|  |
| --- |
| House Party - CI601 The Computing Project Report |

|  |
| --- |
| By Alexander wood – 22827042  Supervisor – Jarod Locke  Second reader – Andrew Blake |

Contents

[1. Introduction 4](#_Toc195640764)

[1.1 Motivation 4](#_Toc195640765)

[1.2 Project aims and objectives 4](#_Toc195640766)

[1.3 Minimal viable product 4](#_Toc195640767)

[Sign in to Spotify account 4](#_Toc195640768)

[Create and manage a party 4](#_Toc195640769)

[Join a party 4](#_Toc195640770)

[Add songs to the host’s queue 4](#_Toc195640771)

[2. Methodology 5](#_Toc195640772)

[2.1 Project management techniques 5](#_Toc195640773)

[2.2 Testing techniques 5](#_Toc195640774)

[White box testing: 5](#_Toc195640775)

[Black box testing 5](#_Toc195640776)

[Unit testing: 5](#_Toc195640777)

[2.3 Development tools and software 6](#_Toc195640778)

[Visual Studio Code (VS Code): 6](#_Toc195640779)

[GitHub: 6](#_Toc195640780)

[GitHub Projects: 6](#_Toc195640781)

[Brighton Domains: 7](#_Toc195640782)

[Draw.io: 7](#_Toc195640783)

[OneDrive: 7](#_Toc195640784)

[Playwright: 7](#_Toc195640785)

[Bootstrap and Font awesome icons: 7](#_Toc195640786)

[Qrcode.js: 7](#_Toc195640787)

[Google fonts: 7](#_Toc195640788)

[Lighthouse: 8](#_Toc195640789)

[2.4 Project risk analysis 8](#_Toc195640790)

[3. Research 8](#_Toc195640791)

[3.1 Literary review 8](#_Toc195640792)

[Spotify API terms of service: 8](#_Toc195640793)

[Spotify Web API documentation: 9](#_Toc195640794)

[Spotify Design & Branding Guidelines: 9](#_Toc195640795)

[Apple Music API documentation: 9](#_Toc195640796)

[Soundcloud API documentation: 9](#_Toc195640797)

[YouTube documentation: 9](#_Toc195640798)

[3.2 API research 9](#_Toc195640799)

[3.3 Proof-of-concept 10](#_Toc195640800)

[3.4 User feature brainstorming group 10](#_Toc195640801)

[3.5 Similar services 10](#_Toc195640802)

[3.6 User testing 10](#_Toc195640803)

[4. Product description 11](#_Toc195640804)

[4.1 Planning 11](#_Toc195640805)

[4.2 Designing 11](#_Toc195640806)

[4.3 Prototyping 11](#_Toc195640807)

[4.4 Developing 12](#_Toc195640808)

[4.5 Problems during development 12](#_Toc195640809)

[Login and verification issue: 12](#_Toc195640810)

[API time miss match: 13](#_Toc195640811)

[Spotify extension submission: 13](#_Toc195640812)

[Unregistered user being able login: 13](#_Toc195640813)

[Tab functions unresponsive in popups 13](#_Toc195640814)

[4.6 Limitations 13](#_Toc195640815)

[Brighton domains: 13](#_Toc195640816)

[Time constraints: 14](#_Toc195640817)

[Real world testing: 14](#_Toc195640818)

[4.7 Final product 14](#_Toc195640819)

[Homepage: 14](#_Toc195640820)

[404: 15](#_Toc195640821)

[Loginerror: 15](#_Toc195640822)

[Join: 15](#_Toc195640823)

[Dashboard: 15](#_Toc195640824)

[Party: 15](#_Toc195640825)

[Critical review 16](#_Toc195640826)

[Conclusion 16](#_Toc195640827)

[References 17](#_Toc195640828)

[Appendices 19](#_Toc195640829)

[Appendix 1: 19](#_Toc195640830)

[Appendix 2: 20](#_Toc195640831)

[Appendix 3: 20](#_Toc195640832)

[Appendix 4: 21](#_Toc195640833)

[Appendix 5: 21](#_Toc195640834)

[Appendix 6: 22](#_Toc195640835)

