# Computer Science NEA

## Analysis

### Intentions of my Project

My project aims to provide users with an intuitive platform to find out new information about their favourite music artists, as well as allowing the users to access an alternative to their preferred music streaming platform to get new music recommendations based on a song, artist or album they input.

### Problem Identification

Research has demonstrated that music offers several psychological and physiological advantages. It has been utilised as a therapeutic treatment to improve cognitive function, lessen stress and anxiety, and increase physical performance. A key part of appreciating music is being able to listen to new music, which not only provides novelty, but also helps develop and expand musical taste. Lack of access to new music can have a significant impact on someone in many ways, including how it affects their ability to find new music, their mood, their immune system, and their general well-being.

The most common way young people today find new music is based on Spotify recommendations of similar artists. The second most common way is recommendations from friends, however many of these recommendations have also come from Spotify’s recommendations. In summary, Spotify is the most significant way most young people are exposed to new music.

Spotify's current recommendation algorithm falls short in introducing consumers to fresh and varied music. The lack of variety and less satisfying listening experience are caused by the fact that many users frequently receive recommendations for the same albums and artists that they have already listened to. This issue is significant because it restricts users' capacity to discover new music, denies them the chance to broaden their musical preferences, and eventually causes user annoyance and a drop in platform happiness.

### Reasons for Change

The Spotify algorithm creates recommendations for each individual user based on information about the songs, albums, and artists that users have previously listened to as well as information about the listening preferences and habits of similar users. According to the algorithm, if a person has previously listened to an artist or album and other users with similar listening habits and tastes have done the same, the user is likely to appreciate it and want to hear it again. If a lot of listeners of a particular song, Song A, also listen to Song B, then anyone who listens to Song A or B will be recommended Song B or A – they become linked. This approach is known as collaborative filtering.

Collaborative filtering is a reasonable approach, but it suffers from the problem of ‘echo chambers’. As soon as Spotify starts to recommend Song B to people who like Song A, then lots of people who like Song A will start to listen to Song B, reinforcing the link within Spotify’s algorithm. It is a positive feedback system. Soon the links will end up focussing on a small number of very strongly linked songs, and it will be very hard for a new song to be recommended as it will always be way down the list of recommendations.

The current Spotify recommendation algorithm, which frequently repeats the same choices and prevents users from discovering new artists and genres, falls short of offering customers a wide variety of music recommendations. This not only makes it more difficult for users to find new music, but also makes listening less enjoyable overall, which annoys users and lowers their satisfaction with the site.

In conclusion, Spotify's subpar recommendation system is a serious issue that restricts the accessibility of new music and makes it difficult for users to diversify their musical tastes. This ultimately causes user annoyance and a drop in platform satisfaction.

Artists are one group that is most impacted by the issue of Spotify's poor music recommendations. Users are less likely to find and listen to new artists when they are not being recommended a wide variety of music. This reduces the potential audience for those artists and makes it harder for them to become known and develop a following. This is a concern because it makes it difficult for up-and-coming musicians to develop popularity and establish themselves in the music business, which ultimately reduces the variety and range of music that is accessible to platform users.

The fact that the Spotify algorithm is built on collaborative filtering may be one explanation for why it regularly suggests the same music and songs that the user has already listened to. Many recommendation systems, like Spotify, use the collaborative filtering technique to create recommendations based on the tastes and actions of similar users.

## In the case of Spotify, the algorithm creates recommendations for each individual user based on information about the songs, albums, and artists that users have previously listened to as well as information about the listening preferences and habits of similar users. According to the algorithm, if a person has previously listened to an artist or album and other users with similar listening habits and tastes have done the same, the user is likely to appreciate it and want to hear it again.

## Data Showing Proof Of Problem

To prove that other users of Spotify share the same issue as me, I created a questionnaire and asked my peers to answer five questions.

1. How much do people listen to music

Chart, pie chart

Description automatically generated

The results here are quite clear, showing an overwhelming majority of people saying that they listen to a lot of music. Most people considering music to be a large part of their life would make finding new music an important issue for many people.

However the question asked was fairly vague, someone may have listened to 100,000 minutes of music in a year, and someone else 20,000, but they both may put that they listen to a lot of music. Not specifying a numerical value means that the data is less accurate as “a lot of music” is very subjective.

2. How difficult is it to find new music

Chart, bar chart, histogram

Description automatically generated

On a scale from 1 to 10, 12 participants responded to the question, "How difficult is finding new music to listen to," with a mean score of 5.583. This suggests that finding new music was often a challenge for participants. It is surprising that Spotify's recommendation system is not more effective in recommending new music to customers given that the company is massive and has access to cutting-edge technology.

However, it is important to keep in mind that the subjective nature of the rating scale and the 10-point scale's lack of precision could lead to bias in the survey's results. Furthermore, the results may not be generalizable to the full population due to the small sample size of only 12 participants, which may not be indicative of Spotify's total user base. When analysing the findings and evaluating the merits of the recommendation algorithm, these constraints should be taken into account.

3. How do you find new music?

Participants were asked how they find out about new music. The results show that for nearly all of those asked (11 out of 12), used Spotify. It also shows that Spotify is much more commonly used than any other music app (11 participants using Spotify vs. 2 using any other music app), and that it is used far more than any other source. This gives a strong indication of the power that Spotify has in influencing the new music that people listen to.

However, it is important to keep in mind the results may not be generalizable to the full population due to the small sample size of only 12 participants, which may not be indicative of the general population.

4. How good are music apps at recommending new music?

Chart, bar chart, histogram

Description automatically generated

In response to the question "How good would you say your preferred music app's ability to recommend new music to you is?" with a 10-point scale where 1 is extremely bad and 10 is amazing, 12 participants provided responses with a mean score of 4.6. This indicates that, on average, participants rated their preferred music app as somewhat poor at recommending new music.

The relatively wide range of replies indicates that participants have a range of thoughts on the ability of their preferred music app to make new music recommendations. While some participants have given their app good ratings, others have given it low ratings; this variation in opinions may be brought on by personal preferences, listening habits, and app experiences.

Furthermore, the mean score of 4.6 is lower than the midpoint of the scale, indicating that a significant number of participants rated their app poorly. This suggests that many users are not satisfied with the recommendations provided by their preferred music app and that there may be a need for improvements in the recommendation algorithm.

In conclusion, according to the survey's findings, participants generally thought their favourite music app did a mediocre job of introducing them to new music. This indicates that there are a variety of user opinions and that a sizable portion of users are dissatisfied with the recommendations made by their app. Even accounting for drawbacks of the questionnaire, the effectiveness of the recommendation algorithm and the concerns that users indicated should be taken into account.

5. Have do you have positive experiences from music?

Chart, pie chart

Description automatically generated

Graphical user interface, text, application

Description automatically generated

A picture containing graphical user interface

Description automatically generated Background pattern

Description automatically generated

In response to the question "Have you experienced positive impacts of listening to music?", 83.3% of participants answered affirmatively, providing reasons such as "It helps me regulate my mood and cheers me up," "Relaxing, feel better on a bad day," "More motivation and concentration," and "Form of Therapy & Relaxation". This indicates that a significant majority of participants have experienced positive impacts from listening to music, such as improved mood, relaxation, and increased motivation.

