Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

Music app in python

Group 1 Project

Yijun Fu - 200552722

Rahid Jakhra - 200540928

Rebacca Min - 200549269

Gurjot Singh - 200555982

Content

1. Introduction----------------------------------------------------------------------------------------------- 3

1.1 Objective -------------------------------------------------------------------------------------------- 3

1.2 Technologies Used--------------------------------------------------------------------------------- 3

* 1. Elements and Libraries Used-------------------------------------------------------------------- 3

1.4 Dataset------------------------------------------------------------------------------------------------ 5

2. Functionalities-------------------------------------------------------------------------------------------- 5

2.1 Searching--------------------------------------------------------------------------------------------- 5

* + 1. Search Function---------------------------------------------------------------------------- 6
    2. User Input----------------------------------------------------------------------------------- 6
    3. Output Handling--------------------------------------------------------------------------- 6
  1. Add Songs------------------------------------------------------------------------------------------ 7
     1. Appending to Favorite Sheet----------------------------------------------------------- 7
     2. Formatted Date---------------------------------------------------------------------------- 7
     3. Saving Changes---------------------------------------------------------------------------- 8
     4. Print Confirmation------------------------------------------------------------------------ 8
  2. Delete Songs-------------------------------------------------------------------------------------- 8
     1. Examination of Codebase--------------------------------------------------------------- 9
     2. Iteration Over Rows---------------------------------------------------------------------- 9
     3. Case-Insensitive Matching-------------------------------------------------------------- 9
     4. Conditional Deletion--------------------------------------------------------------------- 9
     5. Workbook Saving------------------------------------------------------------------------- 9
     6. User Confirmation------------------------------------------------------------------------ 9

2.4 Ranking and Sorting------------------------------------------------------------------------------ 10

2.4.1 Ranking by Released Year----------------------------------------------------------------- 10

* Filtering (selected\_year\_songs) ------------------------------------------------- 10
* Sorting (sorted\_songs ------------------------------------------------------------- 10
* Top 10 Selection (top\_10\_songs) ----------------------------------------------- 11
* Return---------------------------------------------------------------------------------- 11

2.4.2 Ranking by Genre --------------------------------------------------------------------------- 11

3. Code------------------------------------------------------------------------------------------------------ 11

4. Conclusion ---------------------------------------------------------------------------------------------- 11

* Summary of the development of the music app--------------------------------------- 11
* Problem solving of code combination----------------------------------------------------- 11
* Project's suitability for Python developers of varying skill levels------------------- 11

5. Workload Allocation---------------------------------------------------------------------------------- 12

* Details of task distribution among team members--------------------------------------------- 12

6. References ---------------------------------------------------------------------------------------------- 15

**1.Introduction**

In the era of digitalization, music applications have become an integral part of our daily lives, providing us with instant access to a vast library of songs. This Python programming report introduces the development of a music app, highlighting key functionalities such as searching, ranking, sorting, adding to favorites, and removing songs.

* 1. **Objective**

The primary goal of this project is to create a user-friendly music application that allows users to efficiently manage their music collection. By implementing essential features, we aim to provide a seamless experience for users to explore, organize, and enjoy their favorite tunes.

