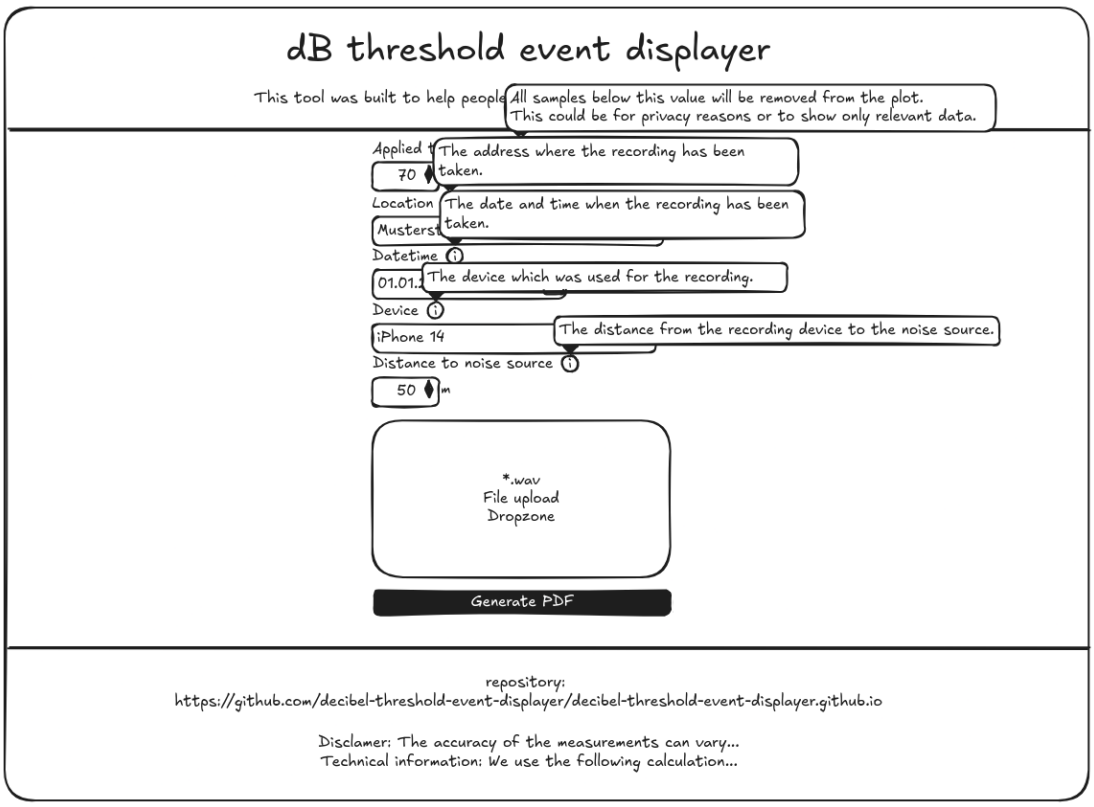


Figure 6: UX Prototype – Website with open tooltips

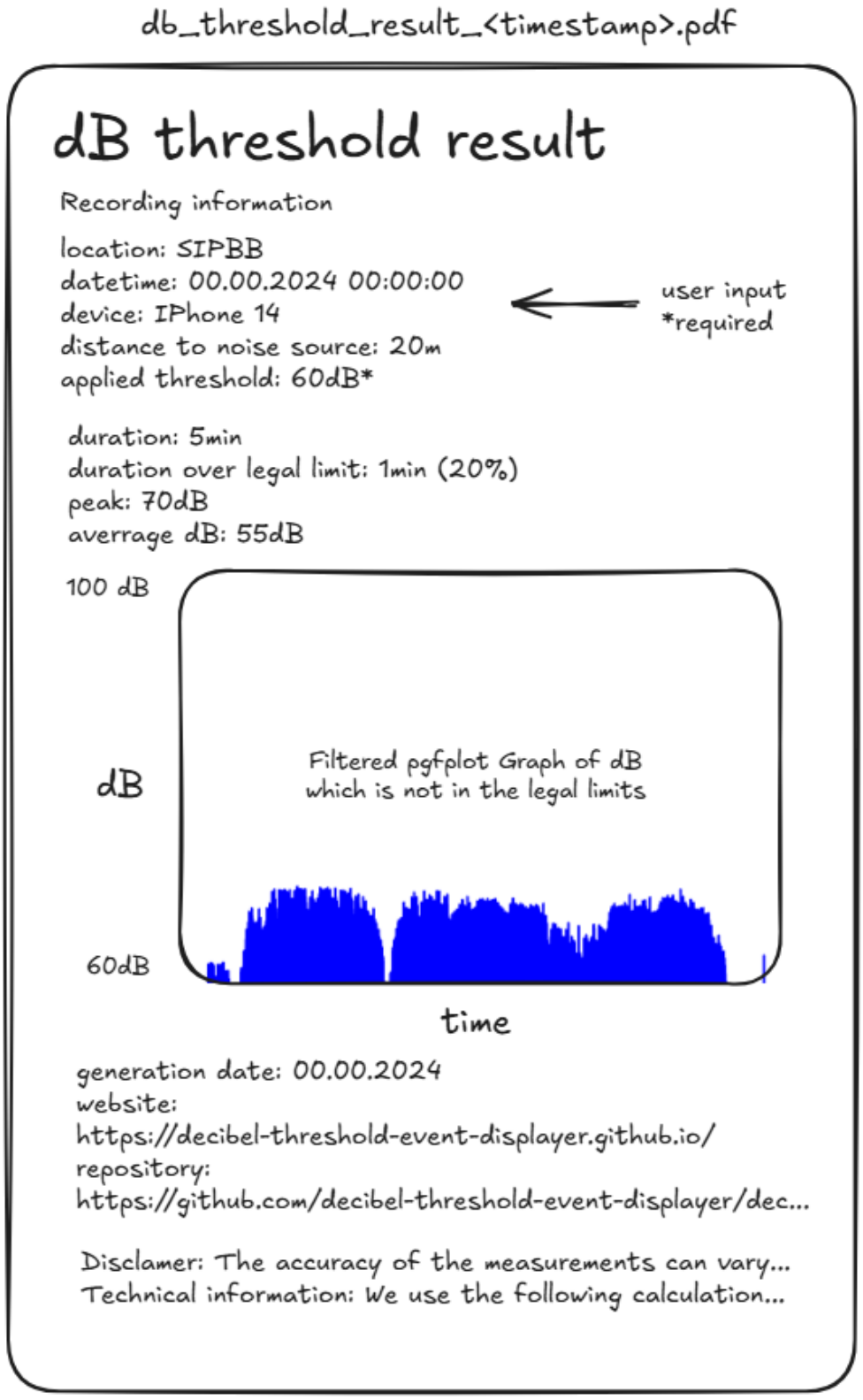


Figure 7: UX Prototype – PDF Report

6 Evaluation

6.1 Technology stack

Since the project description does not mention specific technologies to be used except for the LaTeX package pgfplots, the project team decided to evaluate different technologies for the project by creating working prototypes. The following chapters describe the evaluation criteria and the results of the evaluation.

6.1.1 Criteria

The following criteria are used to decide which technologies should be used for the project:

- Know-how (10%)
- Complexity (20%)
- Usability (30%)
- Distribution (40%)

6.1.2 Calculation

The formula for calculating the weighted value of each criterion is:

$$v_w = \text{weight} \cdot \frac{\text{evaluation}}{5}$$

Where *evaluation* is the scoring between 1–5. The total score for a given technology is the sum of weighted values.

6.1.3 Technologies

The following technologies were evaluated:

- Kotlin with pdflatex as a dependency
- Kotlin bundled with pdflatex
- Website with SwiftLaTeX

6.1.4 Reference LaTeX pgfplots template

For building the prototypes we will use the following reference LaTeX template which includes a pgfplots object: [10]

```
\documentclass{article}

\usepackage{siunitx}
\usepackage{tikz} % To generate the plot from csv
\usepackage{pgfplots}

\pgfplotsset{compat=newest} % Allows to place the legend below plot
\usepgfplotslibrary{units} % Allows to enter the units nicely

\sisetup{
  round-mode          = places,
  round-precision     = 2,
}

\begin{document}

\begin{figure}[h!]
  \begin{center}
    \begin{tikzpicture}
        \begin{axis}[
```

```
            xmin=-3,   xmax=3,
            ymin=-3,   ymax=3,
            extra x ticks={-1,1},
            extra y ticks={-2,2},
            extra tick style={grid=major},
        ]
            \draw[red] \pgfextra{
              \pgfpathellipse{\pgfplotspointaxisxy{0}{0}}
                {\pgfplotspointaxisdirectionxy{1}{0}}
                {\pgfplotspointaxisdirectionxy{0}{2}}
              % see also the documentation of
              % 'axis direction cs' which
              % allows a simpler way to draw this ellipse
            };
            \draw[blue] \pgfextra{
              \pgfpathellipse{\pgfplotspointaxisxy{0}{0}}
                {\pgfplotspointaxisdirectionxy{1}{1}}
                {\pgfplotspointaxisdirectionxy{0}{2}}
            };
            \addplot [only marks,mark=*] coordinates { (0,0) };
        \end{axis}
        \end{tikzpicture}
    \caption{My first autogenerated plot.}
  \end{center}
\end{figure}

\end{document}
```

Listing 1: Reference LaTeX template with a pgfplots object

6.1.5 Kotlin minimal – Kotlin with pdflatex as a dependency

This prototype uses Kotlin to satisfy platform-independence and invokes the pdflatex binary which is included in the TexLive distribution. The project team chose to evaluate this technology because it satisfies the base requirements according to the project description and because the project team is familiar with Kotlin. Another advantage of this technology is that the distribution of the necessary LaTeX installation and packages is relegated to the user. This greatly reduces distribution complexity.
An obvious disadvantage of this approach is that the user needs to install TeXLive (and the pgfplots package) on their system, which might be difficult for non-technical users.

6.1.5.1 Licensing

This prototype is dependent on both TeXLive and pgfplots. TeXLive uses a libre license that allows for the free redistribution with or without modifications [11] in accordance with the Free Software Foundation's free software definition [12]. The pgfplots package uses the GPLv3 license [13].

6.1.5.2 Evaluation

Criterion	Evaluation (1-5)	Weighted value
Know-how	4	8
Complexity	5	20
Usability	1	6
Distribution	5	40
Total	n.a.	**74**

Table 4: Evaluation of Kotlin with pdflatex as a dependency

6.1.6 Kotlin bundled – Kotlin bundled with pdflatex

Similarly to the prototype in Kotlin minimal 6.1.5, this prototype uses Kotlin and invokes the pdflatex binary. The major difference is that the pdflatex binary is provided alongside the software. The advantage of this approach is that the user does not have to install TexLive or any dependencies, greatly improving usability. However, this comes with the disadvantage that the complexity of the distribution is much higher. This approach also limits the advantages of using a platform-independent programming language because for every platform there has to be a release which packages the appropriate binary.

6.1.6.1 Licensing

The licensing is equivalent to the licenses of the prototype in Kotlin minimal 6.1.5.

