A Python
Music Player

MELODIA

Grade 11 Computer Science
Project file

Prepared By :
Adeeb S Rahman
Nazia Niha Kaesh
Subham Sharma

Github Repository

Table of Contents

Melodia

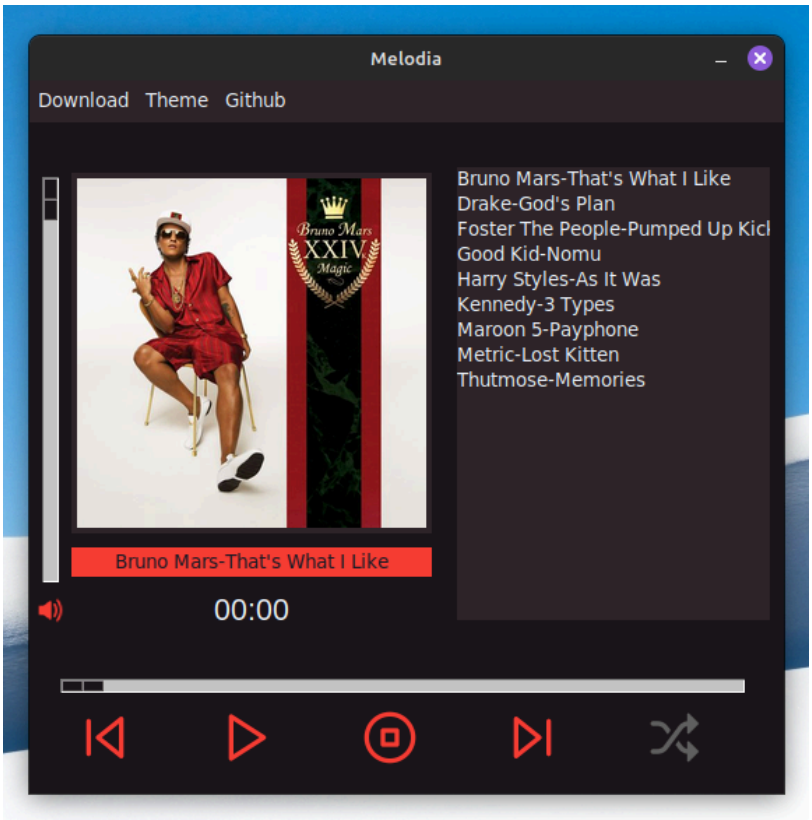
1: Introduction

1.1 : Abstract

Melodia is a python based, feature-rich music playing application that effectively serves as an offline mp3 player, providing users with a wide range of functionalities. It uses the Deezer API and youtube to provide users a way to listen to their favorite songs offline, by downloading the song on the device.It has a easy-to-use, modern UI.

The main features of the application include::

- Provide all standard issue features of a general MP3 player (pause, play, skip, autoplay, etc.)
- Ability to search and save music locally.
- Play, Create and Edit Playlists
- Interact with the app through a modern
- Various themes for user customization



Caption : Preview of Main Screen with “Magma” theme

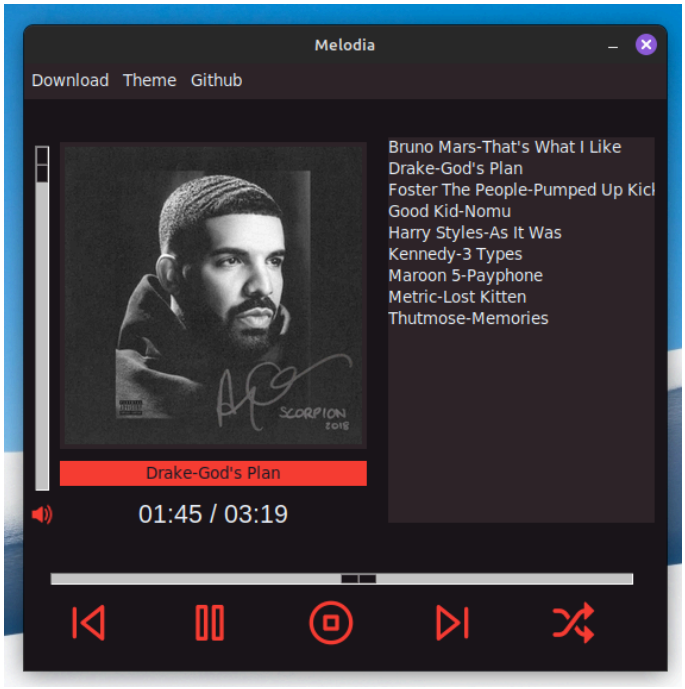
2 : Front End

The program is fully based on python's built-in Tkinter module. This was chosen due to ease of development, light-weight functionality, and user friendliness.