These responses demonstrate how music is significant to a wide range of people and how it significantly affects their wellbeing. However, if a music recommendation algorithm is ineffective at introducing users to fresh and varied music, it may prevent them from benefiting from the benefits of music listening. For instance, users might not find new musicians or genres that could have a positive effect on their motivation or mood if a wide variety of music is not recommended to them.

#### In conclusion, this survey's findings show that a sizable majority of respondents have benefited from listening to music. This emphasises the significance of music in people's lives and implies that poor music recommendations may be detrimental to users' wellbeing.

#### Questionnaire Conclusion

In conclusion, the four questionnaire questions' results show that participants had trouble discovering new music, thought their favourite music app did a poor job of recommending it, and that listening to music had positive effects on their lives. However, there are problems with the subjective nature of the rating scales and the small sample sizes of the questionnaire.

### These findings imply that methods and tools for finding new music need to be improved, and a more effective music recommendation algorithm, or just an alternative is also required. Furthermore, when analysing the findings and evaluating the efficiency of the recommendation algorithm, it is important to take into account the questionnaire's constraints, such as the subjective rating scales and small sample sizes.

### The answers to these survey questions demonstrate the significance of music in people's lives and the shortcomings of the current music recommendation algorithms.

## Identification of Perspective Users

### Feedback

The users who I will receive feedback from will be my classmates, so 17-18 year old students. This comes with advantages and disadvantages. One advantage that comes from having a young demographic for testing my project is that they will be more likely to be using music streaming services and more up to date technology, allowing them to access a greater amount of music. This can provide valuable insights into the preferences and behaviours of young users, which can inform the design and development of the recommendation algorithm. Additionally, because me and my users are in regular contact, we can facilitate direct communication and collaboration, allowing for a more efficient and effective feedback loop.

The drawback of having such a limited age demographic is that it might not be representative of all music streaming platform users. Users who are older may have different preferences and listening styles, and users from various geographic or cultural backgrounds may have different needs and expectations. As a result, the results may be less generalizable and the recommendation algorithm may be less successful. It might be necessary to gather feedback from a wider range of users in order to address this problem, as well as to take into account the unique requirements and preferences of various age groups, geographical locations, and cultural groups when designing and developing the algorithm.

### In conclusion, having classmates between the ages of 17 and 18 as potential users on a music recommendation project has several benefits, including the possibility for open dialogue and collaboration as well as their familiarity with music and technology. The results may not be as generalizable as they could be because they might not be a representative sample of all users. It will be crucial to take into account the unique requirements and preferences of a variety of users in order to maximise the relevance and effectiveness of the algorithm.

### General Users

My program’s main users will just be people interested in music. As my program will both recommend music and also display information about artists.

Users who have a big interest in music will find use in being able to find new music outside of just Spotify, as they will be able to set personal preferences about what music they would like to be recommended e.g., how acoustic/energetic it is.

And users who are less interested in music may enjoy it more after using my program; which, like mentioned above, may have various positive effects on a persons’ mental health. However they may just simply be curious and want to see some information about their favourite artists. Also just because a person has less interest in music, an app that can recommend new music to a person may still have some use, just less frequently.

As shown by my questionnaire, a large proportion of people consider themselves having an interest in music meaning my programs potential userbase is very broad.

### Potential Solutions to my Proposed Problem

|  |  |  |
| --- | --- | --- |
| Solution | Positives | Negatives |
| Website | More easily accessible to users assuming I have put the site on the internet and secured a domain and server.  Allows for more freedom in terms of graphical interface with many built in functions and easily manipulatable objects.  Support for mobile users assuming I have added support for it.  Requires no downloading from the user because they are accessing information from a server running from me or the cloud | Requires HTML CSS and JS knowledge to make my website up to date and effective.  I have limited knowledge of HTML and CSS meaning I would have to put in much more work to create a complex product.  Limited knowledge of the Spotify Web API  Would require internet access to use |
| Python Application | Have much more experience with creating python application as well as multiple projects using PyGame.  The lack of in built GUI objects makes my UI incredibly easy to edit and change to my original vision  Python being a high level language makes my code much easier to understand.  The entire program can mostly be built in one language rather than 3 or 4  Certain features such as the song recommendation could be used without an internet connection | Lack of GUI objects means I have to program them all from scratch  Requires downloading making it harder to access for casual users  Lack of animations or a more fluid GUI experience |

However overall, I decided on using python for my project. I started with trying to create a website and many problems became apparent to me; that is not to say that I have not encountered problem with using python, but my progress is much faster and I believe I will end up with a more polished and complex product.

## Explaining the individual processes, data and solutions I chose for my program

#### User Interface (UI)

For my UI I had 2 main choices to choose from to display information to my users. PyGame and TKinter; in the end I chose PyGame. Although my program is not a video game, PyGame has many advantages over TKinter.

|  |
| --- |
|  |
| Library | Advantages | Disadvantages |
| Pygame | Designed specifically for creating games and interactive applications | May be more difficult to learn and use compared to Tkinter. |
|  | Generally faster and more efficient than Tkinter, especially for rendering graphics | Not included with the standard Python distribution, so it must be installed separately |
|  | Has a wider range of features and functionality, including support for sprite rendering |  |
|  | Has a larger and more active community of users and developers |  |
| Tkinter | Included with the standard Python distribution, so it does not need to be installed separately | Less efficient and slower than Pygame for rendering graphics. |
|  | Generally easier to learn and use compared to Pygame | Has a more limited range of features and functionality compared to Pygame |
|  | Well-suited for creating simple GUI applications |  |

Separate installation is not a problem for me or my users as the completed program can be turned into an executable without needing separate libraries or the need to add python to path/pip install every third party library. Although Tkinter may be slightly simpler and easier to use, I am already familiar with the syntax of PyGame and enjoy the better flexibility of the library. In my personal opinion, Tkinter looks very dated, while PyGame allows you to customise your display much more. However PyGame does not have in built structures like buttons, menus and input boxes meaning I have to code them myself.

In conclusion, I chose PyGame for its better efficiency, higher customisability and its more modern look.

#### UI States

I will use UI states in my programme to control which objects are shown to the user at any given time. These UI states will vary depending on the information that needs to be displayed to the user and will be determined by a UI state variable. For instance, on the program's home screen, the user must enter specific data into a text box before pressing the enter key to continue. The programme will switch to a different UI state and possibly display different objects on the screen depending on the input given by the user.

This allows less memory usage as un-needed objects do not render without being used, and allows a simple way to only display the information that is necessary.

#### Database

A database is vital for storing sensitive user data outside the program, in my database I will be storing information such as usernames and passwords.

For my user’s data, I decided to use an SQL database.

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| SQL Queries can be used to retrieve large amounts of records from a database quickly and efficiently | Not well-suited for handling unstructured data, such as images, videos, and audio files |
| Uniform language that is easy to learn and use, even with a limited knowledge of SQL |  |
| Easy to use and integrate with python |  |

My project will not be using unstructured data, so SQL is an overall good choice.

#### Hashing

Hashing is the practice of taking a string or input key and representing it with a hash value, which is typically determined by an algorithm and constitutes a much shorter string than the original. The hash value is used to index the original value or key and make it easier to find or employ the original string.

In my project I must use hashing for obscuring password data to make sure that users passwords are safe and secure, so they cannot be accessed by anyone opening my database. The hashing algorithm I chose to use for my project was SHA 256. A widely used and efficient hashing algorithm.