6.1.6.2 Evaluation

Criterion	Evaluation (1-5)	Weighted value
Know-how	3	6
Complexity	3	12
Usability	5	30
Distribution	1	8
Total	n.a.	**56**

Table 5: Evaluation of Kotlin bundled with pdflatex

6.1.7 Web with SwiftLaTeX

During the projects initial discussions, which had brainstorm like character, we wondered if we could render latex files directly in the browser, which would have the following benefits:

- Providing the functionality as a web application would be probably the easiest and most accessible way for an end user to use the tool.
- Introducing the additional constraint of running the application only on the client side, would also avoid any infrastructure service and maintenance cost.

After a quick search we found the following open source JavaScript/WebWebassembly project: SwiftLaTeX: WYSIWYG LaTeX Editor for Browsers According to their website, the JavaScript/WebWebassembly library has the following characteristics [14]:

- 100% Browser – PdfTeX and XeTeX written in 100% WebAssembly and run in browsers.

- Compatibility – Produce exact same output you would get from TexLive or MikTeX.
- Library Support – Simply include a script tag and use PdfTeX or XeTeX in your own webpage.
- WYSIWYG – Support WYSIWYG editing on LaTeX documents using XeTeX engine.
- Speed – Run merely 2X slower than native binaries.
- Open Source – Completely Open Source. You can find the code on GitHub.

If the statements on the SwiftLaTeX website hold true, it would neatly full-fill our requirements for building a client side only web application.

6.1.7.1 Verifying basic functionality

To make a first feasibility test we can insert our reference LaTeX pgfplots template 1 into the SwiftLaTeX Demo tool. Even though it took more than one and a half minute to process, we were very surprised that this just worked, see figure 8:

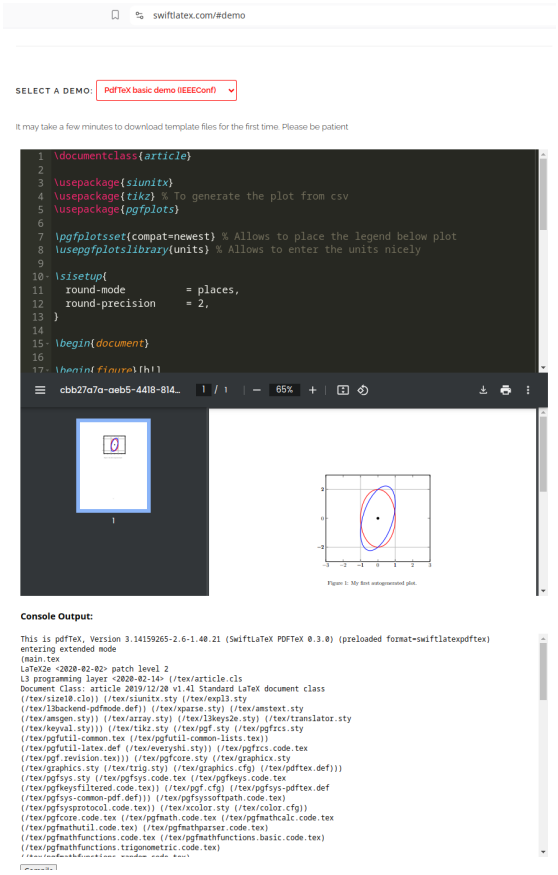


Figure 8: Screenshot of the SwiftLaTeX Demo tool rendering the reference LaTeX pgfplots template

6.1.7.2 Investigating dependencies are loaded

After investigating what actually happens on the page with the browsers developer tools network tab, we saw that SwiftLaTeX will load all dependencies asynchronously via https://texlive2.SwiftLaTeX.com/. This process and how self-host those files, is documented in the README.md of the SwiftLaTeX GitHub library [15]. As we will be able to specify which LaTeX dependencies our application

will use, it would be interesting to download and serve only the files we actually need. We also saw that the dependencies are loaded sequentially, which we would like to optimize to reduce the pdf generation time.

6.1.7.3 Local development setup

To make the same example working on a local machine, we can download the latest release from the GitHub repository: Releases 20/02/2022. The release contains the website visible at SwiftLaTeX.com including the compiled WebAssembly files. As expected, opening the index.html file from the unzipped release folder in a browser will look like the website SwiftLaTeX.com. Validating the basic functionality with our reference LaTeX pgfplots template 1 did not work, as we get the following error message in the JavaScript console:

```
PdfTeXEngine.js:89 Uncaught (in promise) SecurityError: Failed to construct '
    Worker': Script at 'file:///home/lukas/Downloads/20-02-2022/SwiftLaTeXpdftex.
    js' cannot be accessed from origin 'null'.
...
```

Listing 2: SwiftLaTeX local development setup: JavaScript error message

According to the following stackoverflow question, Google Chrome does not load web workers when running scripts from local files. The web worker is required to run WebAssembly, which does all the actual LaTeX to PDF transformation in this prototype. The easiest solution for this issue is to run a webserver, luckily this is trivial with Python. We just have to run the following command in the directory of the unpacked SwiftLaTeX release folder:

```
python -m http.server
```

Listing 3: Start a webserver on http://localhost:8000/ with Python

Validating the basic functionality with our reference LaTeX pgfplots template 1 on http://localhost:8000/ works now as expected.

6.1.7.4 Distribution option 1: Python webserver

This would be the easiest solution, as it is analog to the local development setup described in the previous paragraph. But installing Python, downloading the git repository or downloading a release and running the Python web server in the correct directory, requires at least a lot of technical affinity.

6.1.7.5 Distribution option 2: PyInstaller

To enhance the end users experience we could build an executable from a Python file with PyInstaller which supports builds for Windows, macOS and Linux. Even though this solution worked in a quick proof of concept on a linux system, the user would need to find the public GitLab repository and download a release, which contains the latest executables. Further the user might have security concerns, as downloading an executable from an unknown or untrusted source is really dangerous.

6.1.7.6 Distribution option 3: GitHub Pages

After thinking about it, we figured out that we could deploy the tool as a static HTML page to GitHub pages, this would resolve the need of downloading any source code or executable program. In our opinion this would result in the best user experience, as the user only needs a web browser to access and use the tool. As this solution was worth exploring, we deployed the prototype with GitHub pages: https://vonal3.github.io/

6.1.7.7 Licensing

- SwiftLaTeX: SwiftLaTeX License, GNU AFFERO GENERAL PUBLIC LICENSE
- LaTeX: The LaTeX project public license
- TeXLive: TeX Live licensing, copying, and redistribution, COPYING CONDITIONS FOR TeX Live
- pgfplots: https://ctan.org/pkg/pgfplots, GNU General Public License, version 3 or newer

6.1.7.8 Evaluation

Criterion	Evaluation (1-5)	Weighted value
Know-how	3	6
Complexity	3	12
Usability	4	24
Distribution	5	40
Total	n.a.	**82**

Table 6: Evaluation of SwiftLaTeX JavaScript/WebAssembly library

6.1.8 Chosen Solution

Using the evaluations given to the selected technologies, the following demonstrates the combined results:

Technology	Total score
Kotlin minimal	74
Kotlin bundled	56
Web SwiftLaTeX	82

Table 7: Technology stack evaluation

In accordance with the table above, the project team decided to use **Web SwiftLaTeX** with the distribution option 3 **GitHub Pages**.

6.2 License

After evaluating the different possible technologies, the following third party software will be used:

- SwiftLaTeX: SwiftLaTeX License, AGPL-3.0
- pgfplots: https://ctan.org/pkg/pgfplotspgfplots, GPL-3.0

To be conform with the used software, we will also publish our code under the GPL-3.0 licence.

7 Implementation

7.1 Architecture

We use github pages to deploy our website. The initial request deliviers all HTLM, CSS and JavaScript documents needed for the page to function. All further calls to the server will be request for latex dependencies for rendering the pdf. As seen in Figure 9, the application contains out of 3 core components: the single page application (SPA), the audio processing component and the swiftlatex library for rendering the pdf.

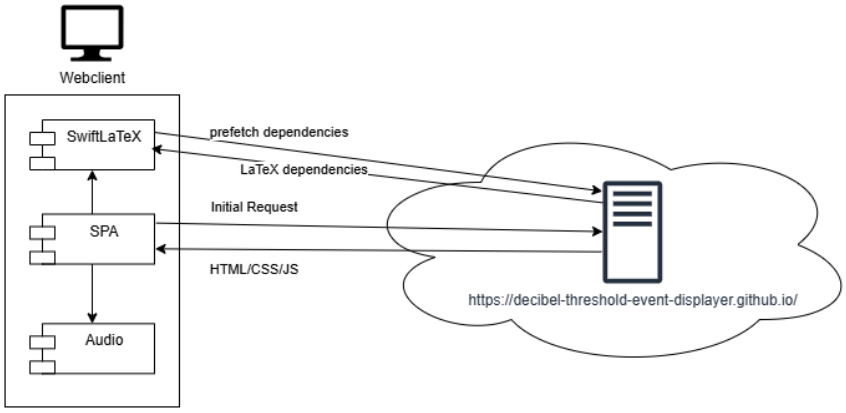


Figure 9: Class diagram

7.1.1 Data Structures and Overview

For conveniently accessing the data contained in the wave file, we created the WaveFileWrapper class. The wrapper is used by the home components to access the relevant data needed to create an instance of the FrameCollection class. A FrameCollection object contains all Frame objects of the audio file and a single instance of the Frame class represent a set of samples covering a time span of 300ms. A full overview of the data structures and how they are related to each other can be seen in 10

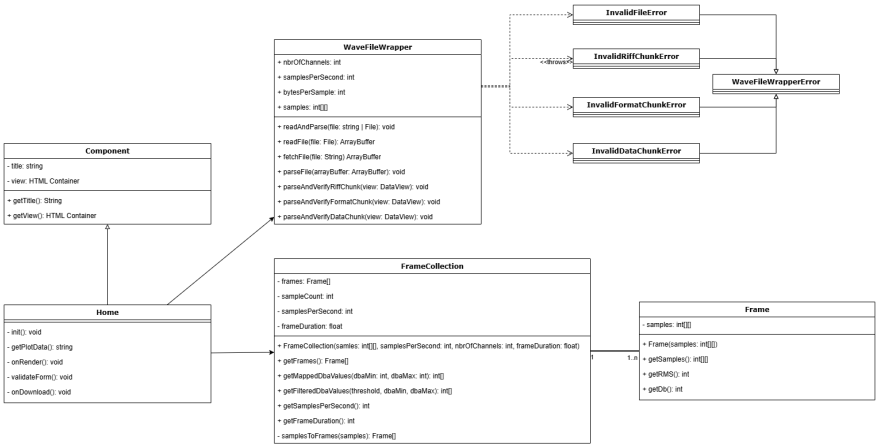


Figure 10: Class diagram

7.1.2 JavaScript Frontend (SPA)

The frontend hosted on Github Pages is implemented as a single page application (SPA) following the architecture provided by the BFH in the module BTI1301 Web Programming. In this

architecture, the web server provides the index.html file as and uses JavaScript to manage the web contents (DOM updates) without changing the HTML file itself. The project team chose to implement a small framework allowing for future expansion without rewriting the entire application. Since the application uses SwiftLaTeX, which is run locally using WebAssembly, the SPA provides an interface for the PDF engine to its components. The Single Page Applications Architecture is visualized in 11

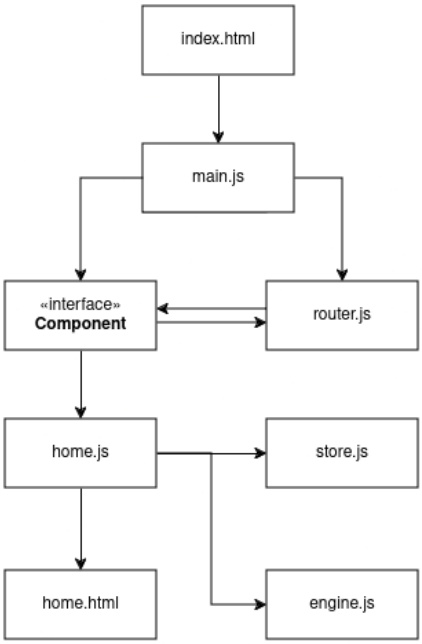


Figure 11: SPA Architecture

In order to ensure a consistent and agreeable look, the project team has decided to use Bootstrap, which is licensed under MIT.