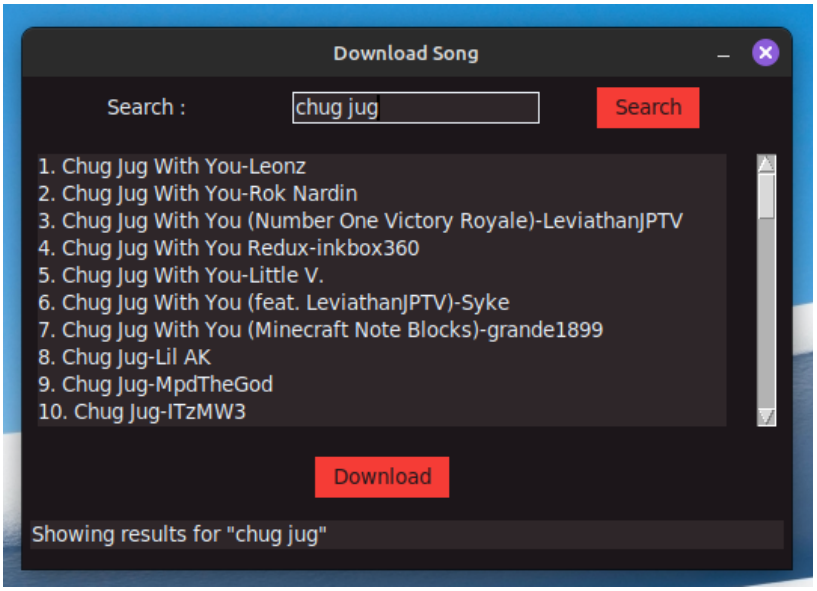
2.1 : Screens

The program consists of two screens:

1) Main Screen : Where users can see their downloaded songs and play them on their device,



2) Download Screen : Where users can search for more songs to download to their device.



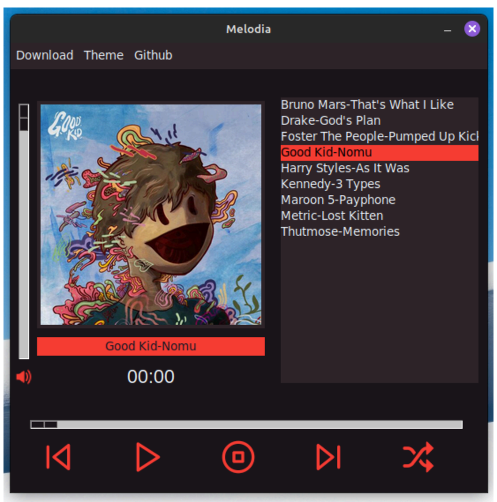
2.2 : Themes

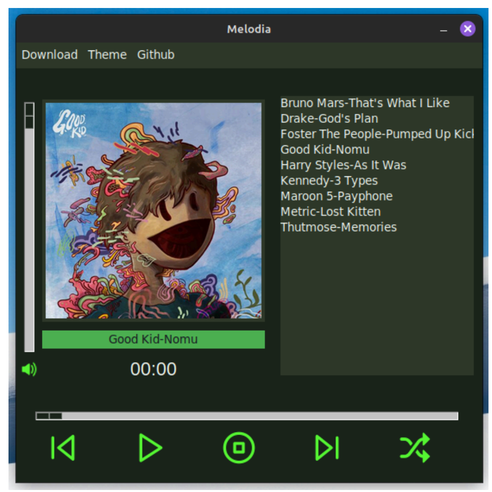
The application has various themes which add a sense of customization and personalization to the user experience. They can be changed dynamically. The themes include:

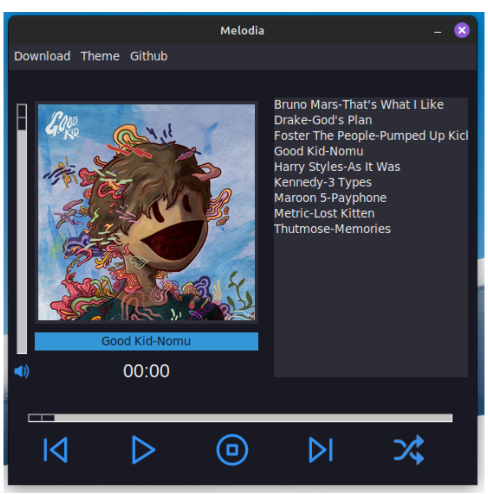
- Magma : Red coloured theme (Top-Left)
- Lush : Green coloured theme (Top-Right)
- Moonlit : Blue coloured theme (Bottom-Left)
- Nebula : Purple coloured theme (Bottom-Right)

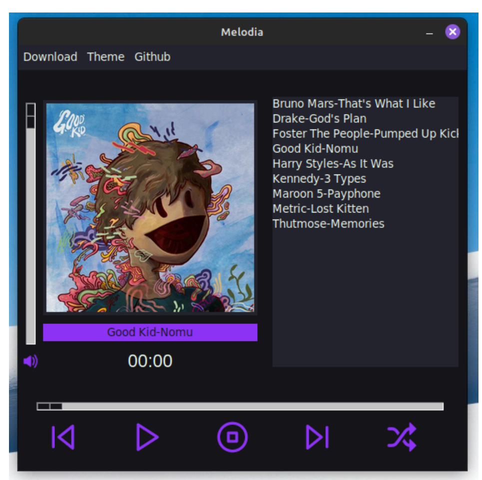
(These themes also carry over to the download page)

The themes are shown below:









3 : Back End

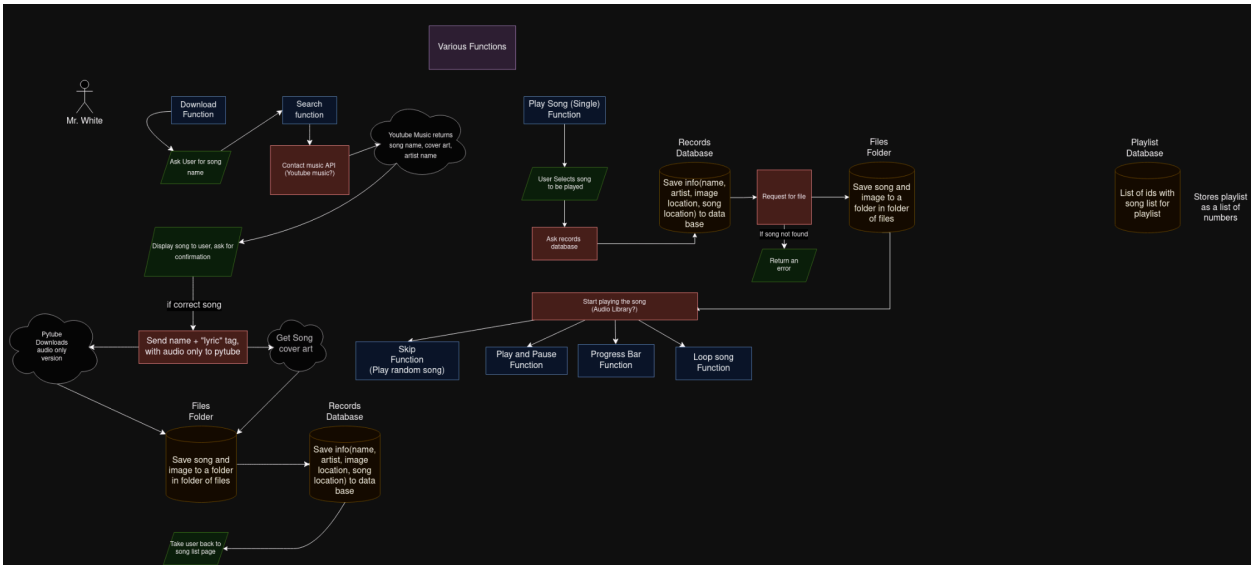
3.1 : Flow of Application

The following is a breakdown of how our back end works :

- **Overview of Scripts:**
  - The project consists of two Python scripts: `main.py` and `download.py`.
  - `main.py` is responsible for the graphical user interface (GUI) using Tkinter.
  - `download.py` manages the download and conversion of music tracks.
- **API Integration:**
  - The Deezer API is used to fetch information about artists and tracks.
- **User Interaction:**
  - Users interact with a Tkinter GUI that prompts them to search for an artist.
- **Download Process:**
  - `download.py` takes care of downloading the chosen track from YouTube.
  - It then converts the downloaded MP4 file to MP3 using the `moviepy` library.
- **File Organization:**
  - The resulting MP3 file is stored in the "./music/" directory.
  - The script also downloads the associated album cover, placing it in "./music/albumCover/".
- **User Instructions:**
  - To use the application, users run `main.py` and provide their Deezer API credentials.
  - The GUI allows users to search for artists, select tracks, and initiate the download process.