The input for hashing would be a users’ password e.g., “password123,” the hash code outputted would be “ef92b778bafe771e89245b89ecbc08a44a4e166c06659911881f383d4473e94f,” a completely unintelligible output that is unable to be translated back into the original string. The user would input their password, the password would then be hashed and compared to the hash value that is already in the database. This means that the users’ password would not have to be in the database preventing security issues.

Hashing is easy to integrate into python with the library Hash lib.

#### Object Oriented Programming

The majority of the my program will be based off objects I create and reuse throughout my program, most of the procedural code in my program will be in my main file, the backbone of the program will be based on object oriented programming.

Object-oriented programming is a programming paradigm that is based on the concept of "objects," which can contain data and code. One of the main advantages of object-oriented programming is that it allows for code reusability. This means that I can reuse code that has already been written, which can save time and reduce the amount of code that needs to be written. Another advantage of object-oriented programming is that it allows for better code organization. By breaking code down into smaller, more manageable pieces, I can more easily understand and maintain their code.

When breaking down my large program into smaller problems, it can be extremely useful to essentially solve each problem by creating an object that I can reuse, especially for my GUI.

#### Gathering data on songs

In order to successfully recommend a new song to a user, my project will need to collect data from many thousands of songs. It will also need a way to evaluate how similar these songs are to each other in a way that is likely to give good recommendations.

Even through Spotify’s algorithm is based on collaborative filtering, Spotify also includes an evaluation of each song along ten different song attributes. These include:

* Acousticness: A measure of how acoustic the track is, on a scale from 0.0 to 1.0. The higher the value, the more acoustic the track is.
* Danceability: A measure of how danceable the track is, on a scale from 0.0 to 1.0. The higher the value, the more danceable the track is.
* Energy: A measure of the intensity and activity of the track, on a scale from 0.0 to 1.0. The higher the value, the more energetic the track is.
* Instrumentalness: A measure of the presence of vocals in the track, on a scale from 0.0 to 1.0. A value of 0.0 indicates that the track has vocals, while a value of 1.0 indicates that the track is instrumental.
* Loudness: The overall loudness of the track in decibels (dB), ranging from -60.0 to 0.0. The higher the value, the louder the track is.
* Speechiness: A measure of the presence of spoken words in the track, on a scale from 0.0 to 1.0. The higher the value, the more spoken words there are in the track.
* Tempo: The tempo of the track in beats per minute (BPM).
* Valence: A measure of the positivity of the track, on a scale from 0.0 to 1.0. The higher the value, the more positive the track is.
* Key: the key of a song
* Liveness: judging if a song is live or not from 0 to 1

For my project, I decided to base song recommendations based on these ten attributes – if one song is very similar to another, in terms of these attributes, then a user who likes one is likely to like another. This is not a perfect way of making recommendations, for example it does not include song genre, however my main aim is to come up with a different set of recommendations to that generated by Spotify, and this approach is likely to achieve this while still making recommendations that many users will enjoy. It also allows me to gather structured data on many thousands of songs, which otherwise would be a hugely time-consuming activity.

The main way of gathering song attribute data is by using Python to access Spotify's database. By accessing different types of data available on Spotify, including track, artist, and album information, Spotipy, a Python library, makes this data collection process easier. I created a free Spotify developer account in order to use this library, and I was given a client ID and password to authenticate my access to the platform's data.

Spotify is a great tool for gathering information because it has a sizable music library that largely contains the music that users may want to access. I can also get information that might not be easily accessible through the Spotify app by using the Spotipy library. This makes it possible for me to compile a larger set of information to help users with music recommendations. Ultimately, more accurate music recommendation functionality is made possible by using Spotify as a data source and Spotipy to gather that data.

##### Process of gathering data off Spotify

For each of my users possible inputs, I decided to create 3 separate objects each storing data important to either, album, song or artist.

Data requirements to be displayed:

* Album:
  + Album art
  + Release date
  + Genres
  + Album name
  + Artist name
  + Track list
  + Label
  + Followers
  + Length
  + Popularity
  + Audio features such as: tempo, mode, key, valence, dancability, energy
* Song:
  + Track name
  + Popularity
  + Artist Name
  + Song art
  + Release Date
  + Audio features such as: tempo, mode, key, valence, dancability, energy
  + Explicit
* Artist:
  + Artist name
  + Artist Wikipedia information
  + Similar Artists
  + Top albums and art
  + Top Songs
  + Followers
  + Popularity

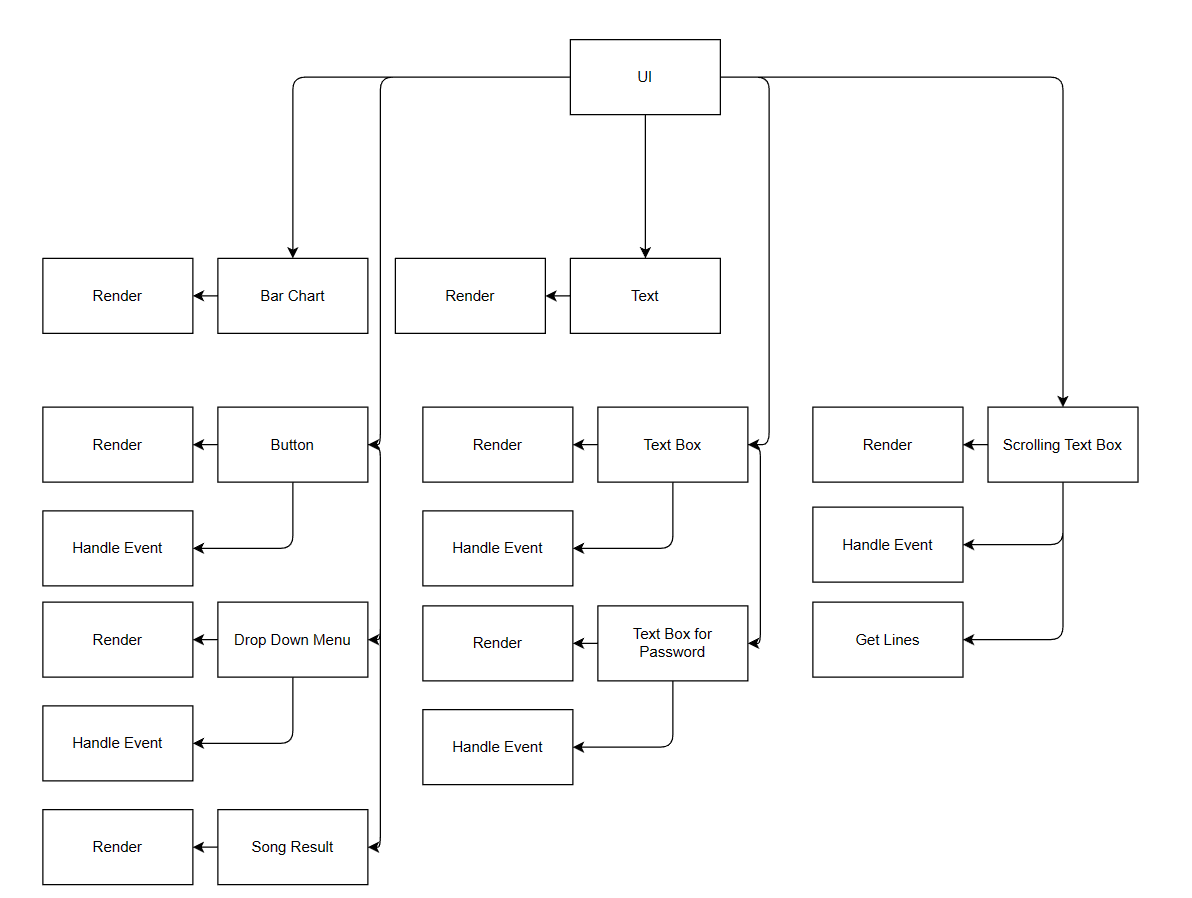
#### Information outside of Spotify - Wikipedia