**1.2 Technologies Used:**

Programming Language: Python

**1.3 Elements and Libraries Used:**

* **Conditional Statements (if/else/elif):** We extensively employ conditional statements to manage user choices, particularly within the main menu. These statements ensure proper flow control based on the selected options.
* **Input Validation:** We use the pyinputplus library for input validation, ensuring that user inputs are constrained within predefined ranges. This enhances the robustness of our program by preventing invalid inputs.
* **Main Loop (while):** A central component of our code is the while True loop, serving as the main menu loop. This loop persists until the user opts to exit (triggered by selecting option 4), providing a continuous and interactive user experience.
* **Iterative Processing (for loops):** For loops are effectively employed throughout our code, notably when iterating through rows in the Excel sheet (for row in sheet.iter\_rows(...)). Additionally, they play a key role in displaying top songs and handling various data structures.
* **Data Structures (lists, strings, tuples, dictionaries, sets):** We make versatile use of diverse data structures, including lists (selected\_year\_songs, sorted\_songs, top\_10\_songs, data), strings (representing song names and artist names), dictionaries (for song attributes), and sets (e.g., {song['released\_year'] for song in data}).
* **Modularization with Functions:** Our code is well-organized through the creation of user-defined functions. Functions such as get\_top\_songs\_by\_year, search\_song, display\_song\_info, and others contribute to code modularity and readability.
* **Error Handling with Exceptions:** Exception handling is integrated into our code to manage potential errors gracefully. Notable instances include checking if the user-provided year is present in the data and raising a ValueError when loading Excel files. This enhances the resilience of our code to unexpected scenarios.