3.2 : Graphical Flowchart

The following is a flowchart to explain the functionalities of the app.



3.3 : Libraries Used

Modules from the following libraries are used. Information is provided with each to explain its function.

tkinter:	This module is used for creating graphical user interfaces (GUIs) of the main window, buttons, labels, and other GUI elements.
pygame:	This module is used for handling audio playback.
PIL(Python Imaging Library, now known as Pillow):	This module is used for handling images. In this code, it's used for resizing and displaying album cover images.
os:	This module provides a way to interact with the operating system. It's used for file and directory operations, such as checking if a file or directory exists, opening a folder, deleting files, etc.
time:	This module provides various time-related functions. In this code, it's used for formatting time durations.
tkinter.ttk:	This submodule of tkinter provides a themed widget set used for styling the ttk.Scale widget (slider).
mutagen.mp3:	This module is used for reading metadata from MP3 files to get information about the length of a song.
tkinter.messagebox:	This submodule of tkinter is used for displaying message boxes.
download:	It contains functions related to downloading songs.
webbrowser:	This module is used to open a GitHub page.
sys:	This module is used to determine the platform (OS) the program is running on.
random:	This module is used for selecting a random song when in shuffle mode.
deezer:	This module is used to access information about artists, albums, tracks, and playlists on Deezer using deezer API.
json:	This module is used to store song lists and encode JSON format (serialization) and decoding JSON data into Python objects (deserialization).
requests:	This module is used for accessing deezer API and fetching

	URLs.
pytube and pytube.search:	This module is used to find song on youtube to be downloaded and downloads it
moviepy.editor	This module is part of the MoviePy library, which is used for converting the downloaded mp4 file into a mp3 file only
shutil:	This module is used to make folders for songs, etc.
urllib:	This module is used to get images from url.
dotenv:	It is used to store API information

Note : Deezer API credentials are securely stored in a `.env` file. To use the application, users run `main.py` and provide their Deezer API credentials.

3.4 : File Structure

The various files and directories are explained below :

./Config	color.txt is used for COLOR of slider in GUI
./Music	Contains the mp3 files of the songs downloaded by the user
./Music/albumcover	It is a subfolder of music containing all the album cover images of downloaded songs.
./Sources	Contains images of the app logo, button icons, placeholder images, etc.
./Sources/ctrlbtn	Contains image of all the buttons used in the GUI
./Archive	Contains “Legacy Versions” of code for retrospective
./Reference	This folder contains various files which are used as reference when coding, such as flow chart, font list, etc
.env_sample	A sample file. This directs the user on how the .env file should look like.

Requirements.txt	All necessary libraries to run the application. This has been put here for ease of installation.

3.5 : Main Functions

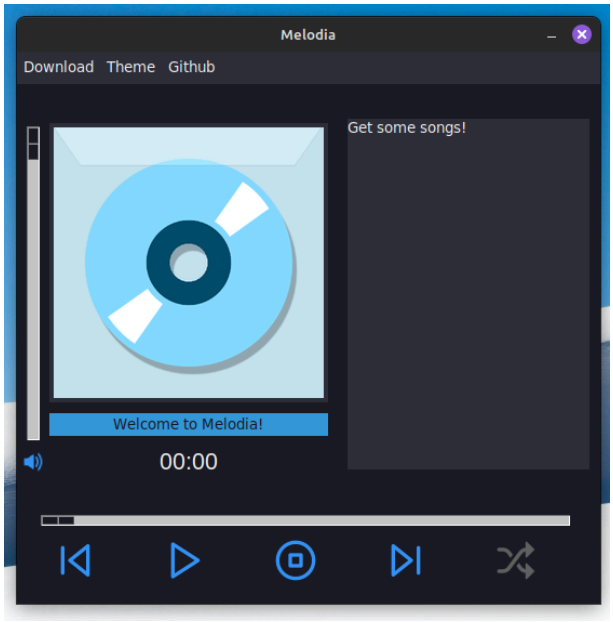
Following is a table explaining the functions used in the code.