Python libraries like requests and Beautiful Soup were used to retrieve data from Wikipedia. While the Beautiful Soup library parsed the HTML code to provide a more structured and user-friendly data format, the requests library allowed for the retrieval of HTML from a given URL. Wikipedia is a useful tool for compiling information on a variety of topics because it contains a vast amount of information. Information on even relatively obscure artists is included in this. It is important to keep in mind, however, that despite Wikipedia's impressive depth of knowledge, the data's accuracy is not always ensured as it depends on the quality level and dependability of the sources used to produce the content.

## Design

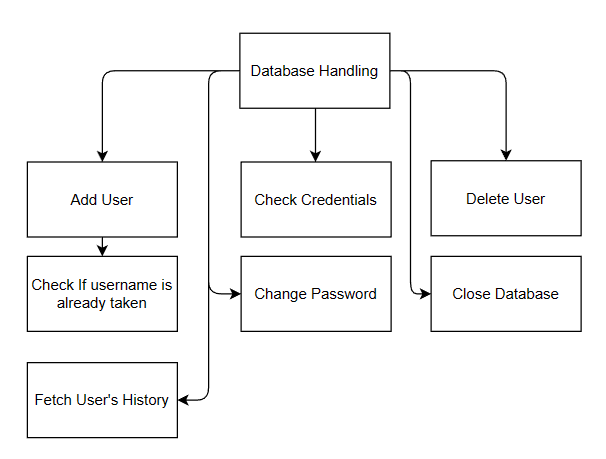
### Program Structure:

#### UI

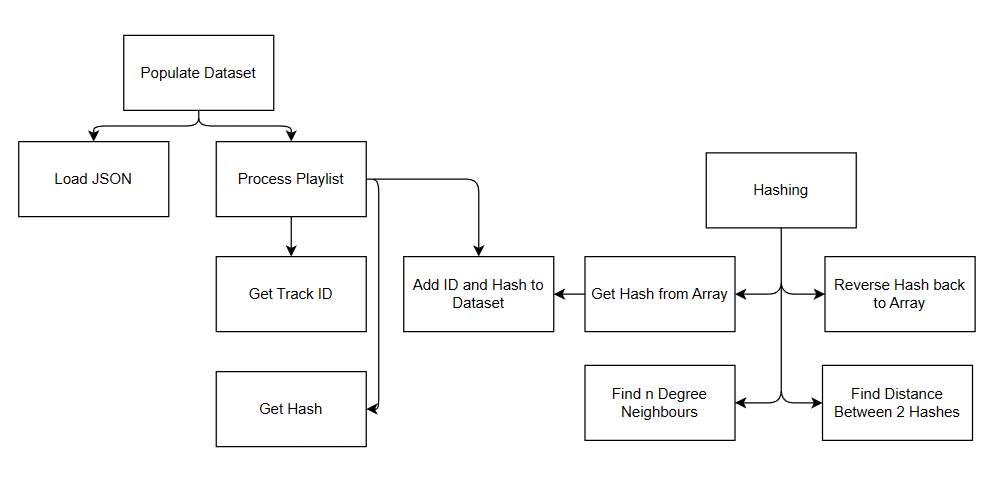


#### Gathering Music Data

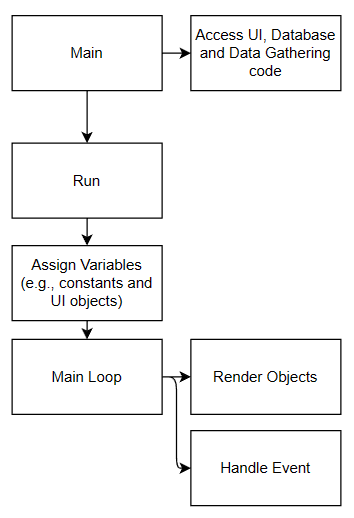
#### Database Handling



#### Populating my dataset and Hashing



#### Main Function



### Flowcharts showing how users interact with each UI State

#### Login

Diagram

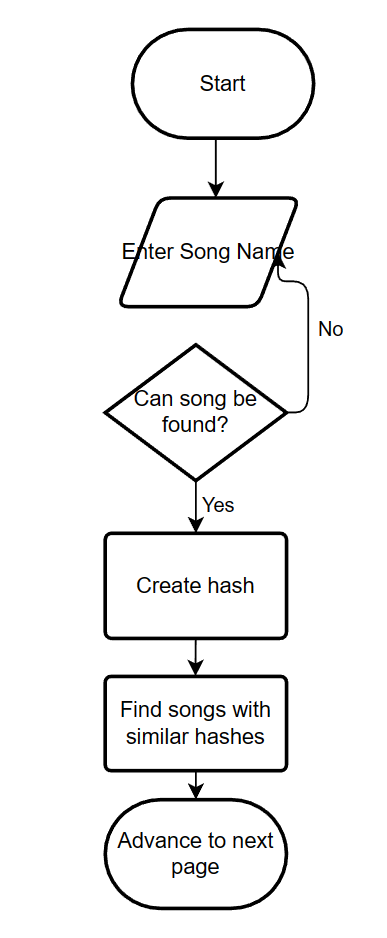
Description automatically generated

#### Music info

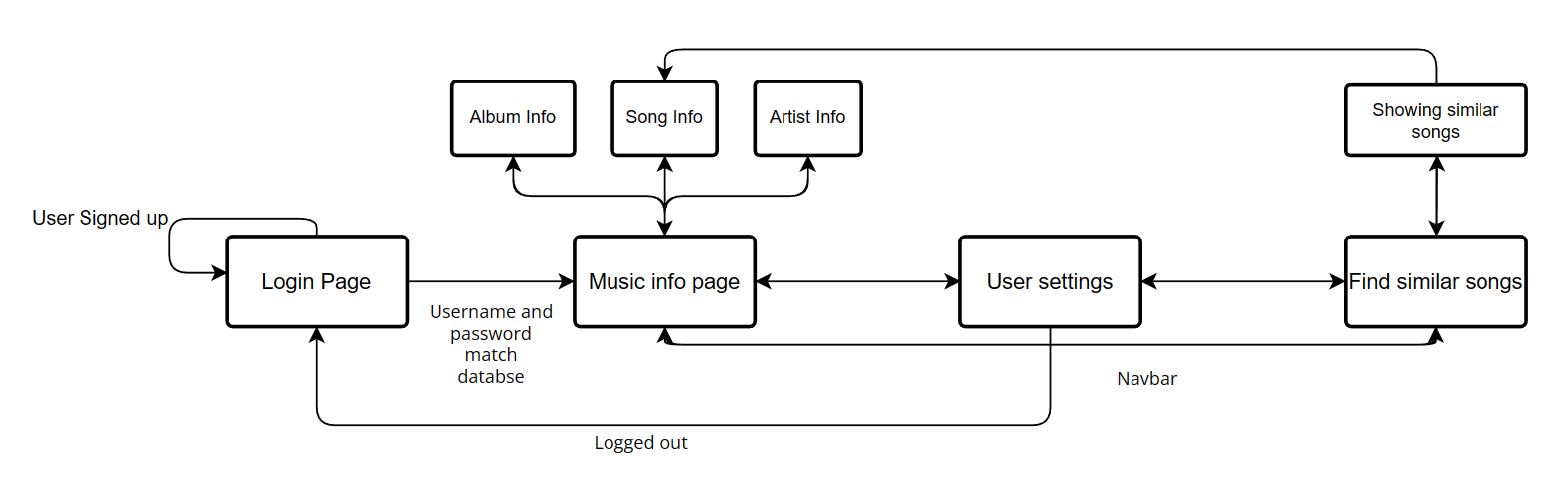
Diagram

Description automatically generated

#### Music recommendation



### UI State Flow Diagram



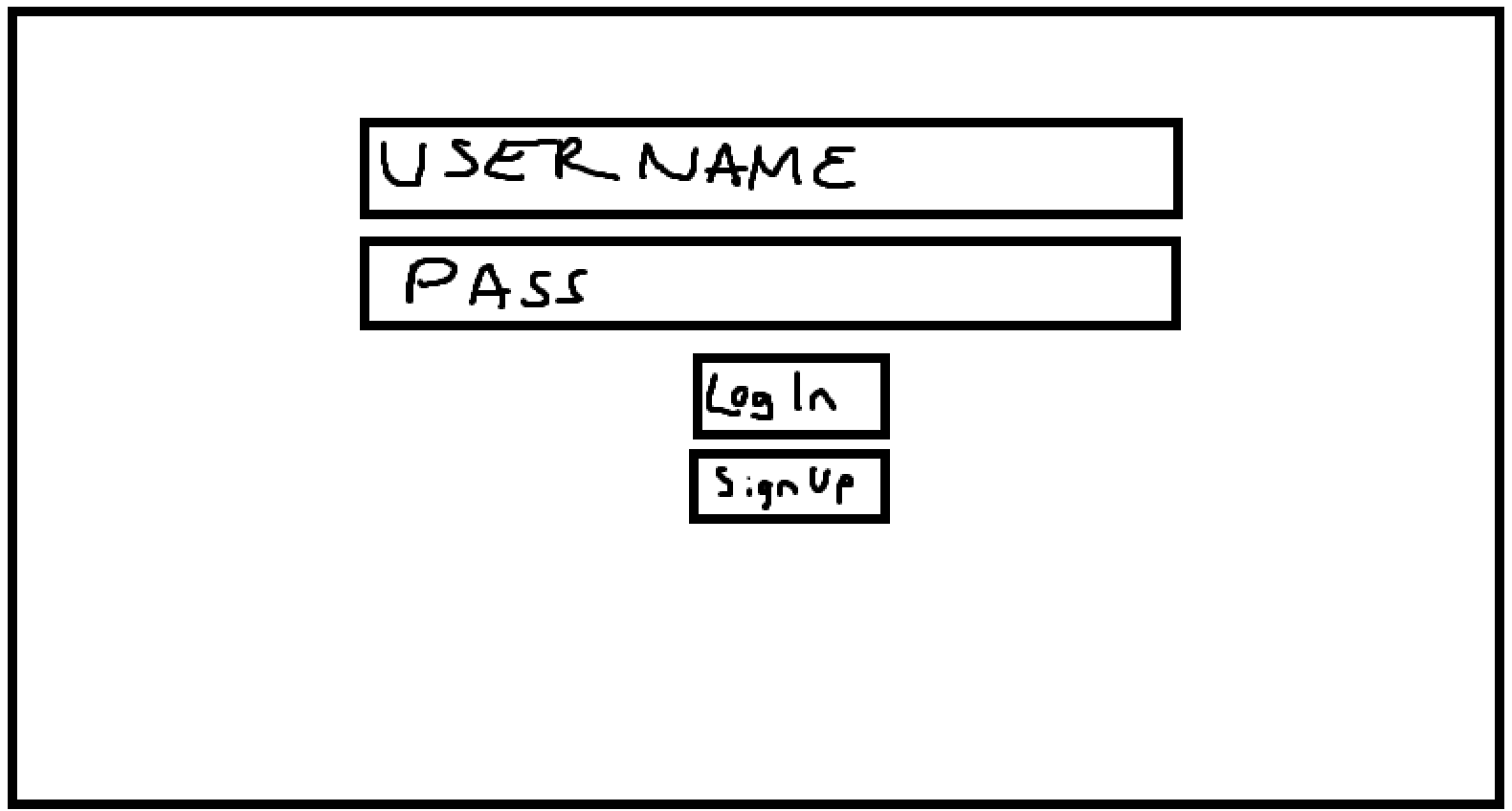
#### UI Objects

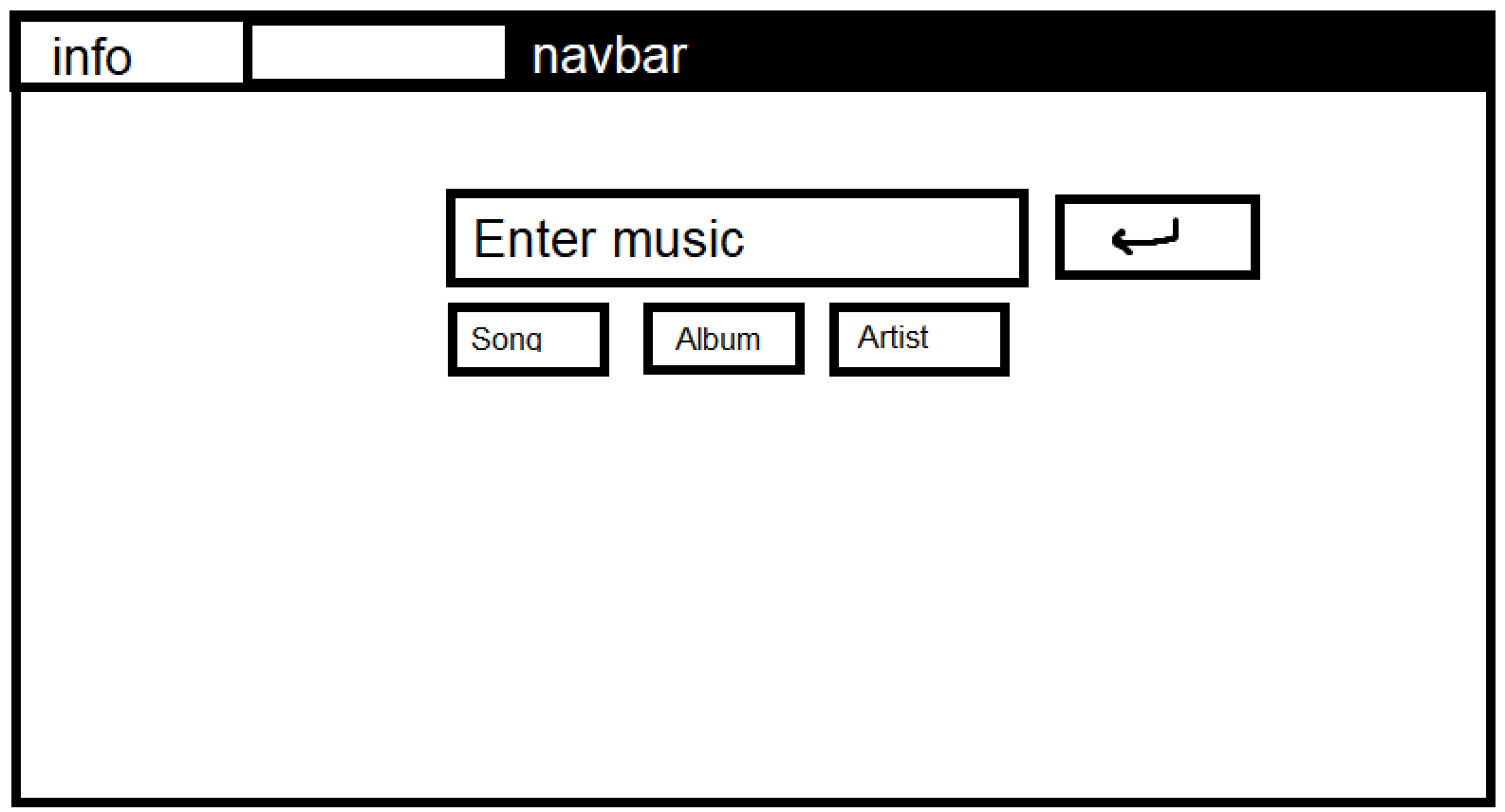
As mentioned before, PyGame does not have integrated objects for a lot of what I want my UI to display, meaning I have to code each object in individually. Although this takes more time than using Tkinter, it allows me customise each object to the exact needs of my program. These are the objects I have had to code into my UI:

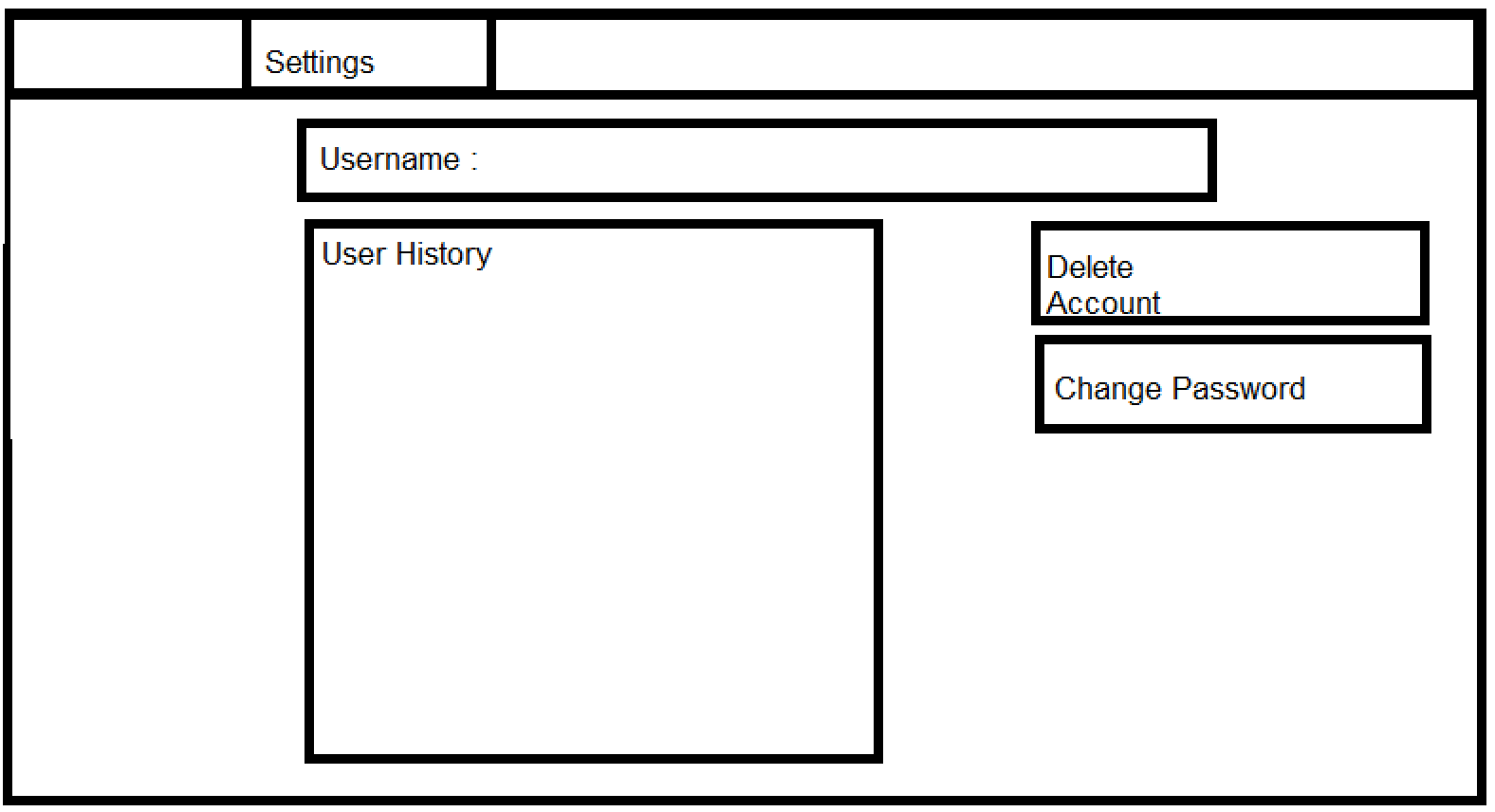
* The Text\_Info\_Box class is used to display a block of text on the screen. It has a scroll attribute that allows the text to be scrolled up or down by adjusting its vertical position. The handle\_event method listens for scroll events (mouse wheel scroll up and down) and adjusts the scroll value accordingly. The render method is responsible for drawing the text box and displaying the text within it.
* The Button class is used to create a button that can be clicked by the user. It has a pressed attribute that keeps track of whether the button is currently being pressed, and a colour attribute that changes to a different value when the button is pressed. The handle\_event method listens for mouse button down events and sets the pressed attribute to True and the colour attribute to a different value if the mouse is hovering over the button. The render\_button method is responsible for drawing the button and displaying the button's text.
* The DropDown class is a subclass of the Button class, meaning it inherits all the attributes and methods of the Button class and can add additional functionality on top of that. It has a selected attribute that keeps track of whether the dropdown menu is currently open or closed, and a list of choices that represent the options available in the dropdown menu. The handle\_event method has been modified to toggle the selected attribute and to allow the user to select an option from the choices list by clicking on it. The render\_button method has also been modified to display the dropdown menu and its options when the selected attribute is True, and to display the text of the selected option or the default text of the button when the selected attribute is False.
* The TextBox class creates an interactive text input field that displays text the user enters. When the user clicks within the field, they can begin typing. Pressing the "Enter" key will store the entered text in an output variable and clear the field. Pressing the "Backspace" key will delete the last character in the field. When the user clicks outside of the field, they can no longer enter text and the field becomes inactive. The appearance of the field changes to reflect whether it is active or inactive.
* The BarChart class creates a few rectangles with a text box underneath. It takes in variables for position, width, height, text and the values to create a bar chart to scale with the values entered; e.g., if you were to input a max bar value of 100, and have the actual value of 25, the bar chart will display a rectangle to scale with the max height, and display a different rectangle on top proportional to the max height/4. It then displays the text under to show what the graph is representing
* The NavBar class is simply an array of buttons, which creates a number of buttons depending on the variables entered. Its use is to display the different menus at the top of the screen to change the UI state.
* PasswordBox, is a child class of the text box class, but instead of displaying the text on the screen as its written, it takes however many characters have been entered and displays them as stars so it is not visible to other people.
* The Text class creates text on the screen, with attributes like position, width and text size. It also creates a rectangle behind the text to make it more visible

