

Automating Spotify Playlist Creation using

Machine Learning

Interim Report

TU856

BSc in Computer Science

**Alexandros Tsiogas**

**C20336236**

**Brendan Tierney**

School of Computer Science

Technological University, Dublin

**31/10/2023**

Abstract

This project aims to create a new way for listeners to find music that they enjoy, using machine learning. This project explores the shortcomings of the music recommendation systems that are currently popular and aims to address such shortcomings through a new recommendation system. The project focuses on addressing perceived issues in the most widely used music streaming platform – Spotify.

This system will be presented through a web application that connects to a user’s Spotify account and accesses their data. The user is presented with a list of their Spotify playlists and may select one, from which a series of recommendations will be generated by a machine learning model. Prior to recommendation generation, a user is given the option of pre-emptively changing their recommendations, by increasing or decreasing variables about the mood and sound of the music. This aims to address the apparent issue of a user’s mood not being considered when they are being recommended music by Spotify’s current system.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

Alex Tsiogas\_\_\_\_

Alexandros Tsiogas

31/10/2023

Acknowledgements

Body text

Table of Contents

[1. Introduction 8](#_Toc150526173)

[1.1. Project Background 8](#_Toc150526174)

[1.2. Project Description 8](#_Toc150526175)

[1.3. Project Aims and Objectives 9](#_Toc150526176)

[1.4. Project Scope 9](#_Toc150526177)

[1.5. Thesis Roadmap 10](#_Toc150526178)

[2. Literature Review 11](#_Toc150526179)

[2.1. Introduction 11](#_Toc150526180)

[2.2. Alternative Existing Solutions 11](#_Toc150526181)

[2.3. Technologies 14](#_Toc150526182)

[2.3.1 Programming Languages 14](#_Toc150526183)

[2.3.2 Web Frameworks 15](#_Toc150526184)

[2.4. Recommender Systems 16](#_Toc150526185)

[2.5. Existing Final Year Projects 17](#_Toc150526186)

[2.5.1 Project 1: Moodify 17](#_Toc150526187)

[2.5.2 Project 2: Football Data Mining, Result Prediction, and Visualization 18](#_Toc150526188)

[2.6. Conclusions 20](#_Toc150526189)

[3. System Design 21](#_Toc150526190)

[3.1. Introduction 21](#_Toc150526191)

[3.2. Software Methodology 21](#_Toc150526192)

[3.3. Software Methodology 21](#_Toc150526193)

[3.4. Overview of System 21](#_Toc150526194)

[3.X. Other Sections 21](#_Toc150526195)

[3.X. Conclusions 21](#_Toc150526196)

[4. Testing and Evaluation 22](#_Toc150526197)

[4.1. Introduction 22](#_Toc150526198)

[4.2. Plan for Testing 22](#_Toc150526199)

[4.3. Plan for Evaluation 22](#_Toc150526200)

[4.4. Conclusions 22](#_Toc150526201)

[5. Prototype Development 23](#_Toc150526202)

[5.1. Introduction 23](#_Toc150526203)

[5.2. Prototype Development 23](#_Toc150526204)

[5.3. Other Sections 23](#_Toc150526205)

[5.4. Conclusions 23](#_Toc150526206)

[6. Issues and Future Work 24](#_Toc150526207)

[6.1. Introduction 24](#_Toc150526208)

[6.2. Issues and Risks 24](#_Toc150526209)

[6.3. Plans and Future Work 24](#_Toc150526210)

[6.3.1. GANTT Chart 24](#_Toc150526211)

[Bibliography 25](#_Toc150526212)

# 1. Introduction

## Project Background

In a time where online music streaming services are more popular than ever, Spotify dominates the market. Stats gathered in 2022 show that Spotify have by far the greatest market share in this domain at 30.5% - over twice that of the nearest competitor, Apple Music (1).