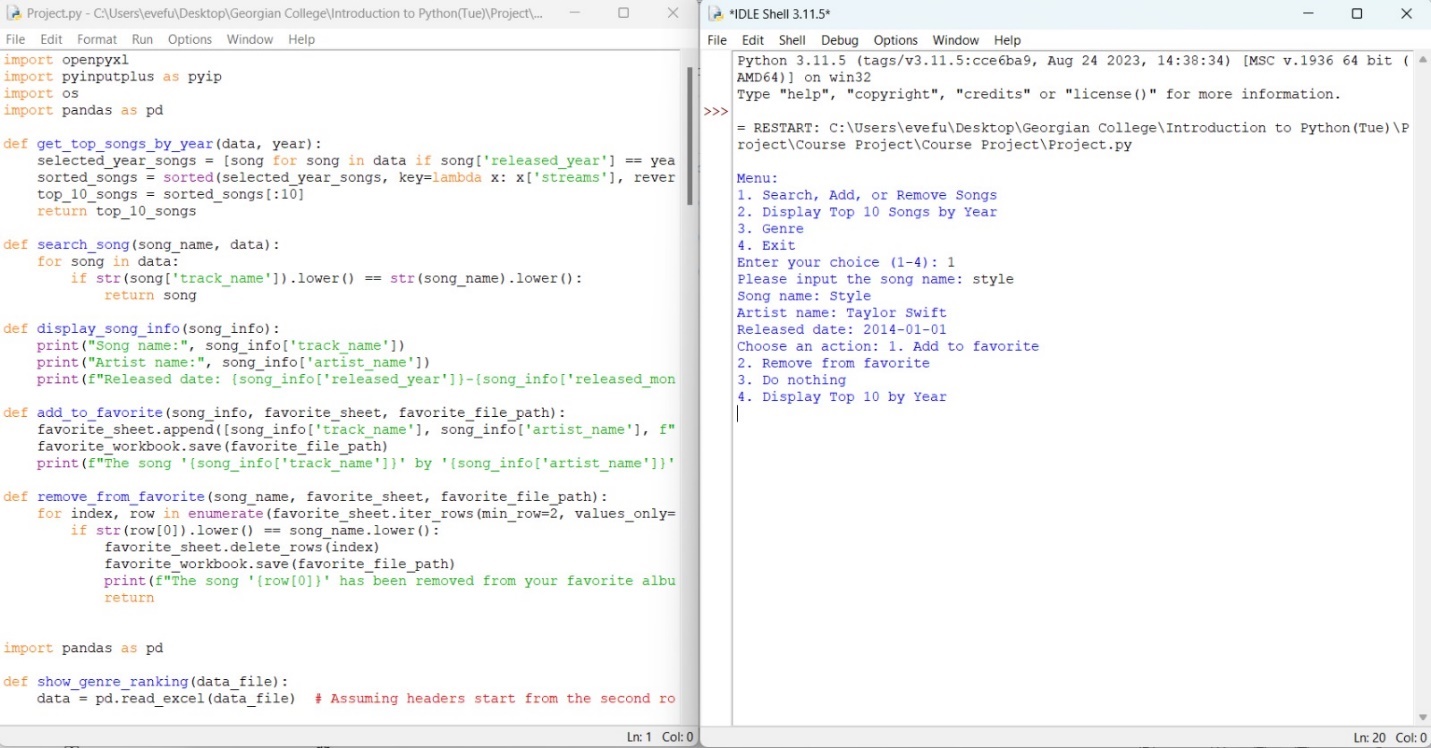
**1.4 Dataset:**

To enhance the functionality of our music app, we leverage a comprehensive dataset featuring the most famous songs of 1930 - 2023, as documented on Spotify. Distinguishing itself from comparable datasets, this resource goes beyond the typical scope by providing a rich array of features. It not only explores the distinctive attributes of each song but also offers a comprehensive view of their popularity and representation across diverse music platforms. The dataset includes details such as track name, artist(s) name, release date, inclusion in Spotify playlists and charts, streaming statistics, presence on Apple Music and Deezer, rankings on Shazam charts, and an assortment of audio features. Integrating this dataset into our application will undoubtedly enhance the user experience, offering a more extensive and insightful music library for exploration and enjoyment.

**2.Functionalities**

**2.1 Searching**

The provided Python script facilitates a music app's search functionality within an Excel dataset. It employs the openpyxl library for Excel file manipulation and pyinputplus for user input validation.



* **Search Function:** The script has a function called search\_song that finds a song in the Excel file based on what the user types. It looks through the rows, starting from the second row, and checks if the user's song name matches the names in the first column of each row. If there's a match, the whole row is given back. Meanwhile, the released date of the song contains three columns of data in database.
* **User Input:** The script asks the user to type in a song name using pyip.inputStr. This makes sure the user gives a proper name. The message "Please input the song name" helps the user know what to do.
* **Output Handling:** After searching, the script checks if it found a matching song. If it did, it takes out useful information from that row, like the song name, artist name, and release date. Then, it prints this information in a neat way. If there's no match (meaning song\_result is None), it prints a message saying no information was found for the song the user typed.

**2.2 Add Songs**

The purpose of this function is to add a song to the user's list of favorite songs. It takes the information about a song (song\_info), the Excel sheet containing the user's favorite songs (favorite\_sheet), and the file path to the favorite workbook (favorite\_file\_path).

A screenshot of a computer screen

Description automatically generated

* **Appending to Favorite Sheet:** The favorite\_sheet.append method adds a new row to the favorite sheet. The row contains information about the song, including the track name, artist name, and the formatted release date.
* **Formatted Date:** The release date of the song is formatted using an f-string to create a string in the form YYYY-MM-DD.
* **Saving Changes:** favorite\_workbook.save(favorite\_file\_path) is used to save the changes made to the favorite sheet back to the Excel workbook.
* **Print Confirmation:** After the song is added, a confirmation message is printed, indicating the track name and artist name of the added song.

**2.3 Delete Songs**

The specific focus of deletion functionality was on enabling users to remove songs from their favorite lists, accomplished through the implementation of the remove\_from\_favorite function. The following steps were undertaken:

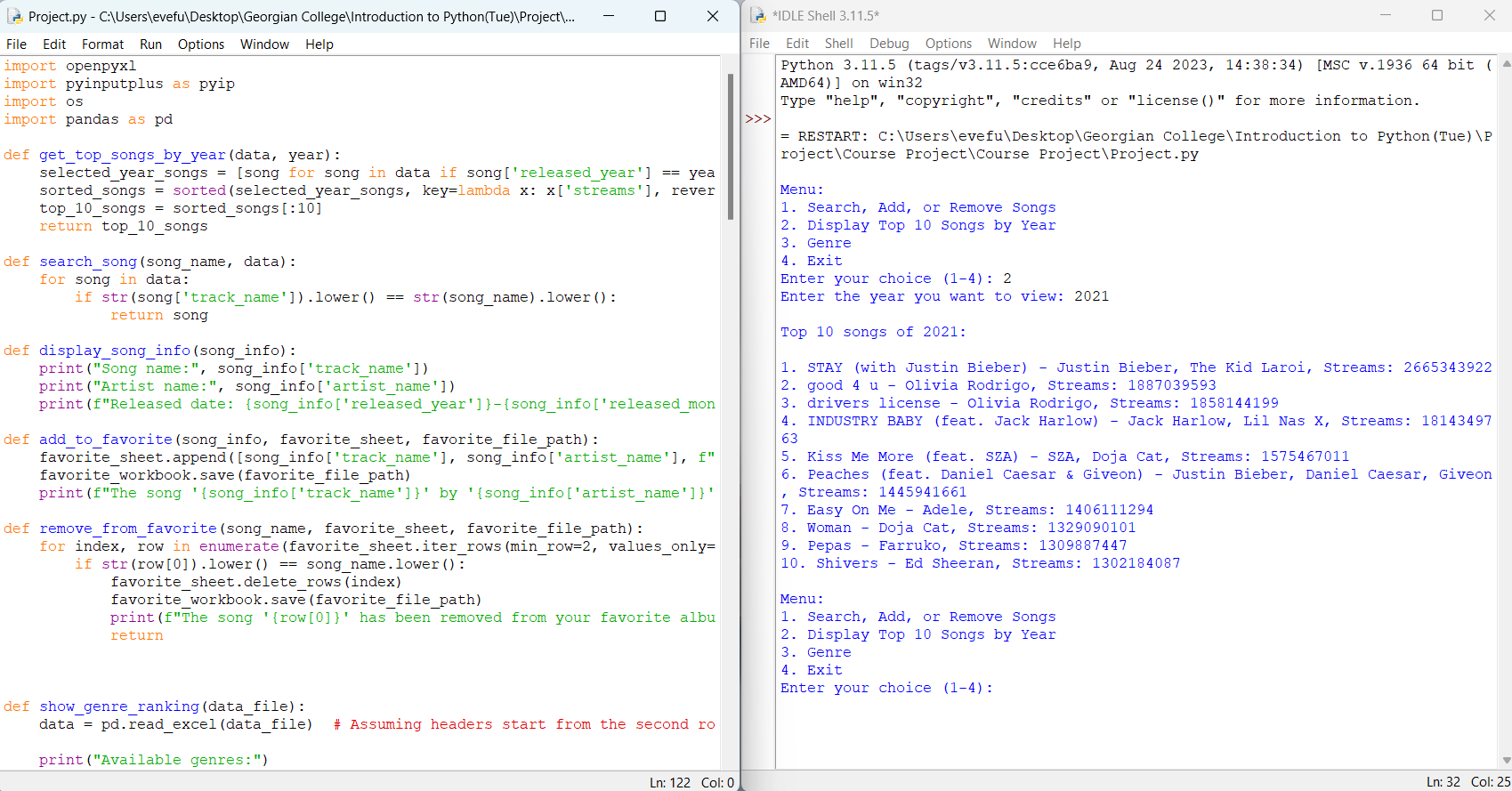
A screenshot of a computer

Description automatically generated

* **Examination of Codebase:** The existing codebase was thoroughly examined to understand its structure and identify components relevant to the deletion functionality. The objective was to seamlessly facilitate the removal of songs from the favorite list.
* **Iteration Over Rows:** A for loop was employed to iterate over the rows in the favorite\_sheet, excluding the header row. This allowed examination of each entry to determine if it matched the user-inputted song name for removal.
* **Case-Insensitive Matching:** For enhanced user-friendliness, a case-insensitive comparison was implemented by converting both the user-inputted song name and the stored song names in the sheet to lowercase using the lower() function.
* **Conditional Deletion:** Upon finding a match between the user-inputted song name and the stored song name in a specific row, the delete\_rows(index) method was utilized to remove that row from the favorite\_sheet, ensuring accurate removal of the desired song entry.
* **Workbook Saving:** The favorite\_workbook.save(favorite\_file\_path) statement was included after the deletion process to persistently save the changes, ensuring the updated favorite list is stored in the workbook.
* **User Confirmation:** A print statement was incorporated to confirm the successful removal of the song from the user's favorite album, providing feedback and clarity on the outcome of the deletion operation.
  1. **Ranking and Sorting**