[Appendix 7: 22](#_Toc195640836)

[Appendix 8: 23](#_Toc195640837)

[Appendix 9: 23](#_Toc195640838)

[Appendix 10: 24](#_Toc195640839)

[Appendix 11: 24](#_Toc195640840)

[Appendix 12: 25](#_Toc195640841)

[Appendix 13: 25](#_Toc195640842)

[Appendix 14: 26](#_Toc195640843)

[Appendix 15: 27](#_Toc195640844)

[Appendix 16: 28](#_Toc195640845)

[Bibliography: 29](#_Toc195640846)

# Introduction

## Motivation

This idea emerged from the fact that there are multiple music streaming services, each with its own method of allowing people to share and control each other’s music, but these platforms do not support each other. At a party, there is no way to invite someone who uses a different platform, hence the concept of creating a website that allows people to add songs to the queue while eliminating the need for a platform-specific account or app.

## Project aims and objectives

* The website should have a simple and easy to use design
* The website should be accessible
* The website should store any sensitive data securely
* Host can invite guests to their hosted parties through multiple methods
* Guests can join parties easily
* Guests can search for songs
* Guests can add songs to the host’s queue
* Guests don’t need a Spotify account
* Host can restrict what songs are added such as not allowing explicit content or blocking duplicate songs
* Host can invite people using a QR code

## Minimal viable product

### Sign in to Spotify account

Allow the user who wants to host a party to login into their Spotify account easily.

### Create and manage a party

Allow someone who is logged into their Spotify account to create a party which meets the needs they want for the party and manage the settings of the party.

### Join a party

Allow guests to join a session hosted by a user that they have been invited to participate in.

### Add songs to the host’s queue

Allow guests using the party to add songs to the hosts queue through the website

# Methodology

## 2.1 Project management techniques

The project methodology which was used to design this website is the waterfall methodology. This will be used for the design process to layout the requirements and the features that will and won’t be in the final product. Then when it comes to the development process of the website, the waterfall methodology will be combined with a kanban board to help manage and know what is being worked on and what’s left to be worked on.

## 2.2 Testing techniques

While developing the website there were multiple different ways of performing tests to help find and fix bugs and improve the UI for the user’s experience the testing types that were used are white box, black box and unit testing each having their own benefits.

### White box testing:

White box testing was performed as development happened and was used to testing elements of the system as they were being implemented to make sure they were being implemented properly and worked as expected before the changes were pushed to the GitHub.

### Black box testing

Black box testing was performed by taking a small group of users and having them use the website. This testing was done to further identify bugs and issues that could arise while using the website. This testing also allowed for changes to the UI and new features to be discussed further, helping to improve the website and its usability.

### Unit testing:

In addition to all the tests that are being performed using human input I also put into place automatic testing of the pages that performs checks to make sure the page acts as it should depending on different variables, I built these tests using a package called playwright. An example of a test created can be seen in **Appendix 2** and the results page that shows how the results of the tests on different platforms and browsers can be found in **Appendix 3** this data is then further broken down so you can find where failures occur in the code which can be seen in **Appendix 4**.

Some tests have been disabled due to security put into place in the API preventing any external users from accessing the API this means the tests can’t be run locally and must be run on the server.

I have also linked the automatic testing to the GitHub making it so that when a pull request is made to merge the changes from the dev branch into the stable branch the tests are run to make sure that there are no changes that have broken any functionality of the website.

## 2.3 Development tools and software

### Visual Studio Code (VS Code):

For the coding and development of the project I have decided to use the programming Integrated development environment (IDE) VS Code. I chose to use this IDE as I have experience using it and it also allows for a lot of customization to your preferences allowing me to make it easier for me to use and understand.

### GitHub:

To store a backup of my code and also allows for version control this allows me to create a repository and upload all my code and files to it storing it securely and also allowing me to create interaction of the code when it comes to adding features or changing features that could cause major issues giving me the opportunity to undo the changes I have done.

By using GitHub, it allows me to control changes through branches allowing me to separate new changes from a working and functional build by having a stable branch and a dev branch. Once testing is done and I know all the changes pushed to the dev branch are functional, I can then merge these changes into the stable branch.