Despite Spotify’s massive usage, listeners seem to be engaging less with their music recommendation functionality. Spotify offers several ways for users to explore new music, such as their “Radio” or “Daily Mix”, and yet surveys have found that more listeners are finding new music through traditional means. In a 2019 survey of over 500 music listeners, 45.9% of responses listed “Active search myself” as their main source of new music, while Spotify and other streaming services accounted for only 26.1% of responses (2).

A pie chart with text

Description automatically generated

Figure : Survey results on how people find new music (2).

These statistics raise questions over why Spotify’s dominance in the music streaming domain cannot be replicated in the music recommendation domain.

The shortcomings of Spotify’s recommendation systems are a matter of clear frustration for users. A 2015 study (3) found that only 3% of Spotify users find that Spotify generated recommendations always match their taste, while 68.75% of users stated they match their taste only sometimes. The fact that users’ moods aren’t considered when they are being recommended music was highlighted as “one of the most important drawbacks” of the recommendation system. 66.7% of users interviewed chose “mood” as the main influence factor on the music they want to listen to.

It is abundantly clear that Spotify is failing to satisfy its user’s desires to find new music that matches their mood, leading to many users searching elsewhere for new music. This project will attempt to address these shortcomings by developing a recommendation system that can better account for the mood of a user.

## Project Description

This project will take the form of an online web application. The app will be presented through web pages designed with HTML and CSS, and the underlying functionality will be made in the Python web framework “Flask”.

Upon entering the web app, users will be presented with a Spotify login screen that prompts them to log in with their username and password. Once they successfully log in, the application is granted access to their saved playlists. This is achieved through the Spotify Web API, which enables the creation of applications that can interact with Spotify's streaming service. The user will then be presented with a list of their saved playlists, from which they can select a playlist as a basis for recommendation creation. The user may then alter some variables about the mood of the music they desire before recommendations are generated. The contents of the playlist and the mood values can then be passed through a machine learning model, which eventually outputs a series of recommendations. These recommendations can be saved directly to a user’s library through the user interface, via the Spotify Web API.

Include a diagram.

## Project Aims and Objectives

The main objective of this project is to produce a web application and machine learning model that are capable of producing music recommendations that account for user mood. In success, these recommendations will be superior to those produced by Spotify – this sentiment will be measured in the end evaluation. The milestones that will be reached throughout the project are as follows:

* Extensively analyse the strengths and shortcomings of the current Spotify recommendation algorithm and determine how my application can improve on it.
* Successfully build a Flask web app that grants access to a user’s Spotify library.
* Implement an aesthetically pleasing front end, with a visually pleasing UI that is easy to navigate and offers a good user experience.
* Create functionality through the Spotify Web API that allows the manipulation of a user’s library via the web application.
* Creation of elements on the web page that allow a user to change variables about the mood of their recommendations, which will be processed by the machine learning algorithm.
* Creation of a machine learning algorithm that can select songs from Spotify’s database to recommend to a user based on their selected playlist and their mood values.

## Project Scope

This project is at heart a machine learning project, focused on designing a machine learning model that can accept direct user input and existing user Spotify data, and producing recommendations based on these. The retrieval, cleaning and processing of user data are all within the scope, as is the start-to-finish construction of a suitable recommender system, including all testing and evaluation.

Furthermore, a user-friendly web application that serves as user interface for interaction with the recommender system will be developed, tested, and evaluated.

This project aims to be compatible with Spotify only, due to the popularity of the service and the availability of the Spotify Web API. Compatibility with any other online music streaming service is considered out of scope.

The application aims to be presented in a web application – no mobile application is in scope.

While the machine learning algorithm will account for user mood, the application will play no role in determining the mood of the user - the user will describe their own mood through on-screen elements.

## Thesis Roadmap

**Chapter 2 – Literature Review**

This chapter will describe the research conducted and literature reviewed thus far in the project.

**Chapter 3 – System Design**

This chapter will describe the planned design of the final system, including diagrams describing architecture and visuals. It will also introduce the methodologies to be used in development.

**Chapter 4 – Testing and Evaluation**

This chapter will describe a plan for the eventual testing and evaluation of the finished project.