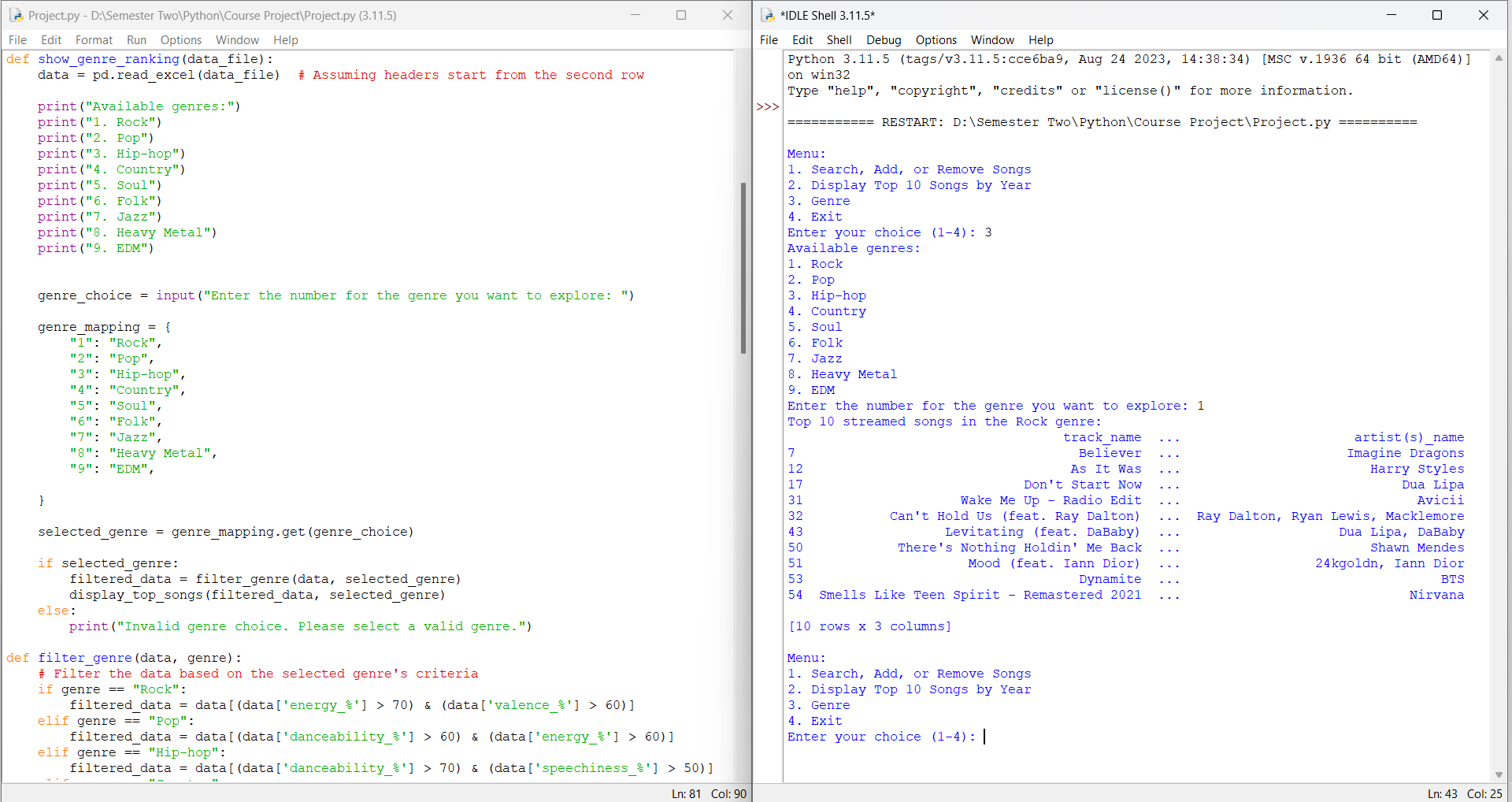
**2.4.1 Ranking by released year:**

This function retrieves and ranks the top 10 songs for a specified year based on the number of streams.