### GitHub Projects:

To keep track of the project GitHub projects was used as it allows for both the code and progress tracker to be kept together as both are stored on GitHub while also offering the same features as other project progress trackers *(GitHub No date)*.

A screenshot of a computer

AI-generated content may be incorrect.

### Brighton Domains:

To host the website, the decision was made to use Brighton domains. This was done as it is already free to use and accessible allowing for the website to be hosted on it straightway with no delay. It also allows for the backend to also be hosted straight away as it supports all the required tools. These include automatic PHP files, database events, databases and the ability to host HTML, JS and CSS meaning everything can be kept together on one server.

### Draw.io:

To make wireframes for the website draw.io was used as it offers a wide range of features while also being free, easy to use and accessible online meaning no software must be installed making it easy to switch between devices.

### OneDrive:

To make it easy to work across multiple devices and to provide an extra backup of the work OneDrive was used as it provides active backups meaning any changes being made to the project and automatically backed up to the cloud without needing any input providing an extra level of security to the files. OneDrive also has benefits of you being able to sync multiple devices to one OneDrive, meaning any file changes on one device are then updated on the second device allowing you to switch between the two easily.

### Playwright:

To perform unit testing on the website and its front-end functionality Playwright was used to write automatic tests which perform a range of checks and then returns the outcome from the tests in a user-friendly way, making it easy to understand.

### Bootstrap and Font awesome icons:

The font awesome and Bootstrap icon libraries were used throughout the website. These icon libraries are easy to use and provide a wide range of icons to use along with built in customizations for the icons *(Font awesome No date) (Mark Otto, J.T. No date)*.

### Qrcode.js:

To generate QR codes for the website which are used to invite guests to a party a package called Qrcode.js was used. This package is a free to use package which was downloaded and hosted locally on the server with the website to make accessing it easy *([davidshimjs](https://github.com/davidshimjs) No date).*

### Google fonts:

The website uses a custom font called Roboto which was downloaded from google fonts and hosted locally with the website.

### Lighthouse:

To help test the website performance and accessibility a tool called lighthouse as used which is a browser extension which runs a set of tests on the website you have open returning statistics on load time, performance and accessibility while also returning this data it also provides information about areas which could be improved to improve the scores *(google 2016).*

## 2.4 Project risk analysis

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk Description** | **Likelihood of risk** | **Impact of risk** | **Action taken** |
| Spotify API outage | LOW | HIGH | No action has been taken for this as there is nothing that can be done for this risk |
| Brighton domains outage | LOW | HIGH | No action has been taken for this as there is nothing that can be done for this risk |
| Externally hosted packages becoming unavailable | LOW | HIGH | To prevent any issues from external packages becoming unavailable instead of using third party hosting services the packages are hosted locally with the website |
| Exceeding Spotify API request rate limit | MEDIUM | LOW | There is no way to prevent this issue from occurring but instead there have been a process put in place to manage exceeding the rate limit, so the user experience is not affected |
| Loss of the project files from file corruption etc | LOW | HIGH | To prevent this from affecting development there were multiple backups put in place. These include GitHub, OneDrive and a copy hosted on Brighton domains allowing for the code to be recovered |
| A bug being pushed to the stable branch and hosted on the servers | LOW | MEDIUM | To prevent this issue from occurring all the code goes through multiple tests and checks before leaving the dev branch of the GitHub this stops any code containing bugs from being hosted on the servers |

# Research

## 3.1 Literary review

### Spotify API terms of service:

This website breaks down the terms of service for the available APIs provided by Spotify showing what’s allowed to be made using the APIs and what isn’t. This website was easy to navigate and easy to understand making it extremely easy to check that what’s being made meets the requirements of the terms of service.

### Spotify Web API documentation:

This website breaks down the functionality of the Spotify web API showing how to use it and how to gain access to it. The documentation in some areas was complicated and took a lot of time to understand fully, but in some areas of the website it allows for you to try out API requests showing you what attributes you can send followed by showing you what the returned data looks like.