**Chapter 5 – Prototype Development**

This chapter will present the prototype that has been created and will detail all work completed and milestones achieved thus far in the development process.

**Chapter 6 – Issues and Future Work**

This chapter will outline any issues faced in the development process so far and will discuss potential ways of resolving these issues going forward. It will also identify a plan of future work needed to complete the project, which will be structured as a GANTT chart.

# Literature Review

## 2.1. Introduction

This chapter describes all research conducted and literature reviewed thus far in the project.

## 2.2. Alternative Existing Solutions

The shortcomings of the Spotify recommendation system are an issue that developers have been trying to address a great deal in recent times, as is evident in the sheer volume of web and mobile applications that have been developed that utilise the Spotify API to build on the application’s core functionality.

**Discz**

One such application is the mobile app “Discz” - a massively successful music recommendation app that operates using the Spotify API. The app allows users to swipe through songs and give their sentiments on them, in a binary “like” or “dislike”. The app uses the users swiping data to learn about the user’s taste, and gradually improve its recommendations.

A screenshot of a phone

Description automatically generated

Figure 2: Discz' swiping functionality

The Discz app uses a machine learning algorithm in its recommendations, which proves the feasibility of my project, and means the app can serve as a comparison to my finished product, and a source of research and comparison for my initial design. The popularity of ‘Discz’ is highlighted in a 2022 Rolling Stone Article, which finds that “more than 15 million songs have been discovered and saved on the app to date” (4). This figure clearly shows the desire that music listeners have for better ways to find new music.

The Discz app serves as an inspiration and a comparison for some facets of my app. One such way is that it eliminates the need for the user to manually add songs to a playlist. The flow of the recommendation system is easy to follow - a user repeatedly swipes left or right, and once they are satisfied with what they have selected, all the songs can be added directly to a newly created playlist. This simple recommendation flow is something I found to be very intuitive, and it aids in streamlining the playlist creation process. It is something I have drawn inspiration from in the design of my own app, where a user will be able to generate recommendations and export them to their Spotify library with the click of a button.

There are some areas where Discz succumbs to the same shortcomings of the Spotify recommendation system – it does not directly account for user mood. The recommendations are not mood based, and the user gets no direct say over what they are recommended. As previously discussed, users want to have their mood considered when they are being recommended music, hence why mood-based recommendation is a primary goal of my project.

**MagicPlaylist**

One application that does make some attempt to account for user mood is MagicPlaylist. Much like my application, this is a web app that offers music recommendations, using the Spotify Web API. The homepage presents the user with a number of “moods”, upon which they can have a playlist generated.

A collage of two people

Description automatically generated

Figure : The "moods" feature of MagicPlaylist

A screenshot of a computer

Description automatically generated

Figure : Recommendations generated by MagicPlaylist.

The strongest merits of this app lie in its UI and UX. The UI is exceptionally intuitive throughout, with every “mood” framed in a large clickable button with a relevant background image.

A screenshot of a computer

Description automatically generated

Figure : The MagicPlaylist UI

The landing page is tidy and concise, comprising of just a search bar and the “mood” buttons. The capabilities of the website are described elegantly in the header ­- “*Type your favourite songs and create perfect playlists. Magic!”* - and a user can easily intuit the how to generate a playlist from here, making user experience straightforward. Comparing it to similar applications in the music recommendation space, MagicPlaylist is one of the most accessible for users. In my research, it served as an inspiration for the design of my UI, and the general flow of user interaction with the website.

The application does however lack somewhat in the mood-based recommendations. The user plays no role in defining the moods - they are predetermined within the application. This detracts from the user’s control over what they are recommended, as none of the predetermined moods may match exactly what they desire. I aim to grant the user more control over describing their mood, by allowing them to fine tune multiple “mood variables”, that their suggestions will be based upon.

Due to it being so close in nature to my application, MagicPlaylist may well serve as a comparison in my final evaluation.

## 2.3. Technologies

This section describes technologies that were researched and considered for use in the project, including those that were ultimately chosen to be used.

### 2.3.1 Programming Languages

This section describes the programming languages researched for the application.