master():	updates the song duration on the GUI, handles the play state, controls shuffle and autoplay.
songLengthGrabber():	Retrieves the length of the currently playing song using the Mutagen library.
changeName():	Updates the displayed name of the currently playing track in the GUI.
stop():	Stops the currently playing song, cancels scheduled updates, and resets the play state.
shuffleBtnFunc():	Toggles the shuffle state and updates the shuffle button's image accordingly.
randSelect():	Selects a random track from the listbox, excluding the currently playing track, the one before, and the one after.
mainBtnFunc(mainQuery):	Controls the play/pause functionality of the main button.
nextTrack(move):	Plays the next or previous track based on the movement direction.(-1 or 1 respectively)

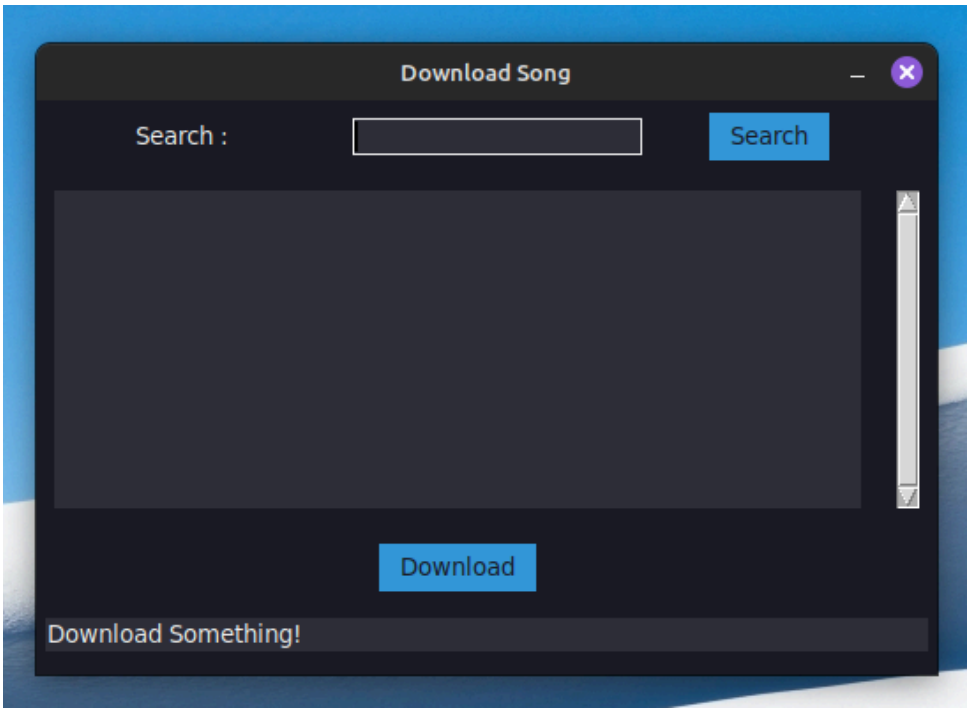
changeCover(trackNum):	Changes the album cover image based on the selected track.
delSong():	Stops the currently playing song, deletes the selected song and its cover, updates the GUI, and moves to the next track.
getSongName(path):	Converts a file path to a readable song name.
getSongPath(name):	Converts a song name to a file path.
getSongCov(name):	Converts a song name to an album cover image file path.
slide(pos):	Controls the slider for song duration and seeks the song to the specified position.
openFolder():	Opens the music folder.
changeColor(scheme):	Changes the color scheme of the GUI based on the selected theme.
reloadTracks():	: Reloads the list of tracks in the GUI.
volSliderFunc(x):	Controls the volume slider functionality and sets the volume for the Pygame mixer.
openGithub():	Opens the GitHub page of the project in the user's default browser.

