|  |
| --- |
| CI601 – House Party Report |

|  |
| --- |
| Alexander wood - 22827042 |

Contents

[Introduction 3](#_Toc194414283)

[Motivation: 3](#_Toc194414284)

[literature review: 3](#_Toc194414285)

[Spotify API terms of service: 3](#_Toc194414286)

[Spotify Web API documentation: 3](#_Toc194414287)

[Apple Music API documentation: 4](#_Toc194414288)

[Soundcloud API documentation: 4](#_Toc194414289)

[YouTube documentation: 4](#_Toc194414290)

[Spotify Design & Branding Guidelines*:* 4](#_Toc194414291)

[Research 4](#_Toc194414292)

[Tools and software being used 4](#_Toc194414293)

[Visual Studio Code (VS Code): 4](#_Toc194414294)

[GitHub: 5](#_Toc194414295)

[GitHub Projects: 5](#_Toc194414296)

[Brighton Domains: 5](#_Toc194414297)

[Planning 5](#_Toc194414298)

[Requirements: 5](#_Toc194414299)

[Design: 6](#_Toc194414300)

[Order of development: 6](#_Toc194414301)

[Development 6](#_Toc194414302)

[Testing 6](#_Toc194414303)

[White box testing: 6](#_Toc194414304)

[Black box testing 7](#_Toc194414305)

[Automatic testing: 7](#_Toc194414306)

[Limitations 7](#_Toc194414307)

[Brighton Domains: 7](#_Toc194414308)

[Real World Testing: 7](#_Toc194414309)

[Design Decisions 8](#_Toc194414310)

[Languages Used: 8](#_Toc194414311)

[Mobile First Design: 8](#_Toc194414312)

[Minified Files: 8](#_Toc194414313)

[Basic Design: 8](#_Toc194414314)

[Micro Services architecture: 8](#_Toc194414315)

[Spotify: 8](#_Toc194414316)

[Spotify guidelines: 9](#_Toc194414317)

[Server security: 9](#_Toc194414318)

[Risks 9](#_Toc194414319)

[Spotify API Dependent: 9](#_Toc194414320)

[Brighton Domain Servers: 9](#_Toc194414321)

[Documentation 9](#_Toc194414322)

[Conclusion 10](#_Toc194414323)

[References 10](#_Toc194414324)

[Appendix 10](#_Toc194414325)

[Appendix 1: 10](#_Toc194414326)

[Appendix 2: 11](#_Toc194414327)

[Appendix 3: 11](#_Toc194414328)

[Appendix 4: 12](#_Toc194414329)

# Introduction

In this project I aim to create a website that will allow someone to link their Spotify account to a party that they create and then invite people to the party allowing them to search and add songs to the Spotify accounts queue without the need for an app or a Spotify account.

# Motivation:

The aim is to create a website that enables someone to link their Spotify account to a party they create and then invite others to the party, allowing them to search and add songs to the Spotify account's queue without needing an app or a Spotify account. 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.

# literature review:

## Spotify API terms of service:

The Spotify API terms of service, I read through the terms of service to determine if my idea for the project would break any of the terms of service which would make the project illegal and unethical to make. The terms of service were very clear and made it very clear on what was possible to be made and what was not allowed to be done while using their API this was done by them breaking down their do and don’ts giving examples of what is not allowed for somethings to make it more clear and easier to understand.

## Spotify Web API documentation:

My next point of research was the API documentation to check to make sure that the project I wanted to do was going to possible. So, I started to by checking to see if the specific functions I needed for my project were present in their API and then followed this by reading through their API to find out how the Spotify login function works and how to handle the user’s authorisation tokens. I found certain parts of the documentation very useful, clear and easy to understand providing a sandbox playground which can be used to experiment with the API requests but there were certain parts of the documentation that required me to do more research outside of the documentation to find out how they work.

## Apple Music API documentation:

* Does not support the API functions required for the website
* Costs money to access to API to use it
* Not easy to read
* Can get lost on the website easily

## Soundcloud API documentation:

* Does not support the API functions required for the website

## YouTube documentation:

* Does not support the API functions required for the website

## Spotify Design & Branding Guidelines*:*

# Research

Before starting work on the project, the first step was research into the Spotify API and terms of service to make sure that the API offered the features required to build to the website and that the website would meet all the requirements of Spotify’s term of service.