* **Filtering (selected\_year\_songs):** The list comprehension [song for song in data if song['released\_year'] == year] filters songs from the input data (data) that match the specified year. It creates a new list (selected\_year\_songs) containing dictionaries of songs released in the target year.
* **Sorting (sorted\_songs):** The sorted function is used to sort selected\_year\_songs based on the number of streams each song has (key=lambda x: x['streams']). Sorting is done in descending order (reverse=True), ensuring that the song with the highest number of streams appears first.
* **Top 10 Selection (top\_10\_songs):** The statement top\_10\_songs = sorted\_songs[:10] extracts the first 10 elements from the sorted list (sorted\_songs). These represent the top 10 songs for the specified year.
* **Return:** The function returns the list (top\_10\_songs) containing dictionaries representing the top 10 songs for the specified year.
  + 1. **Ranking by genre**

This function will display the options for genre and later show the top ten streams for that specific genre. The code creates a music genre ranking system, allowing users to explore various music genres and select top songs within each genre based on different criteria.

****

**The show\_genre\_ranking function:**  handles user interaction by displaying a list of various genres and inviting the user to choose one. Following selection, the system uses the filter\_genre function to identify tracks that match particular genre-associated features.

**The filter\_genre function:**  filters the music dataset based on the distinguishing characteristics of each genre. For example, Rock prioritises great intensity and valence, whereas Hip-hop prioritises danceability and speechiness. The algorithm refines the dataset by using genre-specific filters to highlight songs that reflect the spirit of each genre.

**Error handling:** Despite its usefulness, the system might benefit from improvements in error handling for user inputs and further modification of criteria to properly capture the complexity of each genre. Furthermore, structuring certain functions for enhanced reusability and readability will improve the system's overall resilience and maintainability. Overall, this code provides a great foundation for a music genre rating system, allowing users to dive into numerous genres and discover songs that represent their specific musical characteristics.

**3.Code:**

Please see the project.py in the zip file.

**4.Conclusion and problem solving:**

This Python programming report outlines the development of a cohesive and user-friendly music app that seamlessly integrates various functionalities. The application combines features such as searching, ranking, sorting, adding to favorites, and removing songs, providing users with a comprehensive music management experience. Combining all individual function is very challenging for us. The integration is achieved through a centralized main function, acting as the entry point for the app. The main function presents users with a menu, allowing them to navigate through different options and interact with the app's diverse capabilities. Each menu choice corresponds to a specific set of functions that handle the corresponding functionality, ensuring a streamlined and intuitive user experience. The modular design, encapsulating distinct functionalities within their respective functions, enhances code readability, maintainability, and facilitates collaboration among team members. This consolidated approach ensures that the music app is not only feature-rich but also user-centric, catering to the diverse needs of music enthusiasts.

**5.Workload Allocation:**

|  |  |  |
| --- | --- | --- |
| Name | Student ID | Task |
| Yijun Fu | 200552722 | * Combination of various functions and modules, code testing & debugging * Report Integration * Searching function * Ranking by years function |
| Rahid Jakhra | 200540928 | * Encapsulate the essence of each music genre using music traits such as energy and valence. * Optimized the code by addressing certain issues and enhancing its performance. * Fixing problems faced when combining the code. * Detailed some portion within the comprehensive report. |
| Rebacca Min | 200549269 | * Connecting the combination of Code and Debugging it * Content Integration and Report Integration * Deleting Function * Examination of the app code |
| Gurjot Singh | 200555982 | * Fixing Bugs & Issues * Adding Function * Application Testing * Report Integration |

**6.Reference:**

Elgiriyewithana, N. (2023). Top Spotify Songs 2023. Retrieved from <https://www.kaggle.com/datasets/nelgiriyewithana/top-spotify-songs-2023>

Sweigart, A. (2015). Automate the boring stuff with python: practical programming for total beginners. San Francisco, No Starch Press