4 : Screenshots

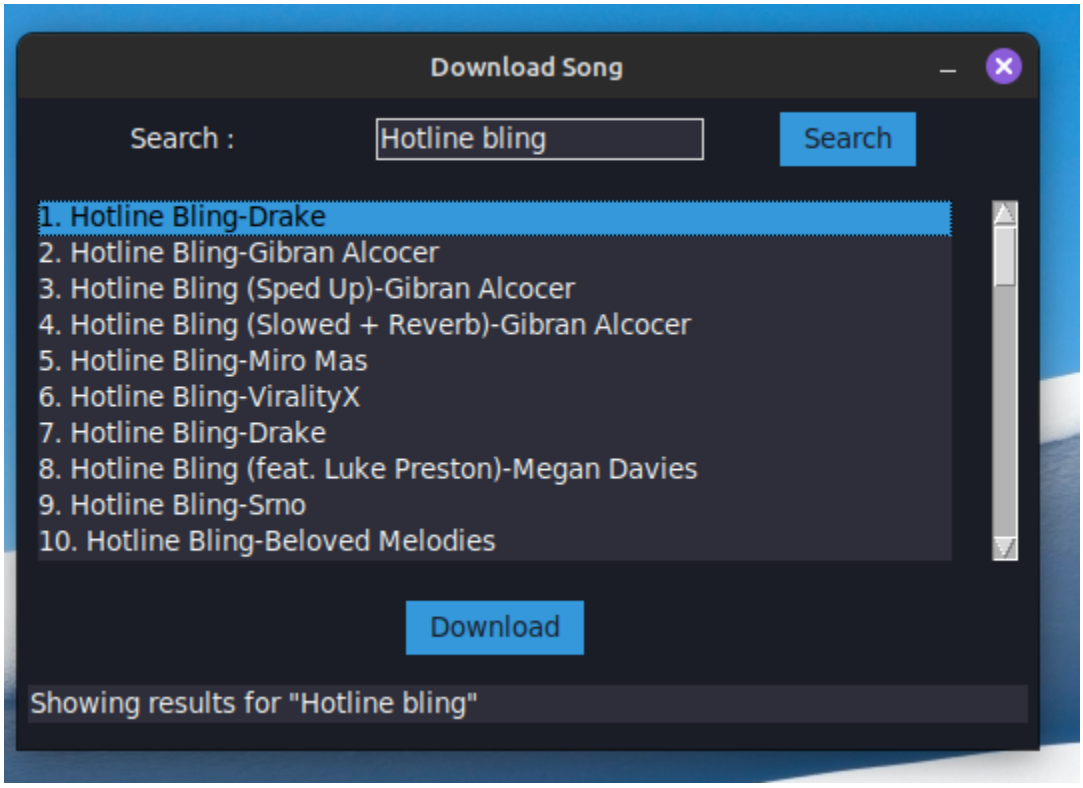
1. The main screen upon installation :



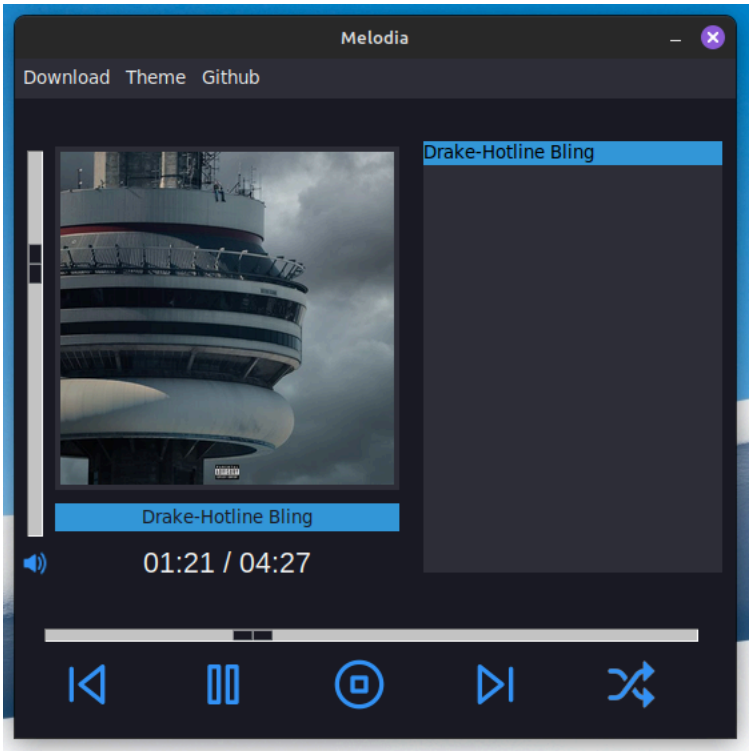
2. Let's download a song!
   Download > Get Song