We also decided to define default values for the threshold that the audio should get checked against, for which we have orientated ourselves on the immission limits from the BAFU.

7.2 Processes

The following diagram 12 gives a broad overview over the core steps involved when creating a noise report from a wave file.

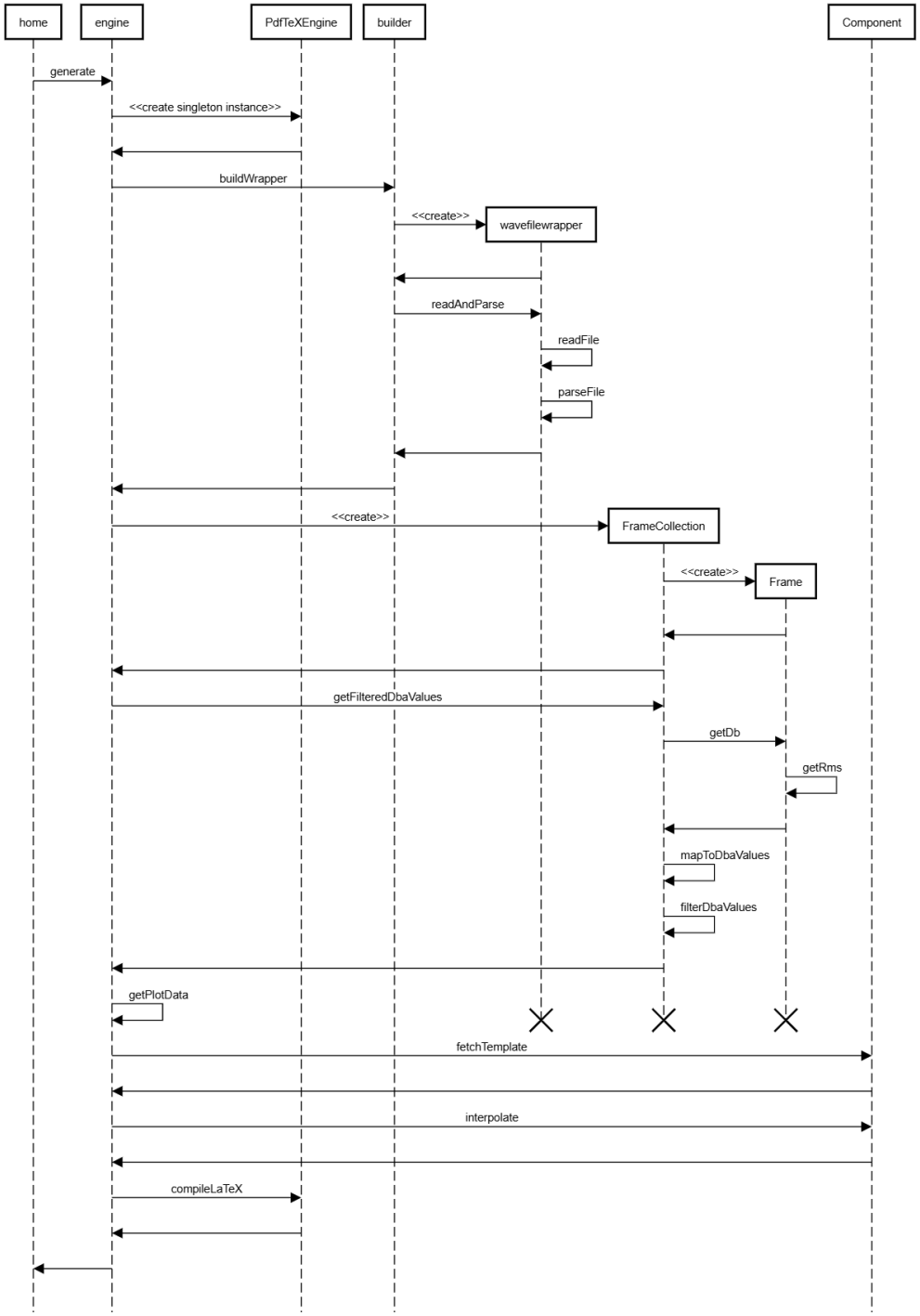


Figure 12: Sequence diagram of the steps involved when creating a noise report

7.2.1 Parsing the Wave File

The WaveFileWrapper class will be responsible for reading and parsing the wave file. Reading a file in a web context always works asynchronous, which is why there is a helper function ‘readAndParse‘ which has to be called with async await. The format itself is straight forward

[16], the only important thing is that the whole file is Little-Endian. It starts with a 12 byte header which we use to verify that the given file is actually a wave file. After the header comes a 24 byte chunk which contains information about the data format. We are interested in the following fields:

- AudioFormat (2 bytes): we only support PCM and IEEE Float format
- NbrChannels (2 bytes): is used to parse individual samples
- Frequency (4 bytes): number of samples per second, this is needed for filtering and summarising
- BitsPerSample (2 bytes): number of bits for a single sample for a single channel, is used to parse individual samples

After the format chunk can come several optional metadata chunks, which we don't need. Therefore we skip those chunks and go straight to the data chunk. This chunk starts with a 4 byte long DataBlocID, which should always have the value 'data'. The next 4 byte represent the number of samples in the data. After that comes the actual audio data. To parse a single sample, we need to parse a frame. A frame contains the data of all channels and has therefore a size of

$$\frac{NbrChannels * BitsPerSample}{8}$$

bytes.

7.2.2 Calculating Decibel Values

If an audio file consists of several channels, we use the average of the channels as the value of the sample. When we want to analyze where a certain decibel threshold has been exceeded, we must have accurate values. People only perceive something as loud if it is loud over a longer period of time. Short peaks are perceived less. Therefore, when determining the effective loudness of audio, we do not use the loudness of individual samples, but rather the average of a number of samples over a certain period of time. Such group of samples over a certain period of time is implemented as the Frame class. The FrameCollection class represents a list of frames for a whole audio file. We will, as usual when working with audio, use the root-mean-square (RMS), as we are interested in the absolute amplitude values.

We implemented the calculation of the root-mean-square as the method 'getRMS()' on the Frame class, which in turn calls the rms function shown in 4.

```
/**
 * values: a list of samples, if the samples has multiple channels they need to
     be averaged beforehand
 */
function rms(values){
    const squared = values.map(sample => Math.pow(sample, 2));
    const sum = squared.reduce((a, b) => a + b);
    const mean = sum / values.length;
    return Math.sqrt(mean);
}
```

Listing 4: JavaScript RMS function

The calculated average then depends heavily on how long the time period is. In practice, 300ms is usually used [17]. The decibel value is calculated with the formula shown in 5, which we derived from [18].

```
/**
 * rms: a rms value
 */
function rmsToDb(rms) {
    return 20 * Math.log10(rms);
}
```

Listing 5: JavaScript RMS to DB function

We implemented again a method 'getDb()' on the Frame class, which computes this value.

7.2.2.1 Mapping RMS to absolute dB(A) values

As mentioned in the introduction, the audio file only contains relative values. We therefore need to map the RMS values to the number range of the reference values, where the smallest sample is mapped to the minimal dB(A) and the largest sample is mapped to the maximal dB(A) measured. This calculation does not work in the context of a single frame and is therefor part of the FrameCollection class. 6 shows the calculation of mapping between dB and dB(A) values.

```
/**
 * db: a frames db value from the previous calculation
 * dbMin: the minimum db value over all frames
 * dbMax: the maximum db value over all frames
 * dbaMin: the minimum dB(A) value, provided by the user
 * dbaMax: the maximum dB(A) value, provided by the user
 */
function dbToDba(db, dbMin, dbMax, dbaMin, dbaMax) {
    return (db - dbMin) * (dbaMax - dbaMin) / (dbMax - dbMin) + dbaMin;
}
```

Listing 6: JavaScript DB to DBA function

7.2.2.2 Filter dB(A) values

One of the requirements is to plot the dB(A) values which are higher than a certain threshold. For this reason we implemented the method ‘getFilteredDbaValues(threshold, dbaMin, dbaMax)‘ which returns the final output, which we can plot to a PDF file.

7.2.3 LaTeX Renderer / PDF engine

As we already stated the application leverages SwiftLaTex for rendering a LaTeX template to a PDF Document, via WebAssembly. The implementation details can be found on the following two resources:

- SwiftLaTeX Webiste
- SwiftLaTeX GitHub Repository

We integrated SwiftLaTeX into our application by downloading the latest release (20/02/2022) and unpack it to the directory ‘app/js/swiftlatex/lib/‘ and move the file ‘swiftlatexpdftex.js‘ to the parent directory ‘app/js/swiftlatex/‘. This way we don’t have to add a dependency management tool such as npm and can we could customize the code as needed. The application does the following steps at runtime:

1. Load the LaTeX template: ‘app/js/swiftlatex/template.tex‘
2. Replace the placeholders in the template with the actual values (interpolate)
3. Handover the output to SwiftLaTeX
4. Return the resulting PDF

Performance improvements Generating the PDF file from a LaTeX document in the browser takes quite some time, as it loads all dependencies on the fly. To speed up this process, we serve all required LaTeX files ourselves and prefetch them in parallel. This means we have to download all LaTeX files from a public source, we are using the same source as SwiftLaTeX: texlive2.swiftlatex.com

We collected all required LaTeX dependencies by modifying the SwiftLaTeX JavaScript, in such a way that during a sample all requests pdf generation, are stored in an array, which is then printed to the console. We put the console output in the file “app/dependencies.json“ and created a python script under “app/download_dependencies.py“, which downloads all LaTeX files to the directory “app/dependencies/“. We then copied the contents of “app/dependencies.json“ to “app/dependencies.local.json“ and made the paths relative. Now we load this file in our main

JavaScript file “app/app.js“ and use “Promise.all“ together with “fetch“ to load all dependencies into the cache of the browser.