##### 2.3.1.1 Java

Java is a general-purpose class-based, object-oriented programming language, designed to have as few implementation dependencies as possible (5). As of 2019, it was reported to be one of the most used programming languages among developers, particularly in developing client-server web applications, with a reported 9 million developers (6). Its widespread popularity in web applications and the large amount of documentation available make java a viable option.

A white background with black text

Description automatically generated

Figure : Some simple Java code (7).

##### 2.3.1.2 Python

Like Java, Python is a widely used general- purpose programming language, used in a range of applications including web applications (8). Python has a greater focus on readability and is less verbose than Java *(see Figure 6).* Python was chosen as the main programming language for this project due to its readability and compatibility with numerous packages, including Spotipy. Spotipy and its importance to my project will be described in the next section.

A white background with black text

Description automatically generated

Figure : Some simple Python code (7).

###### 2.3.1.2.1 Spotipy

Spotipy is a lightweight Python library for the Spotify Web API. It essentially eliminates the need to make direct calls to the Spotify API, instead compressing these calls into built in python functions that can be used to access Spotify data (9). Using Spotipy, data can easily be retrieved from a user’s Spotify library in the form of python arrays and dictionaries. This allows for easy manipulation of the data by other parts of the application that are written in Python, namely the machine learning model. By writing the application in Python and using the Spotipy library, I avoid any potential language interoperability problems (10). Research was conducted on Spotipy through the official Spotipy docs (9) and through past final year projects that utilised it (see section 2.5.1).

### 2.3.2 Web Frameworks

This section describes viable web frameworks researched for the application.

##### 2.3.2.1 Django

Django is a popular python-based web framework. The Django website describes it as being a framework that “encourages rapid development and clean, pragmatic design” Django is generally preferable for large, full-stack projects that require incorporated complex features (11).

##### 2.3.2.2 Flask

Flask is a more lightweight web framework written in Python. It is one of the most widely used web frameworks and has become commonly accepted as the perfect tool for quick and simple solutions. It is generally preferred for smaller and more lightweight web applications (12).

Flask’s more lightweight nature and its popularity with smaller scale applications make it the most viable option for my application. As most of my project’s complexity will fall under the machine learning aspect, the web application will be relatively small-scale. Flask was chosen as the web framework to be used in the app.

## 2.4. Recommender Systems

This section details the research conducted into machine learning and viable models for my application. The two main approaches to recommender systems are **Collaborative** **Filtering** and **Content-Based** **Filtering**, which were the two methods researched.

##### 2.4.1 Collaborative Filtering

Collaborative filtering is a common approach to designing recommender systems. It essentially anticipates the taste of a user by comparing known data about them with that of other users, often by utilising the k nearest neighbours’ algorithm (13).

A table of names and numbers

Description automatically generated with medium confidence

Figure : A table of data to about peoples’ ratings of movies (13)

For example, in figure 7, a collaborative based filtering algorithm should predict a favourable score for Carol for Harry Potter, as her scores closely mirror those of Joe.

Collaborative filtering is only possible if sentiment data is available for a large group of users, which will not be the case for my app, which will deal just with the data of a single user. For this reason, collaborative filtering was ruled out upon research.

##### 2.4.2 Content-Based Filtering

Content-based filtering operates similarly to collaborative filtering, but instead of recommending based on the sentiments of other users, it uses a user’s own data from the past. At a high level, the system essentially gets a feeling for the user’s taste using available data and makes recommendations that appeal to these tastes. When implemented, a content-based filtering system could recommend a user items that are similar to items they have expressed positive opinions about in the past (13).

A screen shot of a white background

Description automatically generated

Figure : A user's list of favourite movies.

Given a user’s list of favourite movies like that in figure 8, a Content Based Filtering System could recommend movies the user may like, from a larger list of movies. This can be manipulated in such a way that it could recommend a user songs that have similar audio features to songs already in their playlist.