I also followed this by doing research into other music streaming services APIs to see if it would be possible to add support for any other streaming platforms in the future, but the research proved that it would not be possible as they do not offer the required functions in their API’s preventing support from being added.

After all the technical research was complete research into the Spotify guidelines around using their branding was then done to make sure that their content would be displayed and presented properly.

# Tools and software being used

## 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 I decided to use GitHub projects I did this as it keeps both the code and the progress tracker together while also offering the same features as other project progress trackers.

A screenshot of a computer

AI-generated content may be incorrect.

## Brighton Domains:

To Host the system, I have made I decided to use Brighton domains I did this as I already have access to Brighton domains allowing me to straight away start hosting my system including a database, database events, automatic PHP files, and HTML which meets all the requirements I needed.

# Planning

## Requirements:

To start the planning phase, 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.

A screenshot of a computer

AI-generated content may be incorrect.

## Design:

The next step in the planning process was to start designing what the website and user interface would look like. This involved drawing small sketches of the website to design the layout, determine where things would be on the page, and envision how it was going to look. These sketches were then taken and made into wireframes of the final design for the website, these wireframes showed how the buttons would be laid out and a more in-depth look at what the final product will look like.

## Order of development:

To finish the planning phase, a new GitHub projects page was created to track development and plan when specific parts of the website should be made. This allowed for a clear starting point in development and ensured the progress stayed on track without losing focus.

# Development

# Testing

While developing the system I used 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 automatic.

## 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 furth 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.

## Automatic 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.

# 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, but it is a suitable replacement for the WebSocket as it can be used to perform the same action.

## 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.

# Design Decisions

## Languages Used:

When it came to the languages that I decided to use to build the system I decided to use JS, HTML, CSS and PHP I decided on these as Brighton domains supports hosting of these types of files, and I have knowledge in how to use these programming languages.

## Mobile First Design:

Due to the system mainly going to be used on mobile devices the decision was made to focus on a design which focuses on mobile devices mainly and does not have a separate view for larger screens instead it just scales to the screen size. By doing this it allowed for some time to be saved and used to focus on features that are more important.

## Minified Files:

To help improve performance of the website I decided to use minified JS and CSS files to help improve loading speeds and deliver a smoother user experience.

## Basic Design:

I decided to make a basic design for the website so it is easy for anyone to pick up and use this was also influenced by the environment it is going to be used in where people do not want complicated interfaces with fancy designs as it can make it harder to use.

## Micro Services architecture:

When building the system, I decided to build it using micro services this allows parts of the website to be reused in other parts as they as they work individually this also improves maintainability this is because the code is easier to find and also mean when updating the parts it updates across all parts of the website that use the function being updated.

## Spotify:

I decided to use Spotify over other music platforms for multiple reasons the first reason is that I use Spotify and have access to a Spotify account where I do not have access to an account from other platforms. The second reason is the API Spotify’s API is free to use and has the required API functionality I need to build the website where other platforms do not have the required functions or are not free to use and require a subscription. The final reason is that Spotify has a large user base meaning there will be more people able to use the website compared to if I used one of the not as popular music platforms.

## Spotify guidelines:

Some parts the of the websites design had to be changed from my original design this was due to the Spotify having guidelines around the design and interaction with the website which must be met for the website to pass the inspection done by Spotify before it can be taken out of development mode.

## Server security:

When it came to adding security to the server side of the website, I decided to make it so that no external connections can be made to either the database or API this was all done by restricting the connections to only allow ones that originate from the same server.

# Risks

## Spotify API Dependent:

A major risk that affects the system is its dependency on the Spotify API this means that if the Spotify API becomes unresponsive or unavailable the system will not be able to work. This also means that if anything about the API changes that affects functions that is used in the system the system will need to be updated to support the new changes leaving it unavailable till the update has been done.

## Brighton Domain Servers:

Since the system is hosted solely on Brighton domains without redundancy or backup measures, it will become unresponsive and inaccessible in the event of a failure of Brighton domains.

# Documentation

Throughout the development of the website there were multiple custom-built tools made so while they were being developed documentation was wrote about all the tools, functions and APIs made for the website this was done to make future development easier. This will make future development easier as it allows for people to come back to the code and understand how each part works without having to figure it out themselves.

# 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).*

*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).*

*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).*

# Appendix

## 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.