Coursework Part 2 Report

Samuel Cattanach
40276600@live.napier.ac.uk
Edinburgh Napier University - Advanced Web Technologies (SET09103)

1 Introduction

During the planning stage in the first part of the coursework, I proposed a plan for a flask application that would allow users generate and configure the frequency bands of white noise. Users would be able to sign up to the website using a username and password and then create and save these sounds. Users would also be able to browse other users' profiles and the sounds that they have created. The main purpose of the website was to provide the user with the ability to design their own white noise sound in order to help their focus when studying or working.

2 Deviations from Initial Plan

The initial proposal for the flask app included many features and Technologies that added to the functionality and robustness of the website. As this plan was very comprehensive and the time available in order to implement such features was a very limiting factor the end result deviates from the initially proposed application in several ways.

With the main functionality of the website being the ability to create sounds and browse sounds from other users, this core functionality has been implemented as shown in figure 1. All users are able to view a list of registered users and can click on a user to visit their profile page, from their they can click on that users sounds to listen to them. This main functionality has not deviated from greatly from the initial plan apart from in two aspects. Firstly, many design elements of the user interface have been altered and rearranged in order to improve the user experience. Secondly, the option to change the sound to rain or waves for example is not available as implementing this additional feature was not crucial to the user experience or objectives of the website.

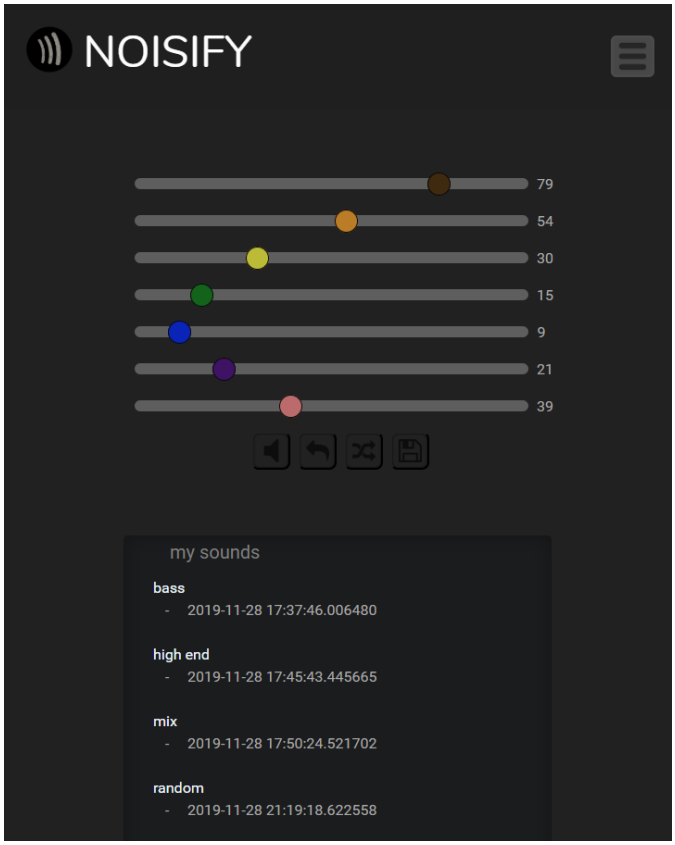

Figure 1: **Sound configuration** - white noise frequency sliders

Another aspect in which the completed website differs from the plan is in providing a data API for people or third-party services to retrieve either sounds or users from the database without needing to navigate the html site in order to do so.

Although a MySQL database was specified in the plan, the database which was actually implemented was an SQLite database. This change was made because using an SQLite database offers the benefit of being "fully self-contained, meaning there aren't any external dependencies you have to install on your system for SQLite to work" (Drake, 2019)[1]. Compared to MySQL, SQLite is simpler to set up as there are less elements and dependencies which could fail.

In the original plan of the relational SQL database, the 'Sounds' database included a 'favourites' column which contained details of how many users have added the specific sound to their list of favourites. This feature was not implemented due to time constraints and its lower priority compared to other more significant features (Fig 2).

Similarly to the sounds' favourites, each sounds' views were not included in the final implementation as a result of the time constraint. Ideally, when browsing the lists of sounds available on each users profile or on the '/sounds' route, the number of times the sound has been played would also be displayed next to the number of times it has been added to users' favourites lists.

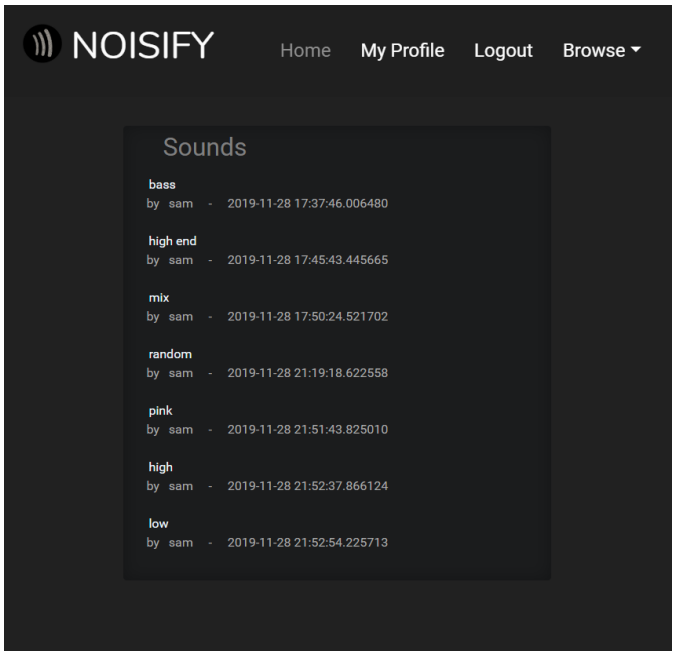

Figure 2: **Sound list** - all saved sounds by users

3 Possible Additions & Improvements to Implementation

In the event that more time was available in order to complete development of the website there are aspects in which possible improvements could be made in order to improve the overall user experience of the website and to attain a closer result to what was first proposed.