Upon research, the content-based filtering approach was chosen as the approach for my application. Details about the planned implementation were described in section (SECTION HERE)

## 2.5. Existing Final Year Projects

As part of my initial research, I reviewed Final Year Projects from years past, that pertain to my project. Two of those projects will be discussed here, as they have served as a source of inspiration for my own project.

### 2.5.1 Project 1: Moodify

**Project Title:** Moodify

**Student:** Louis Miguel Chavez

**Overview:**

This project is an Android application that classifies songs by mood, based on their lyrics and audio features, using AI and the Spotify API

The sentiment and music mood classifiers were made through Python, in Jupyter Notebook – other components were made in PyCharm. The Android application was made in Android Studio, using Java for functionality and XML for the layouts and visual components. A DigitalOcean Managed Cluster was used as the final database functionality.

The main strengths of this project, in my view, are the originality and accuracy of the mood classifier - it offers interesting insight into the listening habits of users. The functionality allowing for the classification of a song’s mood through its lyrics is equally original. I also find the front-end to be aesthetically pleasing and accessible for users of all technical skill levels. The only evident weakness of this project is that it requires users to manually upload playlist files for analysis, as opposed to having the client automatically extract them from the device – this is something I aim to avoid in my project.

**Similarities to my Project and Other Findings:**

This project and mine share appear to share clear similarities in the fact that they deal with the concept of ‘mood’ in music – however, they do so in very different ways. The main premise of this project is classifying the mood of a playlist using machine learning, while my project is more concerned with recommendation systems, and plays no role in determining user mood.

Where the projects are similar is in their use of machine learning models to classify music. This project contains a complex machine learning algorithm for sentiment classification, used to determine the mood of songs through sound and lyrics. While my project will not be used to determine the mood of a song, there is much I can learn from how the creator of this project constructed his machine learning model. A key piece of technology used by this student was **Spotipy**. As discussed in section 2.3, Spotipy is a Python library that can be used to interact with the Spotify Web API. I discovered Spotipy through this final year project and after conducting more research on it, decided to use it in my own.

### 2.5.2 Project 2: Football Data Mining, Result Prediction, and Visualization

**Project Title:** Football Data Mining, Result Prediction, and Visualization

**Student:** Yahia Ragab

**Overview:**

This project is a mobile and web application that allows users to view in depth stats on the English Premier League, and to view predictions of the outcome of matches, created with machine learning.

A huge amount of historical data relating to the English Premier League was mined and stored. A sophisticated machine learning model was developed to allow for predictions to be made, and a detailed front-end was designed that visualises statistics in digestible ways.

Python was used for creating the machine learning model, with use of libraries like NumPy, Pandas and Matplotlib for data visualisation. The web app was presented as a Django application, and Docker was used for deploying cloud components. MYSQL was used as the database management system.

This project contains a machine learning algorithm written in Python, similar in nature to the one I plan on creating in my app. The model is clearly highly sophisticated, and after rigorous testing seemed to yield very accurate results, which I believe is a huge strength of the project. The amount of data made available for viewing on the app is similarly impressive, and the way in which it is visualised is understandable and useful.

The front-end design is quite cluttered and perhaps not very user friendly and lacks some aesthetic appeal – it could have done a more thorough design phase that focused more on user experience. The project is, however, of a very high standard and displays many strengths which I could draw inspiration from.

**Similarities to my Project and Other Findings:**

This project differs from mine in subject matter, as it deals with football scores and results prediction. However, it utilises a machine learning model constructed in Python – something I plan on creating for my project too. What I gained most from researching this project was insight into how to evaluate machine learning models. This project was evaluated through a survey with potential users via a Usability Feedback Testing Form. In this form, users were asked to assess the accuracy of the results they received from the model, as well as their general sentiments about how useful it is. Information of this nature would be very useful to me in evaluating my final product, hence I took inspiration from this project when planning my evaluation – this will be discussed in greater detail in section 4.3.

## 2.6. Conclusions

After performing thorough research on existing literature in the domain of my project, I am prepared to enter the design and early development stages of my project. Many key decisions have been informed by information gathered through reviewing.