### Spotify Design & Branding Guidelines:

This website broke down the requirements of the website regarding how it should look and what needs to be displayed on the website for it to be approved and taken out of development mode by Spotify. This website was clear and easy to understand giving examples and screenshots of what is expected to be shown on the website while also providing downloads and access to required content such as different Spotify logos for different scenarios.

### Apple Music API documentation:

This website explained the capabilities of the Apple Music API and how to gain access to the API functions. The website was quite comprehensive but that made it complicated at first to understand how to navigate and understand the website but after exploring the website it becomes easier to navigate and understand.

### Soundcloud API documentation:

This website broke down and explained the capabilities and functionality of the Soundcloud API. The website is quite simple and provides as much information as possible while making it clear with snippets of code and screenshots making it extremely easy to read and understand.

### YouTube documentation:

This website explained the functionality and capabilities of the YouTube API. This website can be simple in some areas but in others it can become quite complex and hard to understand but it does provide screenshots of example returned data which helps to understand the expected outcome of using the API functions.

## 3.2 API research

Before development started a lot of time was put into researching the Spotify API and its functionality to find out how to make the Spotify login work and use it in the website this was also followed by research into how other functions from the API which are required for the website work and how they should be implemented in *(Spotify No date)*.

Once research around Spotify was completed, research around other music streaming platforms began to figure out if it would be possible to add support for multiple platforms to the website but the research done proved that it would not be possible to add support for other platforms as they do not have the functions required for the website to work built into their APIs making it impossible.

## 3.3 Proof-of-concept

To learn and figure out how to use the Spotify API a proof-of-concept Node.JS was made to help learn how to use the API to perform the actions required for the website. This also helped to make sure it was possible to make what was envisioned *(Spotify No date)*.

## 3.4 User feature brainstorming group

To help find out what people thought about the idea a small group of people was put together and asked three questions the questions were would you use this service, what features would you want this service to have and when would you use this service. This small group was used to test if the idea would be used and what features something like this would need for people to be interested, the ideas they gave as features were then noted down and kept as ideas for the project some features found their way into the final product but most due to time were not able to be added due to them needing a lot of work.

## 3.5 Similar services

Spotify offers a similar service called Spotify jam which allows users to invite people to control their music but offers the guests more control over the music allowing the guests to remove, add, pause and skip songs these features a liked but can become an issue because of all the users having full control people can remove or skip other people’s songs. To use Spotify, jam all users joining the session require a Spotify account and access to the Spotify app without these people are unable to join due to needing a Spotify account this can restrict who can join and add songs to the queue due to people using multiple different music platforms *(Spotify no date)*.

## 3.6 User testing

Before the final prototype was finished a small group of users were put together and asked to use the website this was done to see how users would interact with the website so that the UI could be improved where it needed to be and how It can be improved to make it more user friendly this testing also helped identify bugs that were not found during unit testing and Whitebox testing.

# 4. Product description

## 4.1 Planning

To plan the development of the website the MSCW method was used to help prioritize the features and functionality to be added to the project. This method categorizes the importance of each feature into four different categories: must have, should have, could have, and won’t have *(ProductPlan 2024)*.

A screenshot of a computer

AI-generated content may be incorrect.

After the MSCW was made it was then taken and used to create a progress tracker which included the main functionality and design of the website but also listed the features that were to be added to the website separate from the main functionality this progress tracker was also used to line out the order in which parts of the website were going to be made in a screenshot from the GitHub pages can be seen in **Appendix 9.**

## 4.2 Designing

Following the planning phase work on the design of the website was started, the first step was to create a simple sketch of the layout and where things would go on the page. These sketches were just black and white. An example can be seen in **Appendix 5**.

The basic sketches were then taken and made into wireframes using draw.io adding more details and flushing out what the user interface would look like including features. This can be seen in **Appendix 6**.

## 4.3 Prototyping

During the development process of the website there were 2 prototypes: the first prototype was used to evaluate the development of the backend of the website. During this process a simple UI was introduced which allowed for easy testing. The second prototype was to develop the UI to finalise the design for users. These 2 prototypes were separated as it allowed for focus to be on a specific section of the website, allowing for focus to be on one thing at a time thereby improving performance and allowing for testing to be done on a specific part at a time allowing for more accurate testing.