Shown in figure 3 is the method in which a user saves their configured sounds to their profile. This method involves the details of the sound being copied to the user's clipboard and requires the user to paste such details into the form provided. An obvious way in which the user experience of this process could be improved is by removing the need for the user to paste such details manually and to automate this aspect. Although the purpose of this function Is still currently achieved it would slightly aid in the flow of the website.

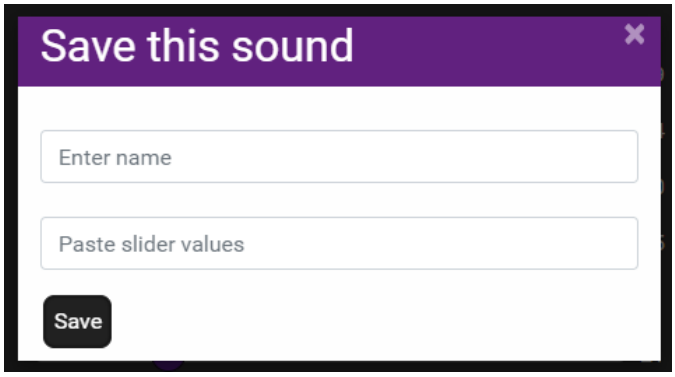

Figure 3: **Sound saving dialog** - user enters name and pastes the copied slider values

In addition to the previously described improvements the site could also be deployed using a uWSGI web server in order to be more accessible to users. Deploying the application in this method would allow for more robustness as the web server could be automatically restarted upon any server-side errors which crash the application.

4 Complications & Accomplishments during Implementation

Throughout the development of this application there have been many setbacks and impediments as I attempted to implement the initial plan with the proposed improvements described in the previous section. The aspect of the website which produced the most difficulty was arguably the sound configuration functionality. Allowing the users to move the seven sliders in order to change the volume of different frequency ranges of white noise was more challenging than I originally prepared for. As a result of the challenges faced during the development of this feature, time which may have otherwise been spent implementing additional features was lost. attempting several different approaches and techniques before a solution was found was a considerable challenge which was overcome after a substantial amount of time was spent on it.

Although many challenges were faced during the development phase of the coursework, many achievements were made in actualising the proposed features and user-interface of the website. The implementation of the process of navigating the site through links was achieved as planned. Users are able to navigate to any page of the site with ease given no guidance as the hierarchical structure is intuitive and memorable. Shown in figure 4, the navigation bar at the top of every page has links to the home page, profile page, users and sounds page, and has an option to log out.

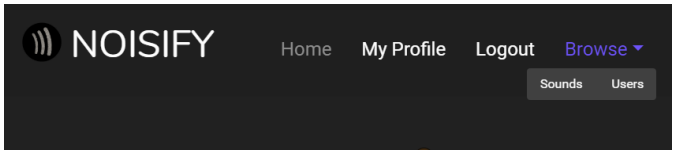

Figure 4: **Navigation bar** - links to parent pages

5 Use of Additional Software

In order to actualise the features and functions of the flask application I utilised additional flask plugins and JavaScript libraries.

Howler.js

To obtain functional white noise sounds I chose to use the JavaScript library Howler.js. This library is used by companies such as Google and Dolby to make ‘’working with audio in JavaScript easy and reliable across all platforms” (howlerjs). I chose to use this library instead of developing my own functionality as it allows audio to be ‘’automatically cached and re-used on subsequent calls for better performance and bandwidth” (howlerjs)[2].

Flask-Login

In order to handle authenticating and authorising users on the site I used Flask-Login. The benefit of using Flask-Login instead of implementing my own system is that it allows ‘’securing parts of [the] app behind login walls, encrypting passwords, and handling sessions” easy to manage and add (hackersandslackers)[3]. Flask-Login also integrates very well with Flask-SQLAlchemy which is also used within my flask application.

Flask-SQLAlchemy

Flask-SQLAlchemy is a Object Relational Mapper flask extension which was used in order to simply protect the database against sql-injection attacks and to add and retrieve rows from the SQLite database.

In order to hash users passwords as they are entered based on the secret key the werkzeug.security package was used as it provided a useful API for both generating and checking hashed passwords from the SQLite database

Bootstrap

In order to aid in achieving a consistent color scheme and user interface elements Bootstrap was used in all HTML pages. Bootstrap is very widely used as it is easily integrated into any site and allows for a significant amount of time to be saved on basic JavaScript functionality and the aesthetics and dynamism of HTML elements. As Bootstrap is commonly used for mobile-first development it has proved helpful in assuring my site is durable across different devices. In addition to using Bootstraps predefined classes and functions I also wrote a large portion of the CSS and JavaScript myself.

References

[1] DigitalOcean, “Sqlite vs mysql vs postgresql: A comparison of relational database management systems,” Sep 2019.

[2] “howler.js - javascript audio library for the modern web.”

[3] T. Birchard, “Authenticating users with flask-login,” Dec 2019.

Appendices

Appendix A

Some of the technologies described in the use of additional software section are not within the default learning environment and can be install into a python virtual environment using the following commands:

```
1 $ virtualenv venv
2 $ source venv/bin/activate
3 $ pip install -r requirements.txt
```