Currently, this process involves some manual steps and could be automated in a later step.

7.3 User Experience (UX)

7.3.1 Main Page

The user experience (frontend) was implemented according to the specification 5.3.4 where possible. The project team decided to use Bootstrap v5 to provide coherent look and because it is licensed under MIT. Except for the dropzone to upload files, all components are provided in vanilla Bootstrap. Unfortunately, Bootstrap does not provide a built-in component to style the file upload as a dropzone. The project team decided against implementing another library due to the additional upkeep required. Instead, they decided for using the file input provided by Bootstrap. See the frontend implementation in the respective figures 13 14.

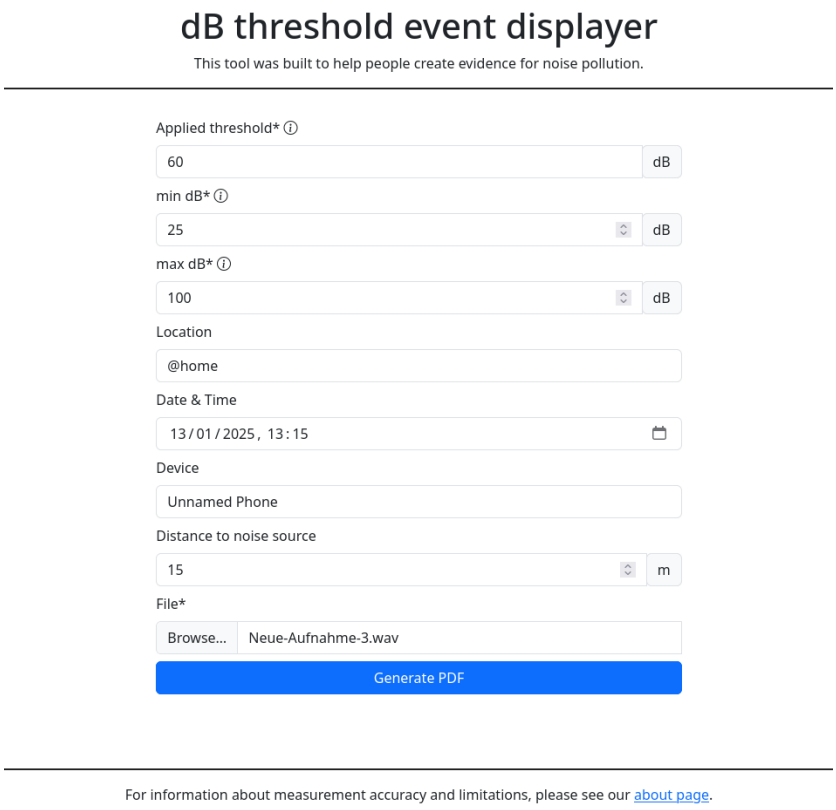


Figure 13: Frontend Implementation

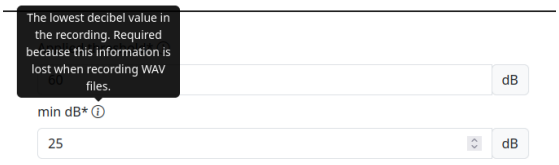


Figure 14: Tooltip

In order to support the user when determining thresholds, the applied threshold form field is implemented as a datalist tag. This allows the user to input custom values, while providing suggestions for pre-determined values 15.

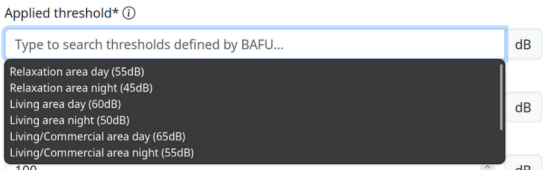


Figure 15: Default Thresholds

Another departure from the specification was the addition of a separate about page 7.3.2. Since it was easy to implement, the project team has also decided to add a preview for the rendered document alongside the download button. This also improves feedback because it allows the user to check what the file looks like before saving, letting them check for typos and other issues 16.

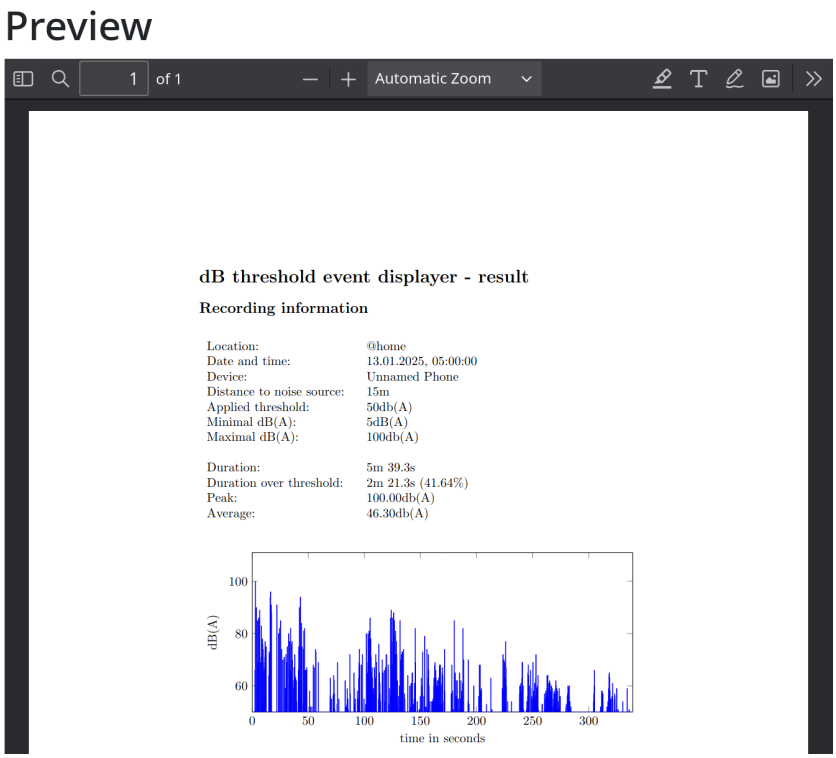


Figure 16: Report Preview

To provide additional feedback and ensure consistent results, the project team has also implemented validations for the important input fields 17.

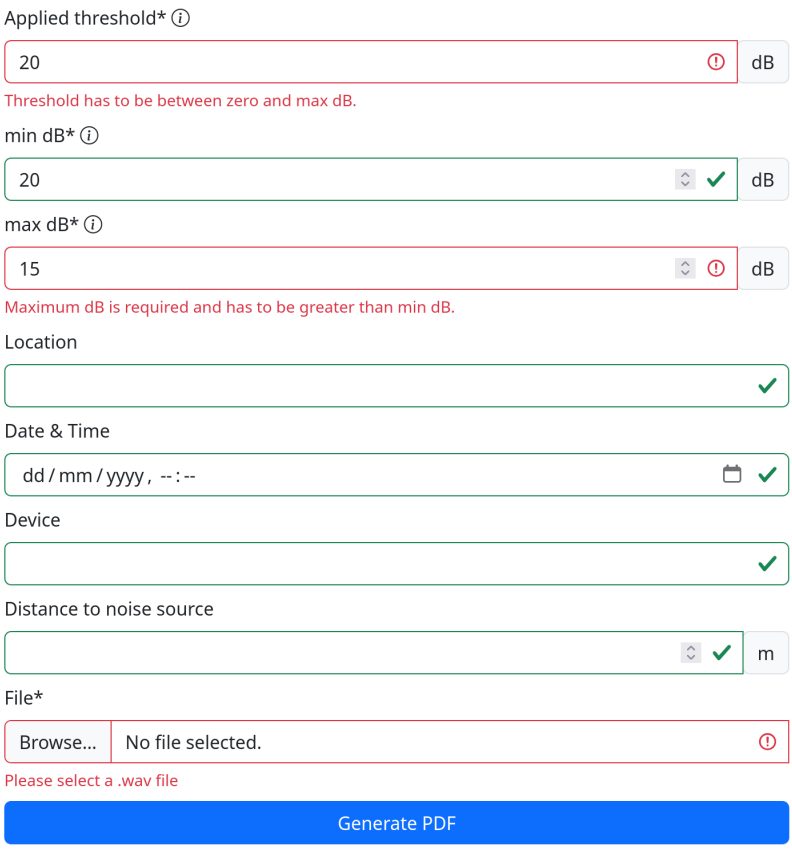


Figure 17: Form Validation

7.3.2 About Page

To provide users with additional information about the application's implementation, privacy concerns, calculation details, as well as an impressum, the project team implemented a separate about page. There is also a link to the GitHub repository, as well as a disclaimer about the accuracy of the results. See 18 for an excerpt of the implementation.

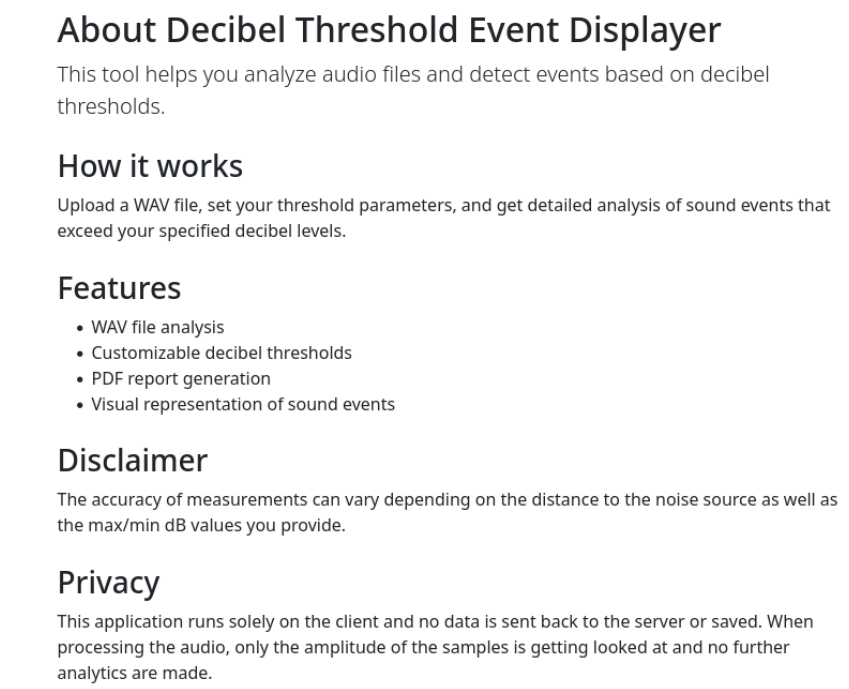


Figure 18: About Page excerpt

7.4 Testing

Because we don't use any Frontend Framework and instead build our own SPA, we don't have a testing framework to use. We therefore decided to build our own rudimentary testing framework. It consists out of 4 assert functions, a test runner and a html page for displaying the results. There is no automatic mechanism running the tests after every commit, the tests have to be run manually.