### 2.6.1 Requirements

While I always planned on making a Spotify music recommendation system, research on the biggest shortcomings of the Spotify recommendation algorithm influenced my decision to shift towards a mood-based recommendation system. Research on existing solutions to this issue revealed that while it’s an issue that’s been tackled often by developers, a perfect solution doesn’t exist and there is plenty of room for improvement in the domain.

### 2.6.2 Technologies

The decisions made on technologies to be used were heavily informed by comparative research conducted on viable options.

Python was chosen as a primary programming language after research, which influenced the choice of Flask as a web framework.

Researching past final year projects also aided in understanding how these technologies can be integrated into a larger system.

Content-based filtering was chosen as the approach to recommender system over collaborative filtering. It was seen as the most suitable option, as only user data will be available to the application, and not data of multiple users.

# 3. System Design

## 3.1. Introduction

This chapter describes the overall design of the project, including the visual style and layout of the frontend, and the structuring of the backend. It will also touch on the software methodologies adhered to during the development process.

## 3.2. Software Methodology

This project has combined multiple methodologies, using valuable and suitable elements of various common methodologies to establish an efficient and productive way of working.

The project has been performed iteratively, using the concept of “Sprints” commonly associated with the Agile and Scrum software development methodologies. The project adopted further principles commonly associated with Agile to enhance the flexibility and efficiency of work. Features of the Kanban methodology were also used, which focuses on visualising work and maximizing project efficiency. Each of these methodologies and the way in which they were used will be explained in this section.

## 3.2.1 Agile

Agile is a widely used software development methodology that defines rule that should be adhered to in the software development lifecycle (Figure 10). When adhered to, these rules make for more flexible development, focussed on continuous improvement and delivering working software that satisfies user needs (14)(15).

A diagram of a software company

Description automatically generated with medium confidence

Figure : The Agile Principles (15).

Some of the main principles of Agile, as described in the Agile Manifesto (15), are that changing requirements are welcomed, and working software is the primary measure of progress. These requirements are usually driven by the Product Owner – a member of an Agile team who puts forward the project objectives and represents the desires of the shareholders (16).

To adhere to the Agile framework, I undertook the role of Product Owner for this project. I did this by defining the requirements and deliverables of the project at an early stage. Changing requirements were welcomed, as they occurred due to development issues or external feedback from my supervisor. Keeping these defined requirements as the primary focus in development made for more efficient development. It also yields working software that can be evaluated throughout the project lifecycle, not only the interim stage.

Adherence to the Agile principles was further upheld by use of the Scrum methodology.

## 3.2.2 Scrum

Scrum is a software development framework commonly used by teams employing the Agile methodology. The main premise of Scrum is to develop software iteratively, to optimize predictability and control risk (17). This iterative development process is upheld through “Sprints” – these are 2/3 week long cycles of development, preceded by thorough planning of the work, and concluded with a retrospective analysis of work done and issues faced (17).

A diagram of a scrum

Description automatically generated

Figure : The Scrum Process (17)

Figure 11 illustrates the Scrum process of taking in requirements, ordering them by priority, completing a 2–3 week sprint of work, and reviewing the sprint. Scrum teams typically comprise of several members, such as developers, a Scrum Master, and a Product Owner.

For this project, I essentially assumed all these roles, and worked in one-week sprints. At the beginning of each week, I defined the tasks to be completed for the week ahead. At the end of each week, I performed a short retrospective, reflecting on the tasks completed and those that weren’t completed, identifying any issues or risks. Working in this way allows for greater predictability and makes risks easier to identify and mitigate.

## 3.2.3 Kanban

Like scrum, kanban is a development framework used to implement agile, but with a large emphasis on dividing up and visualising work into work items. These work items are visualised on a kanban board. Every work item involved in the project is represented on this board, usually in a “To do”, “Doing” or “Done” state. This allows all team members to understand the state of all work at any given time, and gives a clear idea of where the project is in its lifecycle and how much work remains (18).

A kanban board made in Azure DevOps was used in this project, to keep track of work easily and to better implement an agile style of work.