## 4.4 Developing

The development of the website was broken down into pages due to the website being built using a micro service architecture this meant that each page was being focused on independently and only one page was being focused on at one time. This worked well as none of the pages depended on each other as they all worked independently. Alongside the development of the pages the API was also being worked on. This is because all the pages used different API functions due to them performing different actions. This type of development also applied to the micro services, for example the ModalHandler. This service was being used by multiple pages, meaning it was being added to and modified during the development of other pages to add more functionality to it but to also enhance and improve the service.

Once development of both the front end and back end of the website was done and both were combined into the final functional product this was when improvements and enhancements were made to both components improving performance, adding more debugging, adding more error handling, improving maintainability, and improving functionality this was mainly done for the backend as during the process of making the front end areas for improvement were found in the API’s.

## 4.5 Problems during development

### Login and verification issue:

A major problem that arose during the development of the website involved the information used to verify the hosts identity which is both stored in the hosts cookies and the database. When a user logs into the website a refresh token is returned to the server. This combined with the hosts user id is used to verify their identity before performing any involving managing or creating a party. But what was noticed is that occasionally the refresh token that was returned when logging in would change from the previous one provided, meaning that sometimes if a user created a party and logged out, they would then be unable to manage their party.

The solution to this was to add a check into the API that handled the login. The check would see if the user has created a party with a refresh token already and if the new refresh token did not match the old one the user would instead have the old one added to their cookies instead of the new one, making no difference to the users experience but making it so the verification issue would not happen again.

### API time miss match:

An issue which affected the development was a time mismatch. The website was designed to use the UK time zone to work out when to close expired parties, but parties would always end an hour before they should. With further investigation it showed that the server was always an hour behind. The solution to this was when the expiry time is stored an extra hour is added on to counter the issue, allowing everything to work as intended.

### Spotify extension submission:

Once there was a working prototype the website was submitted to Spotify to be taken out of development. This process can take a minimum of a month to be completed so a submission was made early so any issues found could be fixed. The submission that was made was declined due to an issue with how the Spotify logo was being displayed and that there was no link to the song on Spotify. These issues where fixed quickly once they were brought to my attention.

### Unregistered user being able login:

An issue was discovered involving the Spotify login API this issue involves the Spotify login API allowing none registered users from logging into the website this causes the websites functionality to be unresponsive because to use the website while it’s in development mode you need to be registered by the owner on the Spotify website, but the API does not distinguish between these users automatically to fix this there were more checks added to the websites login API catch non registered users and then send them to the login error page.

### Tab functions unresponsive in popups

During user testing a user pointed out that in some scenarios while using the website while using the tab function to navigate it the tab button would not interact with the popups this was fixed by adding a focus grabber to the modalHandler which meant when the popup was opened the focus would be forcefully moved to the popup and locked to only interact with it until the popup was closed.

## 4.6 Limitations

### Brighton domains:

There were plans to use a WebSocket made using Nodejs to handle requests between the user and Brighton domains. I decided to use this as it would improve the performance of the user’s device and put less strain on the servers. But as Nodejs is no longer present on Brighton domains I instead opted to use a polling system which has got some performance drawbacks it is a suitable replacement for the WebSocket.

### Time constraints:

To make sure that all the features planned in the progress tracker would be completed along with the website not all the features that were envisioned for the website were added this was done to make sure that the website would be finished to a high standard without any bugs and at a later date the features that did not make it into the prototypes could be added.

### Real world testing:

I had the idea to perform real world testing but due to me only being able to do testing with people from within the same module doing testing in the real world would be unethical and break multiple rules that are imposed on the assignment due to this I have decided not to do this kind of testing and instead more unit testing and Whitebox testing was used.

## 4.7 Final product

The final product of the website is built up of multiple parts including both server side and client-side code. The server-side codebase includes two automated PHP scripts. One of the scripts handles closing parties which have expired and the other handles refreshing the party’s authentication token for Spotify. The server-side also includes a database which is used to store all the necessary information about the party including verification information, settings and authentication tokens. A screenshot displaying the layout of the database can be seen in **Appendix 8**.