As we are developing an application for analyzing a wave file, nearly every tests first step is to create a WaveFileWrapper object for further use. Because the reading and parsing of the file works asynchronous, the test suit has also to be asynchronous. We decided to use the async/wait approach instead of using promises because promises are more difficult to debug, which is not suitable for a test framework. The following diagram 19 shows the internal structure of the test framework.

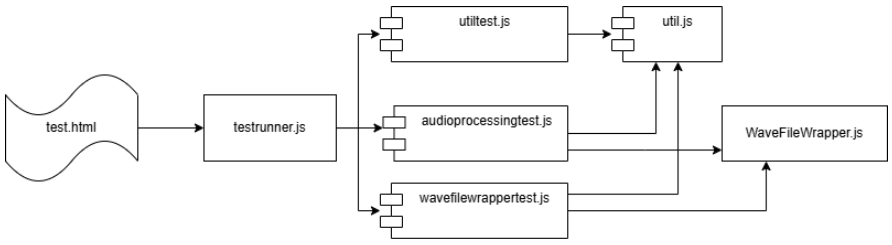


Figure 19: Overview test framework

7.4.1 Assert functions

For implementing tests there are 5 assert functions contained in the util module which can be used:

- assertThrows: verifies that a certain exception is thrown
- assertNotThrows: verifies that no exception is thrown

- assertEquals: verifies that something is equal to a given value
- assertNotEquals: verifies that something is not equal to a given value
- assertGreaterThen: verifies that a number is greater the other

If the assertion of a function fails, it throws an exception with some information on why the test failed. The assertEquals and assertNotEquals function work with any kind of object. They first stringify both objects to a json format and then compare the json. It is certainly not the most efficient method, but it works for all cases. The assertThrows and assertNotThrows both take in a function which will be called in the assert function itself and verified that it throws or not throws an exception. The assertGreaterThen can also take in two arguments from any type, as JavaScript allows such comparisons. With those functions we were able to implement all our tests. To make sure that the assert functions work, we implemented also tests for them. Those test obviously don't use the assert functions and rather have some hard coded tests created for them. Those tests can be found in the utiltest module.

7.4.2 Creating tests

To create tests, one has to create a JavaScript file containing its tests under 'app/js/test/'. All functions that are being exported by the given JavaScript file are considered to be tests and will be executed by the test runner. If a function throws an exception, it is assumed that the test failed. If no exception is thrown, the test passes. We tried to keep the manual work for adding tests minimal, but some manual work is still necessary. To add a new test module, the module has to be imported into the test runner file as shown in 7. After that, a new call to the function runTestGroup can be added as it's shown in 8, which will then add the test. Take a look at the existing tests as a reference.

```
import * as waveFileWrapperTest from './wavefilewrappertest.js';
import * as audioProcessingTest from './audioprocessingtest.js';
import * as utilTest from './utiltest.js';
```

Listing 7: Importing a module with tests

```
/**
 * Runs all test modules in sequence.
 */
function runTests() {
    runTestModule(utilTest, "Util");
    runTestModule(waveFileWrapperTest, "Wave File Wrapper");
    runTestModule(audioProcessingTest, "Audio Processing");
}
```

Listing 8: Adding a method call to run a test module

7.4.3 Test runner

The test runner is responsible for running the tests. It runs every test module in sequence and summarises its results. For each module a table in the result page is created which contains the test names, result and in the case of a failed test a message why it failed. The tests are run when the dedicated button in the frontend is pressed. The test runner always runs every test of every test module. There is currently no function to run individual modules or tests. The test runner also has a function to just test the WaveFileWrapper. It creates such a WaveFileWrapper object from the file specified on the result page and displays the values of its properties.

7.4.4 HTML page

The fronted page of the test framework is used to start the tests and view the results. It is additionally used to create a WaveFileWrapper object and view the properties of it. The test page is accessible under /js/test/test.html. To run the test page on the dev server on localhost, see chapter 8.2.1.

The screenshot 20 shows how the test framework validated an uploaded *.wav file.

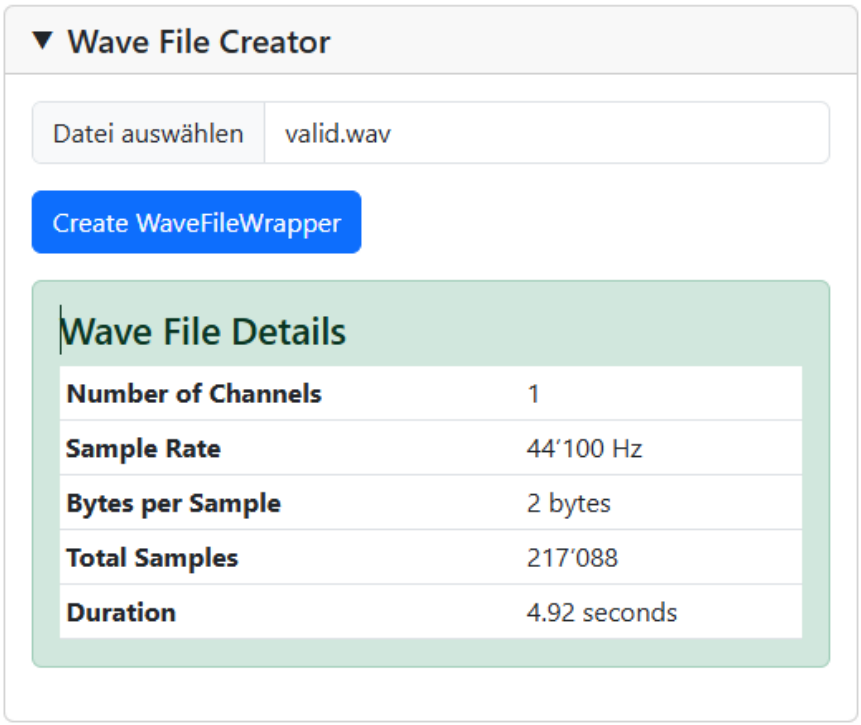


Figure 20: Create a WaveFileWrapper object and view its properties

On screenshot 21 we can see the test results after running the test runner.

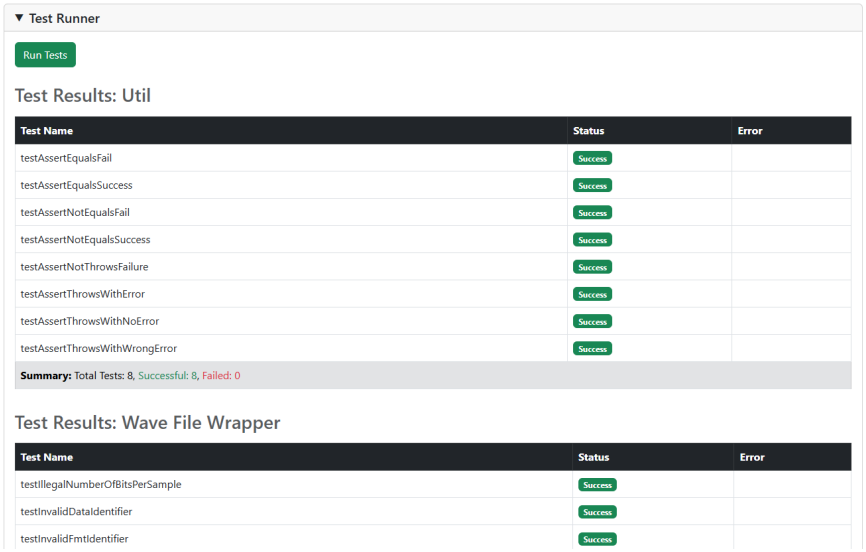


Figure 21: Test results

8 Deployment and Integration

We decided to use GitHub Pages via an organization for distribution because it provides an easy way to access the application for end users and simplifies the maintenance for developers. The GitHub organization is located under: https://github.com/decibel-threshold-event-displayer The application is deployed as a GitHub Page under: https://decibel-threshold-event-displayer.github.io/ The following list and the graphic 22 elaborates on the deployment workflow.

1. A dev pushes or merges code to the main branch
2. GitLab automatically mirrors the repository to GitHub
3. GitHub deploys automatically to GitHub Pages
4. The Application is available under:
   https://decibel-threshold-event-displayer.github.io/

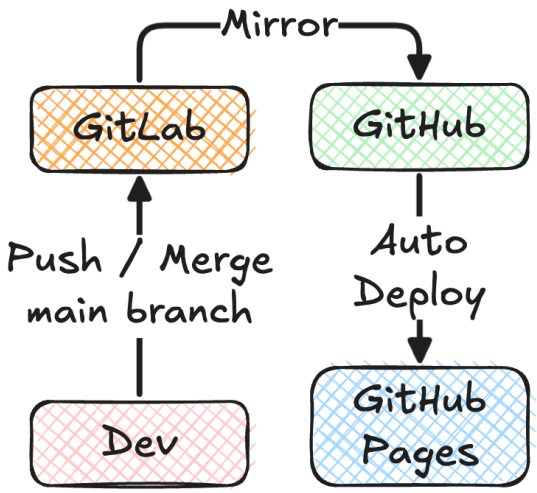


Figure 22: Deployment Flow

8.1 Licensing

We already started to cover the topic in the evaluation under License and even though the project team tried to introduce the least amount of external dependencies, the license still has to be revalidated at this point. The following list contains all dependencies of the final application and their according license:

- SwiftLaTeX: SwiftLaTeX License, GNU AFFERO GENERAL PUBLIC LICENSE
- LaTeX: The LaTeX project public license (GPL Compatible)
- TeXLive: TeX Live licensing, copying, and redistribution, COPYING CONDITIONS FOR TeX Live (GPL Compatible)
- pgfplots: https://ctan.org/pkg/pgfplots, GNU General Public License, version 3 or newer
- Bootstrap: Bootstrap License, MIT License (GPL Compatible)

Considering all licenses of our dependencies are GPL based or GPL compatible, we decided to use the following license for our application: **GPL-3.0 licence (FLOSS)**

8.2 Installation Manuel & Script

We are using a Makefile to automate the development setup, build and deploy the application to GitHub Pages.

8.2.1 Development setup

A local webserver is needed work on the application as some Browsers will not execute web workers on locally served files [19]. By default, the project uses the built-in webserver of Python 3. Python 3 can be installed with the systems package manager or by downloading it from the official website: https://www.python.org/downloads/ Afterward the following command can be used to run the local webserver on http://0.0.0.0:8000/:

```
$ make dev
```

Listing 9: Makefile: Start local webserver

8.2.2 Distribution

The project and the repository is managed via the GitLab of BFH, but we want to use GitHub pages to deploy and distribute our application, the project team decided that we mirror the GitLab repository to GitHub:

- GitLab Repository
- GitHub Repository

8.2.2.1 Mirror setup:

1. Log into gitlab.ti.bfh.ch
2. Navigate to the repository: GitLab Repository
3. Go to the repositories settings page: Settings > Repository
4. Open the Tab “Mirroring repositories“
5. Click the button “Add new“
6. Fill the form as follows:
    - Git repository URL: ssh://git@github.com/decibel-threshold-event-displayer/decibel-threshold-event-displayer.github.io.git
    - Mirror direction: Pull
    - Authentication method: SSH public key
    - Username: github
    - Mirror user: autofilled
    - Overwrite diverged branches: Select
    - Mirror branches > Mirror specific branches: main
7. Click the button “Mirror repository“
8. The first try will fail, as we have to add the public key generated by GitLab to the GitHub repository
9. On the newly created entry, click the clipboard button “Copy SSH public key“ (the public key is now in your clipboard)
10. Keep the current GitLab tab open
11. Log into github.io in a new tab
12. Navigate to the repository: GitHub Repository
13. Go to the repositories deploy keys page: Settings > Deploy Keys
14. Click the button “Add deploy key“

15. Fill the form as follows:
    - Title: github
    - Key: Copy the SSH public key from the clipboard
    - Allow write access:
16. Click the button “Add Key“
17. Go back to the gitlab tab
18. Click the reload button “Update now“
19. Wait until the process is finished

8.2.3 Build

As we have only plain JavaScript files, we don’t have a build step.