A screenshot of a computer

Description automatically generated

Figure : Project Kanban Board.

While scrum and kanban are often seen as two distinct options for development frameworks, it is possible, and often beneficial, to borrow parts of both in the development process (19). Kanban boards can make the scrum processes of planning work and reflecting on work done periodically easier, as they offer a visual description of work done and work remaining. This project made use of both kanban boards and scrum ceremonies & sprints.

## 3.3. System Architecture

This diagram illustrates a high-level architecture of the whole system, detailing how the server (flask application) interacts with the client and the requests made between them. It also describes the requests made by the app to the Spotify Web API. Note that for the prototype build, a large Spotify dataset from which recommendations are derived is stored locally in the server device’s file system – this is going to be exported to a database in the final application. This is discussed further in section (section here).

A diagram of a software company

Description automatically generated

Figure : Software Architecture Diagram

## 3.4. Frontend

## 3.4.1 Use Case Diagram

This Use-Case Diagram illustrates the functionality of the system and how the user can interact with this functionality through the front-end.

A diagram of a software application

Description automatically generated

Figure : Application Use Case Diagram

Once they access the web app, users are prompted to log in via a Spotify authentication page. They can then select a playlist from a list of their playlists and have a recommendation playlist generated. After this they can then choose to save this playlist to their library or have it regenerated with new recommendations. The general flow of this functionality is described in a flowchart in section 3.4.2.

## 3.4.2 Flowchart

This flowchart describes the flow of all of a user’s possible interactions with the web app.A diagram of a software process

Description automatically generated

## 3.4.3 Wireframes

The following wireframes illustrate the initial design of the application screens. Each wireframe was designed using Nielsen’s Heuristics as a reference point and guide. These are 10 principles used for interface design, commonly adopted by developers and web designers. A particular focus was placed on keeping the design aesthetic and minimalist, per Nielsen’s 8th Heuristic (20).

A screenshot of a music player

Description automatically generated

Figure : Application Landing Page

Figure 11 shows the initial landing page of the web app, displayed after a user has authenticated their Spotify account From this page, a user has the option to access the homepage (the current page), an “about” page, or to log out of their Spotify account. The large button in the centre of the screen labelled “Choose a Playlist” will direct a user to the playlist selection screen.

A screenshot of a music player

Description automatically generated

Figure : "Select a Playlist" Top Half of page.

Figure 12 is the top half of the “Select a Playlist” screen. It displays each of the user’s playlists in a scrollable list, retrieved from the Spotify API. The user can select one playlist as the recommendation subject, via a radio button.

A screenshot of a computer

Description automatically generated

Figure : "Select a Playlist" Bottom Half of page.

Figure 13 is the bottom half of the “Select a Playlist” screen. Here the user may alter the mood variables via sliders which are encapsulated in an area labelled “Change the Mood”. This area also contains a button that generates the recommendations and redirects the user to the recommendations page.

A screenshot of a computer

Description automatically generated

Figure : "Your Recommendations" Screen.

Figure 14 illustrates the “Your Recommendations” screen. The generated playlist is displayed here as a scrollable list of songs. A user can choose to regenerate these recommendations, recommendations will refresh to a new set. Otherwise, they can save the recommendations directly to their Spotify library as a playlist.

## 3.5. Backend

# 4. Testing and Evaluation

## 4.1. Introduction

## 4.2. Plan for Testing

## 4.3. Plan for Evaluation

## 4.4. Conclusions

# 5. Prototype Development

**As least 2 pages, but as many as you like (but lots of code samples).**

## 5.1. Introduction

## 5.2. Prototype Development

## 5.3. Other Sections

## 5.4. Conclusions

# 6. Issues and Future Work

## 6.1. Introduction

## 6.2. Issues and Risks

## 6.3. Plans and Future Work

### 6.3.1. GANTT Chart

# Bibliography

1. Statista [Internet]. [cited 2023 Oct 31]. Music streaming services subscribers market shares 2022. Available from: https://www.statista.com/statistics/653926/music-streaming-service-subscriber-share/