The client-side codebase consists of many parts including the HTML for the pages included on the website. These pages are also linked to a JavaScript file specific to the page which handles all the functionality of the page. All the pages are also linked to a CSS file which handles the styling for the pages, controlling how they look and making sure all page’s match when it comes to design. The client-side codebase also contains multiple JavaScript tool to perform certain actions which is shared across multiple pages. An example of one of these tools is the cookies.js which is used across multiple pages to handle and manage the user’s cookies.

Along with the codebase a logo was also designed for the website as a way for people to identify the website. This logo appears when you share the link to the website and when an invite to the party is shared a screenshot of the logo can be seen in **Appendix 7**.

The website is then comprised of six HTML webpages all serving a different purpose of the website these pages are the homepage, 404, loginerror, join, dashboard and party.

### Homepage:

The homepage is the first page that any user going to the website will see unless this have been invited to a party by another user. This page offers the ability to go to navigate to the join page, to login to Spotify account, to navigate to dashboard page if logged into their Spotify a screen shot of the page can be seen in **Appendix 10.**

### 404:

The 404 page is shown when accessing a page which does not exists in website a screenshot from this page can be seen in **Appendix 11** thispage has no functionality other than the ability to return to the homepage.

### Loginerror:

The loginerror page is used to tell the users about any issues that occur while they try to login into their Spotify account through the website an example of one of the errors can be found in **Appendix 12** the page can show 4 different error screens involving no premium account, in development mode, API request limit reached and a general error screen.

### Join:

The join page is used for guests with the party join code can enter the code and join the party the page also can navigate back to the homepage a screenshot from this page can be seen in **Appendix 13**.

### Dashboard:

The dashboard page is separated into 2 screen a create party screen which can be seen in **Appendix 14** and a party management screen which can be seen in **Appendix 15**. The create party screen offer the ability for the user to enable settings such as blocking duplicate songs and allowing explicit songs it also offer the ability for the user to control how many hours the party will stay active for.

The party management page offers the ability for hosts to adjust settings for their active party these included being able to either enable or disable settings, extend the parties duration and being able to end the party early before the duration they set has run out. The page also has information about how the website and party works and contains information about joining parties and displays a QR code which can be used for guests to join and other important information about joining and other functions which can be used to invite guests.

### Party:

The party page is what the invited guests use to add songs to the hosts song queue this page also offer the guest the ability to search for any song available on Spotify through the search function it will then return all the available results a screenshot from the page can be found in **Appendix 16**.

# Critical review

The house party website was created to fill a gap in the market of group music apps and to help bring people together while they are listening to music in groups as it allows all invited guests to add songs to the groups queue.

House party has met the aims and objectives which were set out for it at the beginning of the document one of the aims of for the website was for it to have an easy to use UI which house party was done very well with a very basic and clear to understand UI making it easy to use and accessible to everyone while still providing all of the functionality that was planned to be added to it. The basic design also helps it to meet the aim to be accessible as it makes it easy for tool like screen readers to interpret the website furthermore a lot of development time while working on the UI was spent making sure all aspects of the website would be accessible meaning that all parts of the website can be used by everyone.

House party was made to combat the need for users to have an account or app which house party does successfully offering a free and account less experience for guests unlike other service this means it is not restricted to just users who use one music platform allowing anyone to be invited to add songs to the music queue.

The final prototype of the website does not have all the features and tools that were planned for the website this was planned to allow for the main features and functionality of the website to be perfected and finalised so the final prototype would be of a high standard but the features that weren’t available in the protype could be added at a later time to build upon the final product.

# Conclusion

# References

*Spotify Web API (no date) Web API | Spotify for Developers. Available at:* [*https://developer.spotify.com/documentation/web-api*](https://developer.spotify.com/documentation/web-api) *(Accessed: 20 September 2024).*

*Spotify developer terms (no date) Spotify Developer Terms | Spotify for Developers. Available at:* [*https://developer.spotify.com/terms*](https://developer.spotify.com/terms) *(Accessed: 20 September 2024).*

*Start or join a Jam* (no date) *Spotify*. Available at: <https://support.spotify.com/uk/article/jam/> (*Accessed: 20 September 2024*).