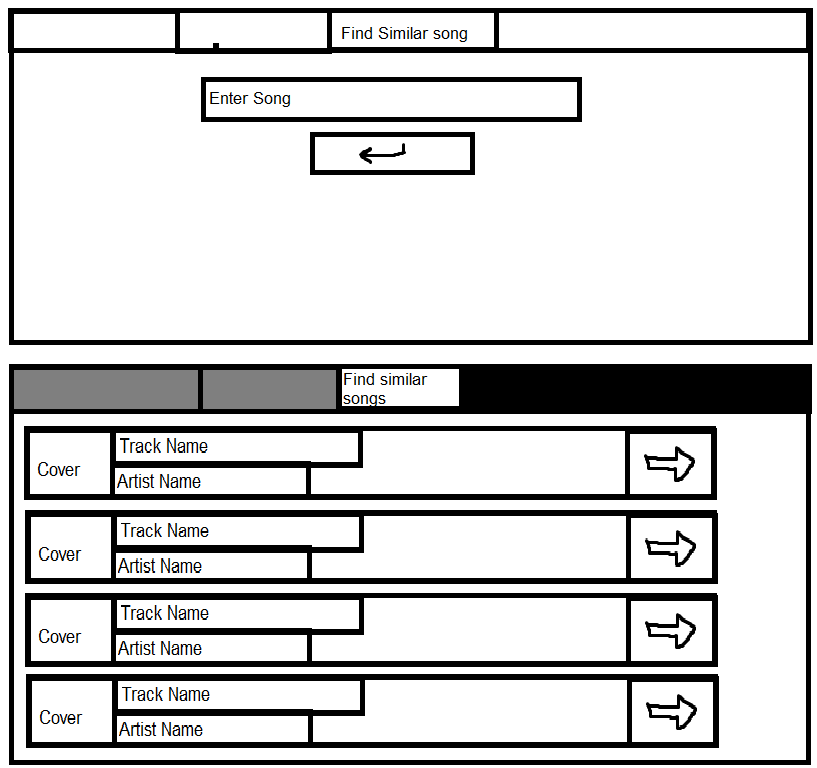
### Mock ups for UI

Before creating my UI, I tried to create what I thought would look like a good user interface in MS paint; I based most of my UI around my mock ups.









### Internal and external data storage

Internally, within my main file I store data such as:

* UI data: each object that appears on the screen must be assigned data such as, position, width, height and text. All this data is stored internally. Additionally data about the font I want to use for my UI is stored in my program
* User data: although my users password and user data is stored externally in a database, multiple times throughout my code the database is accessed and local variables such as “username” and “password” are assigned internally to verify credentials and edit user data
* Music data: when a user enters a song data must be stored in my program to be displayed on the screen. This data includes attributes like: name, id etc.
* UI state: keeping track of what screen the user is on is essential to make the program functional and keep the program from taking up memory and CPU power rendering and handling objects that are not needed

Externally I store:

* User database: my database of users includes information such as, username, password and user history data. This is necessary to be stored outside of my program because otherwise whenever the program is terminated, all user data would be lost
* Song dataset: when a user enters a song to find recommendations, my program must check the hash of the entered song to find recommendations. To examine the data of an extremely large number of songs in real time is not feasible and would take a very large amount of time; so my dataset is essential for the function of my algorithm and saving time.
* Images: When I need to display images of music cover art on my screen, pygame requires a directory of the image I want to display. So I must convert the cover art URL into an image in my programs directory

### Recommendation Algorithm

Finding an efficient way to sort songs based on a similarity was a difficult task; however I used the spotipy audio features function to first gather data about the sound of a track. This data includes the 10 parameters mentioned above.

After this the 10 parameters must be normalised to a value 0 to 1, e.g., if the loudness of a song is -35, I would add 60, making the loudness 25. I would then divide the value by 60, giving us a normalised value of 0.42.

I now have an array of 10 parameters that look something like this: [0.33, 0.532, 0.16, 0.8, 0.0263, 0.00101, 0.709, 0.101, 0.218, 0.311]. I now get the hash of this array by multiplying each value in the array by 5, and converting it to an integer. Now the array looks something like this: [1, 2, 0, 4, 0, 0, 3, 0, 1, 1]. Every parameter is sorted into a bucket with a value of 0-4.

The next step ins converting this into a base 5 number. This is done by reversing the array and multiplying them all by 5 to a power depending on the position of the array. At this point we have t he hash value and it would look something like this: 2797256

Now we have the hash value me must find neighbours of this hash value, as a way to find songs with only a small difference in their sound based on the original parameters. Now I run a function to find the 2nd degree neighbours of my hash.

## Testing

To ensure that my code is robust, I developed a comprehensive test plan. This plan covers the following areas:

* The registration process
* The login process
* The user entering a song, album or artist and initiating the search
* The code working under conditions where there is not access to Wikipedia or Spotify

For the elements of code that require a user to input data, including their user information and the name of the song, album, or artist that they are interested in, I ensured that the test plan covered the following kinds of input data:

* Normal data (e.g. the name of an artist), used in Test IDs 1,2,11,12,17-25,27,28,32 & 33
* Erroneous data (e.g. blank entries, special characters where normal text is expected), used in Test IDs 4-8, 10, 12-15 and 16
* Boundary data (e.g. passwords that do not meet the minimum requirements for length – namely four characters or more), used in Test IDs 3, 9, and 30

#### Test Plan

The table below shows that unique tests that the code must pass in order for me to be confident that it is robust.