8.2.4 Deploy

As described above, we created a Mirror of this repository on GitHub. Based on the following documentation, we configured GitHub to automatically deploy to GitHub Pages: Creating a GitHub Pages site

Further we had to add a configuration file under “.github/workflows/static.yml“ and change value of “path: ./“ to “path: ./app“. The initial file was generated by the following setup page: Repository > Settings > GitHub Pages > Static HTML > Configure

8.3 User Manual

The project team decided to not create an additional document for the user manual, because all the necessary information for using the application are provided by following list of best practices:

- Following common UI patterns
- Well-known form components
- Tooltips where needed
- Human-readable error messages
- About page with detailed technical information

Part II
Scrum

9 Scrum Roles

The project team has, in coordination with the tutor, determined that the association Lärmliga (`https://laermliga.ch/`) will not be directly involved in the project. Instead, the tutor Dr. Simon Kramer, will take on the role of the stakeholder and render the decisions to be made by the customer.
The other scrum roles were determined internally by the project team. At first, the decision was made for Dr. Simon Kramer to also be the Product Owner. However, the project team realized it would be impractical to include Dr. Simon Kramer in every product decision and especially in the prioritization of the backlog. The project team felt that, due to their relative inexperience, it would be best to have the Product Owner in the team, so decisions could be made quickly and validated with the stakeholders afterward, if necessary.

Name	Role(s)
Dr. Simon Kramer	Stakeholder, Tutor
Dominic Gernert	Product Owner, Developer
Lukas von Allmen	Scrum Master, Developer
Darius Degel	Developer

Table 8: Scrum Roles

10 Sprint Goals

The sprint goals are defined by the project team at the Sprint Planning right after the conclusion of the previous sprint. They are formulated to be compliant with the S.M.A.R.T principles. The project team decides the sprint goals based on the backlog prioritization and the issues assigned to the next sprint. The sprint goals are defined and reviewed in a Markdown file in the repository (Gitlab). If we did not achieve a sprint goal and not stated otherwise, it's implicitly assumed to be done in the following sprint.

10.1 Sprint 1: 10.10. - 23.10.2024

For the first sprint, the project team decided to focus on research and prototyping. This can be understood as a feasibility study. Due to the team's relative inexperience with LaTeX and due to the project scope, they felt that having a working prototype had to be made before a design decision could realistically be made.

Goal	Reached
Prototypes with two different technologies are implemented and their pros and cons are evaluated	Yes
Tech stack for the project has been chosen	Yes
Licence is defined	No
Git repository and documentation skeleton are created	No

Table 9: Sprint goals of sprint 1

In sprint 1, the project team managed to complete the first two goals (prototyping and tech stack evaluation). However, they were unable to evaluate the licensing because the decision about which tech stack should be used was made only at the end of the sprint. The project team underestimated the issue weights, leading to goals that were not reached.

10.2 Sprint 2: 24.10. - 06.11.2024

For the second sprint we already had to set the focus on the intermediate presentation, further we had to finalize the groundwork such as specifying Requirements, System delimitation, UX-Prototypes and how to process audio files. Although the project team defined more goals for the second sprint than the first, it is important to note that the total weight of tasks assigned to this sprint is equal to the first sprint and estimates are more conservative.

Goal	Reached
Intermediate presentation is prepared and presented	Yes
Requirements are specified	Yes
UX-Prototype is defined	Yes
System delimitation is specified	Yes
Decibel values can be calculated	Yes
Licence is defined	Yes
Git repository and documentation skeleton where created	Yes

Table 10: Sprint Goals of Sprint 2

In this sprint the whole team made a big effort to achieve all the goals and finished the most important groundwork in the project, and we have now a clear way forward to build the product. Even though the criteria for the intermediate presentation were available on a short notice and creating a presentation in LaTeX turned out to be a bit challenging, the presentation was prepared in time and went well.

10.3 Sprint 3: 07.11. - 20.11.204

Based on the important groundwork in the last sprint, we could now start with the implementation of the application and the deployment and distribution setup. The amount of sprint goals is similar to the previous sprint, but we went with a bit less story points as we will also have to absolve the BFH project week 3.

Goal	Reached
Write documentation for interface	Yes
Read and parse *.wav files correctly	Yes
Filter audio data correctly	No
Enable repository mirroring for distribution	Yes
Implement MVP frontend application	No

Table 11: Sprint Goals of Sprint 3

Because the BFH special week 3 took our focus entirely (we were in the same team), we could not complete all tasks in this sprint. Nonetheless, we were able to finalize the first implementation and deployment tasks. Getting started again with JavaScript was a bit challenging, but with the realization of a rudimentary javascript test environment we could set another important cornerstone for a well working application.

10.4 Sprint 4: 21.11. - 04.12.2024

The focus off this sprint is to progress on the implementation of the application. The project team only defined 3 goals, because they already contain more than enough story points. After this sprint we should be able to test each system component individually.

Goal	Reached
Filter audio data correctly	No
Implement MVP frontend application	Yes
Improve javascript test environment	No

Table 12: Sprint Goals of Sprint 4

We only completed one of the three sprint goals, because achieving the definition of done, especially completing the documentation, for each task often took more time than we initially estimated. However, we made massive progress and could implement and test all the system components individually.

10.5 Sprint 5: 05.12. - 18.12.2024

In sprint 5 the project team will be able to connect all the system components and test the application for the first time as a whole.

Goal	Reached
Finalize and document 'Filter audio data correctly'	No
Document 'Improve javascript test environment'	Yes
Create LaTeX template and fill in placeholders	No
Render LaTeX to PDF	No

Table 13: Sprint Goals of Sprint 5

On paper we achieved only one of four sprint goals as in the end the project team has not enough time for completing the documentation part of the tickets. As a consequence, we decided to add more weight to the tasks, even if we would estimate them lower. On the other hand we made great overall progress and could generate our first pdf plot via the web gui from an uploaded *.wav file, which means we have a first version of the application where all system components are connected.

10.6 Sprint 6: 19.12.2024 - 01.01.2025

According to our plan this should be the last sprint and there is a lot of finalization, documentation and cleanup do be done on all different fronts. We decided to go with 5 sprint goals which is really ambitious, because afterward the project would be more or less done.

Goal	Reached
Finalize and document 'Filter audio data correctly'	No
Finalize LaTeX template and fill in placeholders	No
Finalize LaTeX to PDF	No
Finalize SPA	No
Define legal requirements	Yes

Table 14: Sprint Goals of Sprint 6

As we already knew, the sprint goals where really ambitious and more than half of this sprint is during the holiday season, so we actually completed only one of the five sprint goals. We still

managed to get a lot of things done and even though we still have open tasks on all fronts, we are almost finished with the application.

10.7 Finalization: 01.01. - 15.01.2025

The project team decided that it does not make sense to start another sprint just for the sake of it, the leftover sprint goals and tasks will be done in a finalization round, which can be thought of an extension of sprint 6, as it was the last sprint anyway.

Goal	Reached
Finalize and document 'Filter audio data correctly'	Yes
Finalize LaTeX template and fill in placeholders	Yes
Finalize LaTeX to PDF	Yes
Finalize SPA	Yes
Final presentation is prepared and presented	Yes

Table 15: Finalization Goals

11 Requirements

11.1 Epics

Epics were defined at the very beginning of the project and are largely product focussed. Epics are defined in the Epics section of the Planning tab in Gitlab (Gitlab). Epics are subject to change but for the sake of completeness, they are included in this document anyway (see Figure 23).

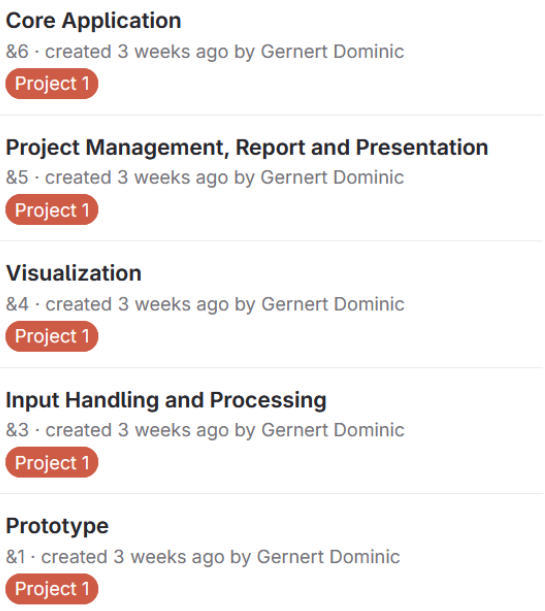


Figure 23: Epics

11.2 User Stories

User Stories are created as Issues in Gitlab. They are associated with an Epic and are prioritised. Issues contain a *Definition of Ready* (11.2.1), Acceptance Criteria, as well as a *Definition of Done*

(11.2.2). When an issue is selected for a sprint, sub-tasks are created, estimated and assigned. The issue itself is assigned to the person with the most assigned sub-tasks. The full list of issues is available in Gitlab's issue list or the Development Board.

11.2.1 Definition of Ready

The *Definition of Ready* is defined as a checklist on every User Story. In every Sprint Planning the issues selected for the next Sprint are validated by crossing off the checklist. All necessary adjustments are made together by the project team before the issue is estimated and planned. The *Definition of Ready* is part of the Issue Template in the repository (see Gitlab).

Definition of Ready:

- Requirements and Acceptance Criteria Defined
- Acceptance Criteria must be testable
- Understood by the Team
- Sized and estimated
- Prioritised in the Backlog
- No major Impediments

11.2.2 Definition of Done

Similarly to the *Definition of Ready* (11.2.1), the *Definition of Done* is a static checklist that is part of the issue template. The tasks are checked off by the reviewer, which is the Product Owner unless specified otherwise.