*API reference  |  YouTube Data API  |  google for developers (no date) Google. Available at:* [*https://developers.google.com/youtube/v3/docs/*](https://developers.google.com/youtube/v3/docs/%20) *(Accessed: 10 December 2024).*

*Apple Music Api (no date) Apple Developer Documentation. Available at:* [*https://developer.apple.com/documentation/applemusicapi*](https://developer.apple.com/documentation/applemusicapi%20) *(Accessed: 10 December 2024).*

*SoundCloud for developers (no date) API - Guide - SoundCloud Developers. Available at:* [*https://developers.soundcloud.com/docs/api/guide*](https://developers.soundcloud.com/docs/api/guide) *(Accessed: 10 December 2024).*

*Moscow prioritization (2024) ProductPlan. Available at:* [*https://www.productplan.com/glossary/moscow-prioritization/*](https://www.productplan.com/glossary/moscow-prioritization/) *(Accessed: 11 December 2024).*

*Design & Branding Guidelines (no date) Design & Branding Guidelines | Spotify for Developers. Available at:* [*https://developer.spotify.com/documentation/design*](https://developer.spotify.com/documentation/design) *(Accessed: 13 February 2025).*

*Roboto (no date) Google Fonts. Available at:* [*https://fonts.google.com/specimen/Roboto*](https://fonts.google.com/specimen/Roboto) *(Accessed: 14 February 2025).*

*About projects (no date) GitHub Docs. Available at:* [*https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects*](https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects) *(Accessed: 15 February 2024).*

*Mark Otto, J.T. (no date) Bootstrap, Bootstrap · The most popular HTML, CSS, and JS library in the world. Available at:* [*https://getbootstrap.com/*](https://getbootstrap.com/) *(Accessed: 14 February 2025).*

*Font awesome (no date) Font Awesome. Available at:* [*https://fontawesome.com/*](https://fontawesome.com/) *(Accessed: 14 February 2025).*

[*davidshimjs*](https://github.com/davidshimjs) *(No date) Qrcode.js. Available at:* [*https://davidshimjs.github.io/qrcodejs/*](https://davidshimjs.github.io/qrcodejs/) *(Accessed: 14 February 2025).*

*Introduction to lighthouse  :  chrome for developers (2016) Chrome for Developers. Available at:* [*https://developer.chrome.com/docs/lighthouse/overview*](https://developer.chrome.com/docs/lighthouse/overview) *(Accessed: 15 February 2025).*

*Fast and reliable end-to-end testing for modern web apps (no date) Playwright. Available at:* [*https://playwright.dev/*](https://playwright.dev/) *(Accessed: 15 February 2025).*

# Appendices

## Appendix 1:

|  |  |
| --- | --- |
| Date | Meeting notes |
| 26/09/2024 | * Spoke about project idea * Talked about the legal side of the project * Spoke about what research needs to be done into the terms of service |
| 03/10/2024 | * Updated on legal research * Updated on project progress |
| 10/10/2024 | * Progress update on designing the project * Did ethical form * Updated on project progress |
| 24/10/2024 | * Updated on project progress * Spoke about submitting the Spotify API forms to have the project recognised by them |
| 13/11/2024 | * Spoke about the interim report focusing on what I have in It already and what I could add to it * Updated on project progress |
| 18/02/2025 | * Updated on development of the project * Shown a demonstration of a prototype working * Updated on the status of Spotify Application * Discussed features that could be added or discussed in report * Discussed the report of the dissertation |

## Appendix 2:

A computer screen shot of a program code

AI-generated content may be incorrect.

## Appendix 3:

A screenshot of a computer

AI-generated content may be incorrect.

## Appendix 4:

A screenshot of a computer program

AI-generated content may be incorrect.

## Appendix 5:

A screenshot of a computer

AI-generated content may be incorrect.

## Appendix 6:

A screenshot of a computer

AI-generated content may be incorrect.

## Appendix 7:

A green and black sign

AI-generated content may be incorrect.

## Appendix 8:

A black background with white text

AI-generated content may be incorrect.

## Appendix 9:

A screenshot of a computer

Description automatically generated

## Appendix 10:

A screenshot of a computer

AI-generated content may be incorrect.