2. Dachs P. Hear what I’ve found — A survey on music discovery [Internet]. The Sound of AI. 2019 [cited 2023 Oct 31]. Available from: https://medium.com/the-sound-of-ai/hear-what-ive-found-a-survey-on-music-discovery-3869b6b6cbe4

3. Ding Y, Liu C. Exploring drawbacks in music recommender systems : the Spotify case [Internet]. 2015 [cited 2023 Oct 31]. Available from: https://urn.kb.se/resolve?urn=urn:nbn:se:hb:diva-8690

4. Leight E. It Started as a College Project. Now It’s the Music Industry’s Hottest Recommendation Tool [Internet]. Rolling Stone. 2022 [cited 2023 Nov 1]. Available from: https://www.rollingstone.com/music/music-features/discz-tiktok-recommendation-app-1324785/

5. Guttag BL with J. Program Development in Java - Abstraction, Specification, and Object-Oriented Design. USA, Addison Wesley; 2000.

6. Business Insider [Internet]. [cited 2023 Nov 7]. The 10 most popular programming languages, according to the “Facebook for programmers.” Available from: https://www.businessinsider.in/the-10-most-popular-programming-languages-according-to-the-facebook-for-programmers/articleshow/66253927.cms

7. Java Vs Python - Key Differences Between Java And Python [Internet]. [cited 2023 Nov 7]. Available from: https://www.softwaretestinghelp.com/java-vs-python/

8. Python (programming language) ~ Information Technology ~ 2420 ~ kelas-karyawan-bali.kurikulum.org [Internet]. [cited 2023 Nov 7]. Available from: http://kelas-karyawan-bali.kurikulum.org/IT/en/2420-2301/Python\_3721\_kelas-karyawan-bali-kurikulumngetesumum.html

9. Welcome to Spotipy! — spotipy 2.0 documentation [Internet]. [cited 2023 Nov 7]. Available from: https://spotipy.readthedocs.io/en/2.22.1/

10. Chisnall D. The Challenge of Cross-language Interoperability: Interfacing between languages is increasingly important. Queue. 2013 Oct 8;11(10):20–8.

11. Django Project [Internet]. [cited 2023 Nov 7]. Django. Available from: https://www.djangoproject.com/

12. Copperwaite M, Leifer C. Learning Flask Framework. Packt Publishing Ltd; 2015. 250 p.

13. B.Thorat P, M. Goudar R, Barve S. Survey on Collaborative Filtering, Content-based Filtering and Hybrid Recommendation System. Int J Comput Appl. 2015 Jan 16;110(4):31–6.

14. Atlassian. Atlassian. [cited 2023 Nov 11]. What is Agile? Available from: https://www.atlassian.com/agile

15. agile-manifesto-download-2019.pdf [Internet]. [cited 2023 Nov 11]. Available from: https://www.agilealliance.org/wp-content/uploads/2019/09/agile-manifesto-download-2019.pdf

16. Sverrisdottir HS, Ingason HT, Jonasson HI. The Role of the Product Owner in Scrum-comparison between Theory and Practices. Procedia - Soc Behav Sci. 2014 Mar;119:257–67.

17. Sachdeva S. Scrum Methodology. Int J Eng Comput Sci [Internet]. 2016 Jun 6 [cited 2023 Nov 11]; Available from: http://ijecs.in/issue/v5-i6/11%20ijecs.pdf

18. Atlassian. Atlassian. [cited 2023 Nov 11]. Kanban - A brief introduction. Available from: https://www.atlassian.com/agile/kanban

19. Kniberg H, Skarin M. Kanban and Scrum - Making the Most of Both. Lulu.com; 2010. 122 p.

20. Gonzalez-Holland E, Whitmer D, Moralez L, Mouloua M. Examination of the Use of Nielsen’s 10 Usability Heuristics & Outlooks for the Future. Proc Hum Factors Ergon Soc Annu Meet. 2017 Sep 1;61(1):1472–5.