Definition of Done:

- Acceptance Criteria met
- Tested and no critical bugs
- Documentation updated
- Reviewed and approved

11.3 Product Backlog

Issues are prioritised first and foremost by the Product Owner. This prioritisation is implemented as tags in Gitlab and is can be viewed on the Development Board (see Figure 24)

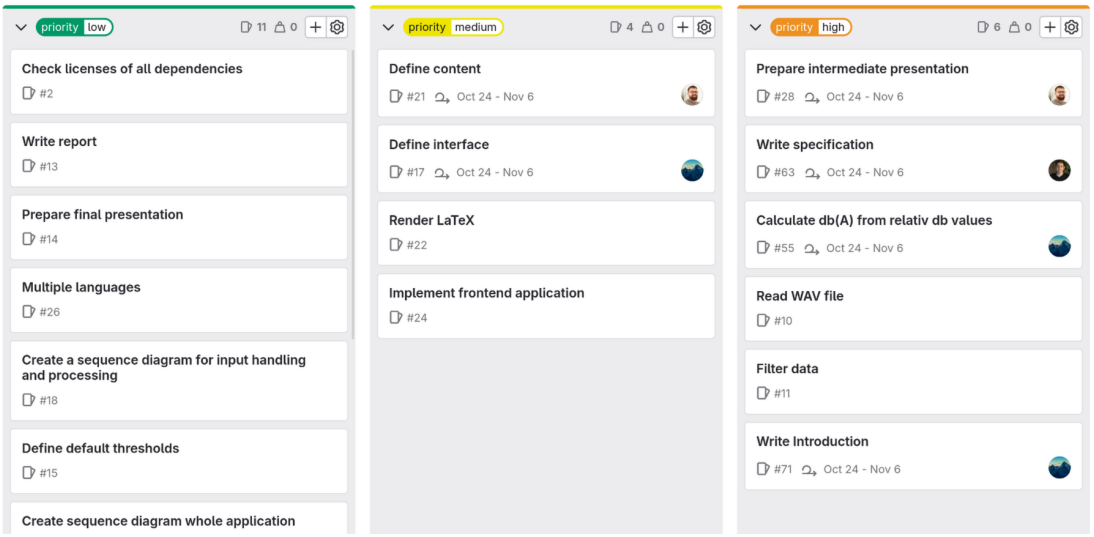


Figure 24: Product Backlog

11.4 Sprint Backlog

The Sprint Backlog of the currently running sprint is displayed as a column on the Development Board. In combination with the prioritisation (see 11.3), this makes for a convenient way for the project team to select issues for the next sprint based on priority (see Figure 25).

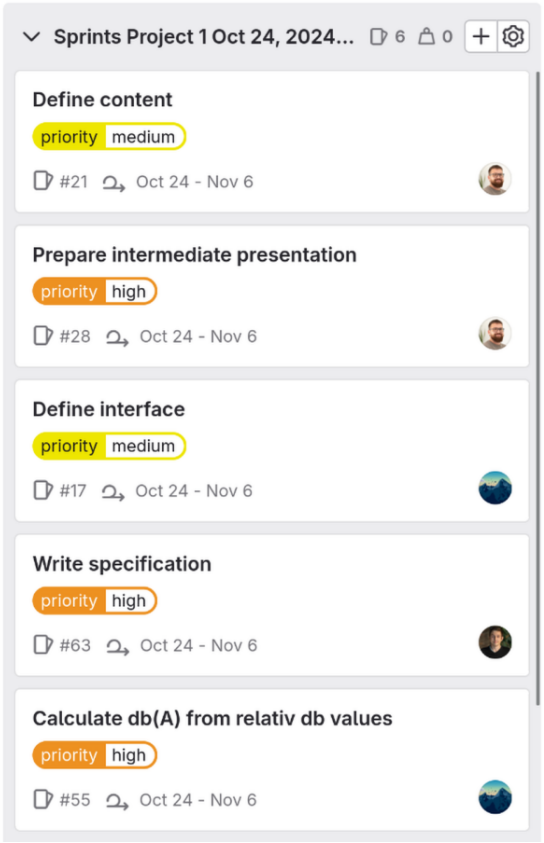


Figure 25: Sprint Backlog

11.5 Impediment Backlog

Impediments are also created as Issues in Gitlab. They are however, assigned the label *Impediment* and are displayed on a separate Impediment Board (see Figure 26 and Gitlab). Impediments are created either ad hoc or during the Daily Standup meeting. There is a template for Impediments analogous to the Issue Template (see Gitlab).

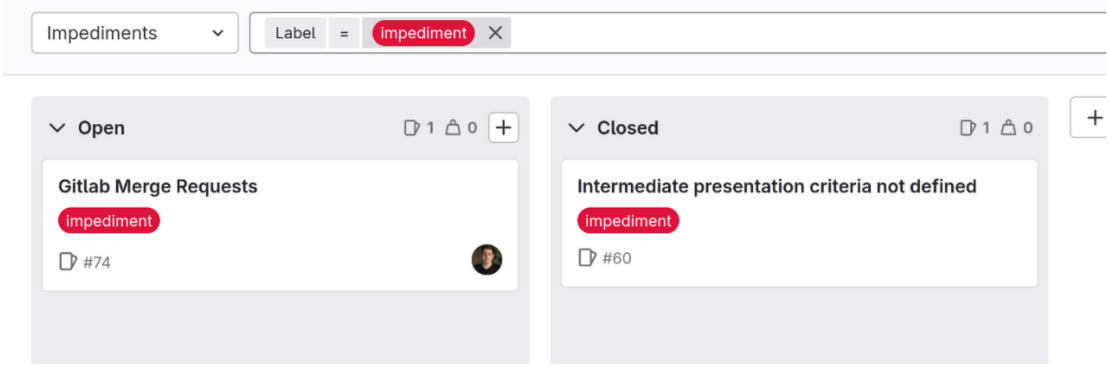


Figure 26: Impediment Board

12 Scrum Adaptations

12.1 Gitlab

Mr. Frank Helbling was added to the gitlab "Decibel Threshold Event Displayer" group of our project as a reporter (Gitlab Group). He has access to inspect Epics on group level. Issues and Issue Boards can be accessed on project level. The Daily Journals and Sprint Notes can be accessed in the doc/scrum folder (Scrum Notes).

12.2 Product Owner

As described in section 9, the Product Owner role has gone from our tutor, Dr. Simon Kramer, to Dominic Gernert. Although not specifically mentioned in the Scrum Guide [20], in the experience of the project team and an article on applied frameworks [21], the Product Owner is usually someone with a business background and not a developer. This makes sense, because they need to be able to make decisions about the product that only a business expert could make. Since this is not the real world but a school project, the project team had to reduce the coordination effort. Thus, the team decided to make Dominic Gernert the Product Owner. Since he is not a business expert for our project, decisions are still made as a collective. If however, there is a disagreement on a decision, the Product Owner's suggestion is followed. For bigger decisions,the Stakeholder Dr. Simon Kramer is involved.

12.3 Daily Scrum

As suggested by Mr. Frank Helbling in the Scrum presentation [22], the project team has decided to confine their Daily Scrum to 15 minutes. They are scheduled weekly on Wednesday at 18:15. The Scrum meetings that took place until the completion date of this document took about 30 minutes, however. The likely reason for this is that the project team meets only weekly, but each person actually works on the project at multiple days of the week. This leads to more information, questions, and impediments being generated and having to be discussed during the Daily Scrum. The project team is still trying to keep their Daily Scrum short (less than 20min.) in the future.

12.4 Release Plan

The project team has decided against implementing a Release Plan. Mainly because it was never explicitly demanded by the tutor, Dr. Simon Kramer. The other reason for forgoing the Release Plan is that there is no real customer awaiting product updates and Dr. Kramer is capable of running the application himself should he want to do so. Product demonstrations by the team are not on a fixed scheduled and are done if necessary or requested by the stakeholder.

12.5 Retro

Instead of following the Keep-Try-Drop model for the Sprint Retrospective, the project team decided to focus on successes, problems, and improvements. A reason for that is the focus on the individual by the team due to the low number of members. This focus on successes and problems provides a way for each team member to explain their frustrations and discuss possible improvements. This adaptation is comparatively small and mostly one of wording. The project team feared however, that using the Keep-Try-Drop model, they would have to justify using certain tools when running into minor problems. Instead, the team hopes to focus more on the product than the tool chains.

13 Project Setup Review

13.1 Identifying the initial situation

For identifying the initial situation we analysed 21.1 Project description and used multiple online resources such as the websites of Lärmliga Schweiz and Federal Office for the Environment FOEN. Based on those information we were able to derive some of the requirements and specifications for our application, which now provides a convenient way for affected people to document noise pollution.

13.2 Topic analysis

To solve the issue at hand, one needs knowledge about how audio recoding and measurement works. We invested the time needed to analyse the topic and gather the know-how needed to process and analyse audio files. A short summary of the problematic can be found under 4.3 Problems with Audio Files. This analysis provided the technical basis for achieving the product goal.

13.3 Stakeholder / Stakeholder Management

Initially we identified a list of possible stakeholders:

- Tutor: Dr. Simon Kramer
- People affected by noise pollution
- Noise producers (Construction Sites, Night Clubs, Highways)
- Lärmliga Schweiz
- Federal Office for the Environment FOEN

For the context of this module we decided to only consider our tutor as the sole stakeholder, because otherwise the stakeholder management would cost much more time without much benefit. This way the stakeholder management turned out to be straight forward, as Dr. Kramer provided his expectations and the scope of the project beforehand, only impediments and key decisions had to be discussed together.

13.4 Organisation

The project organisation and how the project team implemented scrum is documented under the following three chapters:

- 9 Scrum Roles: The distribution of the Scrum Roles has been proven practicable, even though some of the project members didn't have any experience in their respective roles
- 12 Scrum Adaptations: The project team was really happy about the way we adapted Scrum in this project as it was really product focused and did not introduce too much overhead
- 11 Requirements: We found the definition of ready and the definition of done were especially helpful, as it made sure the tickets had a minimum quality, a clear achieveable goal and are well understood by the team.

Further the project team mainly worked together remotely which reduced the unnecessary over head of commuting.

13.5 Installations

The project team used the following tools for carrying out project 1:

- Project Management: GitLab
- Version Control (Code and Documentation): GitLab
- Documentation and Presentation: LaTeX
- E-Mail: MS Outlook
- Team communication (Chat and Video Calls): MS Teams
- Visualizations: Excalidraw
- Diagrams: draw.io

14 Review

14.1 Product goals

The following list summarizes the 4.2 Product Goals:

1. Analyze Audio File
2. Summarize findings in a PDF
3. Easy to use

From our point of view, the first two product goals are clearly achieved, as we built a working application which can parse and analyze audio files and then summarizes the findings in a PDF document. The third goal of 'easy to use' is not as straightforward to measure, but we argue that this is achieved by considering the following list of best practices:

1. Cross-platform and no download/installation required: Achieved by building a public web application
2. No login or personal information required
3. Clarity, consistency, responsiveness and familiar patterns: Achieved by using Bootstrap, from Twitter

4. Help and documentation: Achieved with integrating tooltips where needed and show human-readable error messages

5. Performance and fast load times: Achieved by realizing the application as a minimal SPA and preloading all necessary LaTeX resources in the background while the user is filling out the form

14.2 Sprint goals

The sprint goals were defined in their respective sprints and are documented under 10 Sprint Goals. We often did not fully achieve the defined sprint goals in the respective sprint because we underestimated the amount of work going into solving a ticket, especially the effort for the documentation. So even though on paper it looks like we did not reach our goals, we made good progress in each sprint and finalized the goal in the followup sprint. Keep in mind that the sprint goals were achieved implicitly and are not mentioned again in the followup sprint. In other words, we still achieved all sprint goals, just not in the sprint they were defined.