## Appendix 11:

A screenshot of a computer error

AI-generated content may be incorrect.

## Appendix 12:

A screen shot of a computer error

AI-generated content may be incorrect.

## Appendix 13:

A black screen with green text and green buttons

AI-generated content may be incorrect.

## Appendix 14:

A screenshot of a computer

AI-generated content may be incorrect.

## Appendix 15:

A screenshot of a party settings

AI-generated content may be incorrect.

## Appendix 16:

A screenshot of a computer

AI-generated content may be incorrect.

# Bibliography:

*Spotify Web API (no date) Web API | Spotify for Developers. Available at:* [*https://developer.spotify.com/documentation/web-api*](https://developer.spotify.com/documentation/web-api) *(Accessed: 20 September 2024).*

*Spotify developer terms (no date) Spotify Developer Terms | Spotify for Developers. Available at:* [*https://developer.spotify.com/terms*](https://developer.spotify.com/terms) *(Accessed: 20 September 2024).*

*Start or join a Jam* (no date) *Spotify*. Available at: <https://support.spotify.com/uk/article/jam/> (*Accessed: 20 September 2024*).

*API reference  |  YouTube Data API  |  google for developers (no date) Google. Available at:* [*https://developers.google.com/youtube/v3/docs/*](https://developers.google.com/youtube/v3/docs/%20) *(Accessed: 10 December 2024).*

*Apple Music Api (no date) Apple Developer Documentation. Available at:* [*https://developer.apple.com/documentation/applemusicapi*](https://developer.apple.com/documentation/applemusicapi%20) *(Accessed: 10 December 2024).*

*SoundCloud for developers (no date) API - Guide - SoundCloud Developers. Available at:* [*https://developers.soundcloud.com/docs/api/guide*](https://developers.soundcloud.com/docs/api/guide) *(Accessed: 10 December 2024).*

*Moscow prioritization (2024) ProductPlan. Available at:* [*https://www.productplan.com/glossary/moscow-prioritization/*](https://www.productplan.com/glossary/moscow-prioritization/) *(Accessed: 11 December 2024).*

*Design & Branding Guidelines (no date) Design & Branding Guidelines | Spotify for Developers. Available at:* [*https://developer.spotify.com/documentation/design*](https://developer.spotify.com/documentation/design) *(Accessed: 13 February 2025).*

*(No date a) Every noise at once. Available at:* [*https://everynoise.com/*](https://everynoise.com/%20) *(Accessed: 13 February 2025).*

*(No date) Tuner. Available at:* [*https://tommygeiger.com/tuner/*](https://tommygeiger.com/tuner/) *(Accessed: 13 February 2025).*

*Roboto (no date) Google Fonts. Available at:* [*https://fonts.google.com/specimen/Roboto*](https://fonts.google.com/specimen/Roboto) *(Accessed: 14 February 2025).*

*About projects (no date) GitHub Docs. Available at:* [*https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects*](https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects) *(Accessed: 15 February 2024).*

*Mark Otto, J.T. (no date) Bootstrap, Bootstrap · The most popular HTML, CSS, and JS library in the world. Available at:* [*https://getbootstrap.com/*](https://getbootstrap.com/) *(Accessed: 14 February 2025).*

*Font awesome (no date) Font Awesome. Available at:* [*https://fontawesome.com/*](https://fontawesome.com/) *(Accessed: 14 February 2025).*

[*davidshimjs*](https://github.com/davidshimjs) *(No date) Qrcode.js. Available at:* [*https://davidshimjs.github.io/qrcodejs/*](https://davidshimjs.github.io/qrcodejs/) *(Accessed: 14 February 2025).*

*Introduction to lighthouse  :  chrome for developers (2016) Chrome for Developers. Available at:* [*https://developer.chrome.com/docs/lighthouse/overview*](https://developer.chrome.com/docs/lighthouse/overview) *(Accessed: 15 February 2025).*

*Fast and reliable end-to-end testing for modern web apps (no date) Playwright. Available at:* [*https://playwright.dev/*](https://playwright.dev/) *(Accessed: 15 February 2025).*