| **Test ID** | **Description** | **Test Data** | **Expected Results** | **Test done successfully and Time Stamp** |
| --- | --- | --- | --- | --- |
| New user registration | | | | |
| 1 | Check that a new user can register with normal data – both normal username and normal password | Name: ‘Jacob17’  Password: ‘H3ll0’  Click ‘Sign Up’ | 1. Message telling the user that they have been registered 2. User data added to the database of users | Yes  00:00 |
| 2 | Check that a new user cannot use the same username as an existing user | Name: ‘Jacob17’  Password: ‘G00dby3’  Click ‘Sign Up’ | 1. Error message telling user that there is already a user of that name and asking for another username 2. No user data added to the database | Yes  00:00 |
| 3 | Check that a new user cannot use a username with fewer than four characters | Name: ‘B0b’  Password: ‘Hell0'  Click ‘Sign Up’ | 1. Error message telling the user that their proposed username is invalid and explaining what the validation rules are 2. No user data added to the database | Yes  00:00 |
| 4 | Check that a new user cannot register using a null username | Name: <null>  Password: Hell0  Click ‘Sign Up’ | Same as test ID 3 | Yes  00:00 |
| 5 | Check that a new user cannot use a username with a special character | Name: #478783 or  Password: S33Y0u  Click ‘Sign Up’ | Same as test ID 3 | Yes  00:00 |
| 6 | Check that a new user cannot use a username with a space | Name: ‘Hi Th3r3’  Password: ‘S33Y0u’  Click ‘Sign Up’ | Same as test ID 3 | Yes  00:00 |
| 7 | Check that a new user cannot use a null password | Name: ‘Jackie3’  Password: <null>  Click ‘Sign Up’ | 1. Error message telling the user that their proposed password is invalid and explaining what the validation rules are 2. No user data added to the database | Yes  00:00 |
| 8 | Check that a new user cannot use a password involving special characters | Name: ‘Marly01’  Password: ‘#92kl’  Click ‘Sign Up’ | Same as test ID 7 | Yes  00:00 |
| 9 | Check that a new user cannot use a password with insufficient characters | Name: ‘Theo10’  Password: ‘B3y’  Click ‘Sign Up’ | Same as test ID 7 | Yes  00:00 |
| 10 | Check that a new user cannot use a password with spaces | Name: ‘Theo10’  Password: ‘H1 Th3r3’  Click ‘Sign Up’ | Same as test ID 7 | Yes  00:00 |
| Login | | | | |
| 11 | Check that a user can login with normal data – both normal username and normal password, and a user that has already registered | Name: ‘Jacob17’  Password: Hell0  Click ‘Login’ | 1. Closing of the login UI object 2. Opening of the Search window | Yes  00:00 |
| 12 | Check that a user cannot log in if there is no one with that username | Name: ‘Christian93’  Password: ‘Lat3rM8’  Click ‘Login’ | 1. Error message telling the user that their username and/or password cannot be found | Yes  00:00 |
| 13 | Check that a user cannot login with a username that includes special characters | Name: ‘#ksfuio1934’  Password: ‘T3st17’  Click ‘Login’ | Same as test ID 12 | Yes  00:00 |
| 14 | Check that a user cannot login with a password that contains special characters | Name: ‘Jacob17’  Password: ‘#ksfuio1934’  Click ‘Login’ | Same as test ID 12 | Yes  00:00 |
| 15 | Check that a user cannot login with a <null> username | Name: <null>  Password: ‘Hell0’  Click ‘Login’ | Same as test ID 12 | Yes  00:00 |
| 16 | Check that a user cannot login with a <null> password | Name: ‘Jacob17’  Password: <null>  Click ‘Login’ | Same as test ID 12 | Yes  00:00 |
| Search – Song/Artist/Album | | | | |
| 17 | Check that the user can get information with a real song | Toggle: ‘Song’  Query: ‘Blind’  Click on ‘Enter’ | 1. Information for that song is displayed 2. The search for that song is added to the search history in the database | Yes  00:00 |
| 18 | Check that the user is informed if no song with that title can be found | Toggle: ‘Song’  Query: ‘Blund’  Click on ‘Enter’ | 1. An error message is displayed telling the user that no song with that title was found | Yes  00:00 |
| 19 | Check that the user can get information with a real artist | Toggle: ‘Artist’  Query: ‘Panchiko’  Click on ‘Enter’ | 1. Information for that artist is displayed 2. The search for that artist is added to the search history in the database | Yes  00:00 |
| 20 | Check that the user is informed if no artist with that title can be found | Toggle: ‘Artist’  Query: ‘Panchika’  Click on ‘Enter’ | 1. An error message is displayed telling the user that no artist with that name was found | Yes  00:00 |
| 21 | Check that the user can get information with a real album | Toggle: ‘Album’  Query: ‘To be kind’  Click on ‘Enter’ | 1. Information for that album is displayed 2. The search for that album is added to the search history in the database | Yes  00:00 |
| 22 | Check that the user is informed if no album with that title can be found | Toggle: ‘Album’  Query: ‘To be kynd’  Click on ‘Enter’ | 1. An error message is displayed telling the user that no album with that name was found | Yes  00:00 |
| User Settings (logged on as ‘Jacob17’) | | | | |
| 23 | Check that the user cannot delete their account if they do not enter the correct password | Enter Password: ‘Hello’ (note that this is incorrect)  Click on ‘Delete Account’ | 1. A message is displayed telling the user that their password is incorrect 2. The account is NOT deleted from the database of users | Yes  00:00 |
| 24 | Check that the user can change their mind about deleting the account if they are not sure they want to | Click on ‘Delete Account’  Click on ‘No’ to show that the user is NOT sure they want to proceed | 1. The User Settings window is displayed 2. The account is NOT deleted from the database of users | Yes  00:00 |
| 25 | Check that the user can delete their account | Click on ‘Delete Account’  Enter Password: ‘H3ll0’  Click on ‘Yes’ to confirm that the user is sure they want to proceed  Note: after test 25, is completed successfully, redo Test 1 to re-create the user ‘Jacob17’ with password ‘H3ll0’, and Tests 17, 19, and 21 to create a search history | 1. A message is displayed telling the user that their account is now deleted 2. The Login/Sign Up window is displayed 3. The account is deleted from the database of users | Yes  00:00 |
| 26 | Check that ‘Clear History’ works | Click on ‘Clear History’ | 1. The ‘User History’ is cleared for the User Settings 2. User history is cleared in the database | Yes  00:00 |
| 27 | Check that the user can change their password | New Password: ‘H1Th3r3’  Old Password: ‘H3ll0’  Click on ‘Change Password’ | 1. A message is displayed telling the user that their password has now been changed 2. The new password replaces the old password in the database | Yes  00:00 |
| 28 | Check that the user cannot change their password if they do not know their old password (however the password is alphanumeric) | New Password: ‘G00dby3’  Old Password: ‘HiThere’  Click on ‘Change Password’ | 1. A message is displayed telling the user that their password has not been changed as their old password is incorrect 2. The old password in the database is not changed | Yes  00:00 |
| 29 | Check that the user cannot change their password if they use <null> as the old password | New Password: ‘G00dby3’  Old Password: <null>  Click on ‘Change Password’ | Same as test ID 28 | Yes  00:00 |
| 30 | Check that the user cannot change their password if new password has fewer than four characters | New Password: ‘By3’  Old Password: ‘H1Th3r3’  Click on ‘Change Password’ | 1. A message is displayed telling the user that their password has not been changed as their new password does not meet the password rules 2. The old password in the database is not changed | Yes  00:00 |
| 31 | Check that the user cannot change their password if new password contains special characters | New Password: ‘#S33Y0u’  Old Password: ‘H1Th3r3’  Click on ‘Change Password’ | Same as test ID 30 | Yes  00:00 |
| Find Similar | | | | |
| 32 | Check that the user can find similar songs if they enter a real song | Enter Song: ‘Blind’  Click on ‘Enter’ | 1. A list of similar songs is displayed 2. The search for that similar song is added to the search history in the database | Yes  00:00 |
| 33 | Check that the user is informed if the song title is not found | Enter Song: ‘Blynd’  Click on ‘Enter’ | 1. A message is displayed telling the user that the song could not be found | Yes  00:00 |

#### Test Results

The above tests were all performed successfully. I have taken a video to provide evidence of this, which is available on YouTube under the following url: [insert URL]

## Evaluation

#### Outcome vs. requirements

#### Feedback

#### Discussion

#### Possible Improvements