14.3 Product delimitation

The product delimitation is split into 5.1.1 System Environment (statics) and 5.1.2 Process Environment (dynamics). With the help of these two delimitations, we were able to clearly differentiate what is relevant for the product and what is out of scope.

14.4 Deliverables

All deliverables were specified in their respective tickets and according to our definition of done they must be documented in the project report. We do not list all deliverables here, as they can be viewed as part of the projects report: Project Report. Our main deliverable is the final application, which is available online under the following link: dB threshold event displayer.

14.5 Product backlog / Sprint backlog

The product and sprint backlogs are documented under 11.3 Product Backlog and 11.4 Sprint Backlog respectively. The project team was happy to use them as defined. Even though there are still tickets left in the product backlog, we have no open tickets with the priority medium or high. This means we achieved a minimal viable product within the module's time frame.

15 Retrospective

15.1 Retrospective I: Scrum roles and stakeholders

- Product Owner: Dominic already has quite some experience with Scrum and thus made a perfect job from the beginning.
- Scrum Master: Even though Lukas held the role for the first time he could fulfill it after a bit of initial support from Dominic.
- Developers: Thanks to the clear definition of ready and definition of done, the developers always knew what to do in every task.
- Stakeholder: The project team enjoyed working with Dr. Kramer as the expectations were communicated beforehand and clearly.

15.2 Retrospective II: Scrum Events and Artifacts

- Product Backlog: The Product Owner did a perfect job with specifying and managing the product backlog. This helped the team to progress without any blockers.

- Sprint planning meeting: The planning was always really smooth, as it was clear which the logical next steps are.
- Sprint backlog: The Sprint backlogs were always a bit overambitious, as we almost never achieved all the sprint goals.
- Daily Scrum meetings: The 'Dailies' were actually held once a week, nevertheless they helped us to resolve issues on the current tasks and kept all team members in sync.
- Burndown Charts: As we did not work on the project every day, the burndown charts looked not like a gradual line downwards but instead showed a clear trend of the team doing work on weekends and within the time slot of the Proejct 1 module.
- Finished work: As we had a clear definition of done, the finished work was always documented in the projects report.
- Sprint Review: The project team really enjoyed the sprint reviews, as it made the progress visible.
- Sprint Retrospective: At the start of the project, we decided to use a minimal approach for the retrospective and the project team was really happy with this decision.

15.3 Retrospective III: Tools/Instruments

- GitLab: The project team was not always happy with the workflows provided by GitLab, such as the issue boards and the differentiation between group and project.
- MS Teams: This tool is an amazing, unified communication channel for working together, but the Linux support could be better.
- MS Outlook: MS Outlook just works as expected.
- LaTeX: Even though we did not have a lot of experience with LaTeX, we decided to use it for all documentation and presentation purposes. The project team enjoyed working with LaTeX, but for non-academic contexts we would have chosen less complex tools such as MS Word and MS PowerPoint.
- Excalidraw: This easy to use online tool helped us to create a lot of the diagrams and UI-Prototypes.
- draw.io: Another easy to use online tool for creating diagrams where the results look a bit more old-school / professional.
- Controlling:
  - Burndown Chart: The Burndown chart was a good visual indicator to see how the sprints progress is going, but as we often underestimated tasks it could be quite misleading for an external observer.
  - Product Backlog: Thanks to the detailed initial product backlog we could always see where we stood in the project.

16 Lessons learned

16.1 Insights into framework conditions

For the team members which already had quite some experience with Scrum, there was still the learning curve of setting up everything from scratch. The other member had quite some experience with alternative forms of agile software development and could learn a lot about the Scrum specific methods. In our point of view, the most important Scrum ceremonies were the sprint review and planning which we always held subsequently. The sprint review resulted in some interesting demos, kept the whole team in sync, showed the current progress, and already

indicated what the next steps could be. On the other hand there is no sprint without the sprint planning and after it, every team member knows what he has to do next. In the context of project 1 the key takeaway for the project team is, that Scrum also works when nobody works full-time on the project and all the work is done remotely.

16.2 Cooperation

16.2.0.1 Team:

The key challenge for the project team in terms of cooperation was the limbo between working, other modules, and project 1. This resulted in the reality that we only worked together in the modules official time block. In the beginning a project member had issues being on time for meetings, but luckily he could resolve this issue. Besides the minor challenges and issues, the team work was amazing and we enjoyed working together.

16.2.0.2 Specialist advisor / Stakeholder:

We consider the cooperation with Mr. Kramer as good, as he communicated his expectations clearly from the start. If we had issues or questions we received an answer within reasonable time or could schedule a meeting with him in the modules official time block.

16.2.0.3 Project Management Coach:

The cooperation needed with the project management coach was minimal as all requirements were clearly documented in moodle. When something was unclear we received an answer within reasonable time.

Part III
Conclusion

17 Conclusion

We learned a lot about how audio works and to process it. We learned especially a lot about how the wave file format or rather the riff format family works, because we implemented our own parser for that.

17.1 Review

We initialy struggled with the task of filtering the audio with a given threshold. This was due to the fact that the audio samples in wave files represent relative loudness and not absolute. And to filter the audio we need absolute values. After researching we found out that we can map the values from the samples to absolute ones when we have some reference values. We also had issues with the parsing of individual samples because of the way that javascript handles numberic values. In the end we were able to fix this by using the appropriate functions to read bytes. Besides does minor difficulites, we did not encounter any issues which could endanger the achievement of our targets. The project management with Scrum gave a noticable overhead for the whole project and we spent a good amount of time every week to hold the scrum meetings. We think that this was a bit to much for a project of this size. But never the less it helped us to manage and prioritize our tasks and to focus on independent features to implement.

17.2 Bottom Line

Lorem ipsum

17.3 Future Work

Lorem ipsum - Automate Latex dependencies management

18 Glossary

- dB(A): a decibel value with a scale relativ to the human ear.
- RMS: root mean square

19 Index

Lorem ipsum

20 Bibliography

References

[1] Federal Office for the Environment FOEN. Noise and vibrations: In brief. https://www.bafu.admin.ch/bafu/en/home/topics/noise/in-brief.html. Accessed: 2024-11-4.

[2] how to know the real-world db level of a file? https://community.adobe.com/t5/audition-discussions/how-to-know-the-real-world-db-level-of-a-file/m-p/13939424. Accessed: 2024-11-4.

[3] How can i calculate audio db level? https://stackoverflow.com/questions/2445756/how-can-i-calculate-audio-db-level. Accessed: 2024-11-4.

[4] Extracting sound pressure from wav file. https://dsp.stackexchange.com/questions/35250/extracting-sound-pressure-from-wav-file. Accessed: 2024-11-4.

[5] user545125. Answer to "how can i calculate audio db level?". https://stackoverflow.com/a/4464385. Accessed: 2024-11-02.

[6] Audacity Team. Audacity. https://www.audacityteam.org. Accessed: 2024-11-02.

[7] Audacity Team. Audacity amplify. https://manual.audacityteam.org/man/amplify.html. Accessed: 2024-11-02.

[8] Decibel X (iOS). https://apps.apple.com/de/app/decibel-x-db-sound-level-meter/id448155923?l=en-GB. Accessed: 2024-11-02.

[9] Decibel X (Android). https://play.google.com/store/apps/details?id=com.skypaw.decibel&hl=en&pli=1. Accessed: 2024-11-02.

[10] Pgfplots gallery. https://pgfplots.sourceforge.net/gallery.html. Accessed: 2024-10-22.

[11] TeX Users Group. TeX Live license. https://tug.org/texlive/LICENSE.TL. Accessed: 2024-10-22.

[12] Free Software Foundation (FSF). FSF Free Software definition. https://www.gnu.org/philosophy/free-sw.html. Accessed: 2024-10-22.

[13] Christian Feuersänger. pgfplots. https://ctan.org/pkg/pgfplots. Accessed: 2024-10-22.

[14] Latex.wasm: Latex engines in browsers. https://www.swiftlatex.com/. Accessed: 2024-10-22.

[15] Swiftlatex github repository. https://github.com/SwiftLaTeX/SwiftLaTeX/. Accessed: 2024-10-22.

[16] Wav file format. https://en.wikipedia.org/wiki/WAV. Accessed: 2024-11-5.

[17] Timespan for calculating audio rms. https://majormixing.com/what-is-rms-in-audio-world/. Accessed: 2024-11-5.

[18] Decibel wikipedia. https://en.wikipedia.org/wiki/Decibel. Accessed: 2024-11-5.

[19] Chrome can't load web worker. https://stackoverflow.com/questions/21408510/chrome-cant-load-web-worker. Accessed: 2024-10-28.

[20] Ken Schwaber and Jeff Sutherland. The 2020 scrum guide. https://scrumguides.org/scrum-guide.html. Accessed: 2024-11-03.

[21] Carlton Nettleton. Is the product owner part of business or part of it? https://appliedframeworks.com/blog/is-the-product-owner-part-of-business-or-part-of-it. Accessed: 2024-11-03.

[22] Frank Helbling. BTI3031 - Intro Scrum (part 3). https://moodle.bfh.ch/pluginfile.php/3229204/mod_resource/content/11/BTI3031%20-%20Intro%20Scrum%20%28part%203%29.pdf. Accessed: 2024-11-03, authentication required.

21 Appendix

21.1 Project description

The **goal (what)** of this project is to deliver a FLOSS-licensed, platform-independent piece of software (computer program), called the *Decibel Threshold Event Displayer*, that

1. takes as inputs a WAV-file and a list of sound level thresholds in decibels (e.g., legal day and nighttime noise maxima above which your health deteriorates);
2. filters out all data points in the file that correspond to sound events below the lowest of the above thresholds; and
3. displays the remaining data points (as a blue vertical comb plot) on a horizontal time axis (with the dates and times corresponding to the data points) as well as the thresholds (as horizontal red lines) in decibel, and statistically summarises the data set with the help of the LaTeX-package pgfplots.

The **purpose (why)** of this project is to empower poor folks who suffer from insomnia due to ambient noise (https://laermliga.ch/) by arming them with the (peaceful) means of proving their noise hell (a smartphone app such as https://apps.apple.com/ch/app/dezibel-x-pro-l%C3%A4rm-messger%C3%A4t/id1257651611 together with your software) to the police and the courts of law.

The code should be minimal, modular, and self-explaining.

The project report should be concise (maximally informative, minimally long). It must contain this project description as a quotation.

21.2 Declaration of Authorship

The project team, namely Dominic Gernert, Lukas von Allmen, and Darius Degel, hereby declare that the report submitted is our own unaided work. All direct or indirect sources used are acknowledged as references.