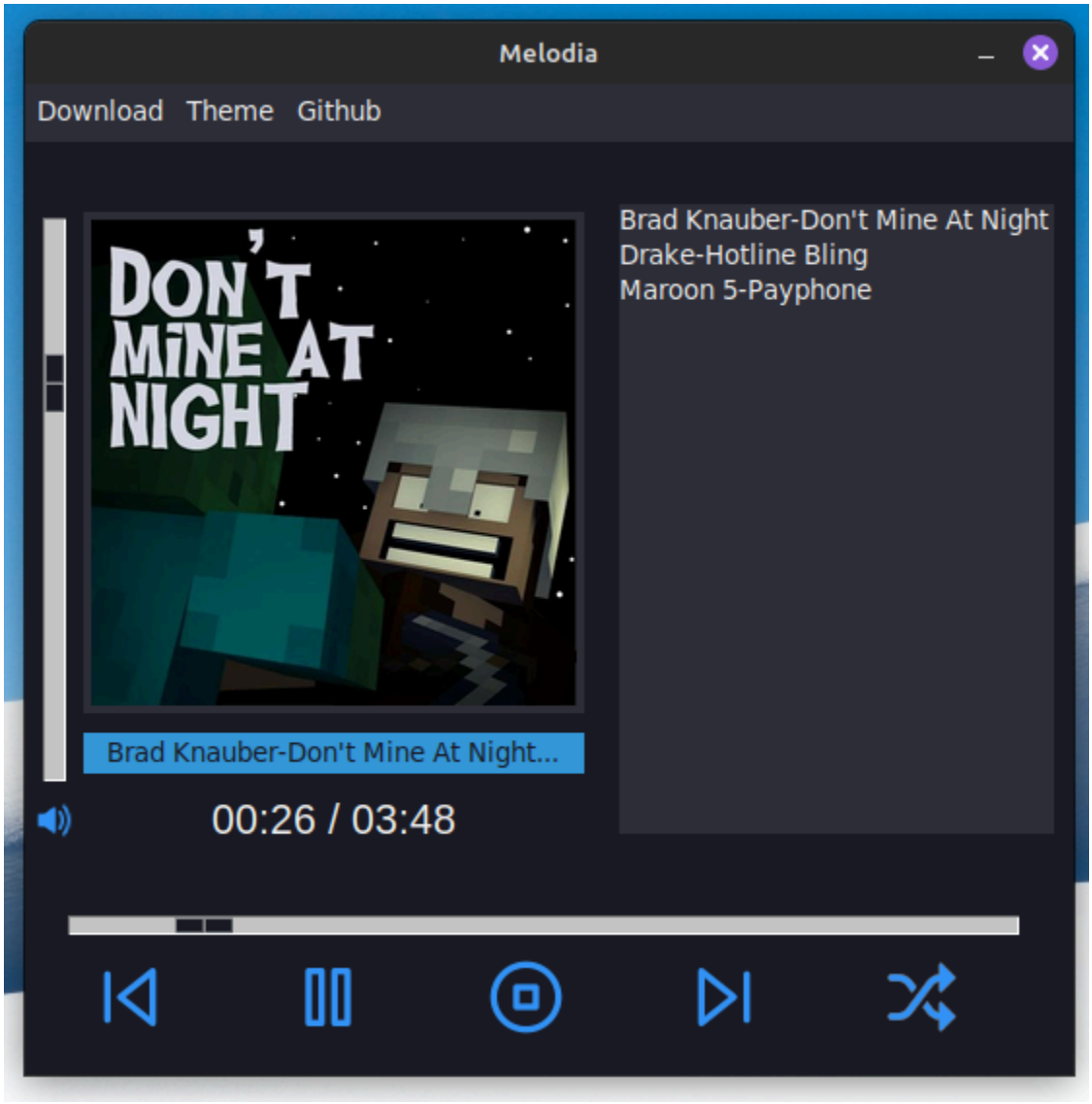
3. Searching and Downloading



4. Now we can play it!



5. We can take advantage of the shuffle play/ Autoplay if we have many songs



5 : Conclusion

5.1 : Conclusion

5.2 : Achievements

Through this project we learned how to use a tkinter based GUI with a comprehensive backend suite. We were able to learn about various python libraries to manage our backend, such as pygame Mixer, Mutagen, Shutil, movie py etc. We also learned how to contact an API (Deezer) which enabled us to get the data.

5.3 : Improvements

Various features can be added to enhance the user experience. These include:

- Change to a more reliable music search API key
- Create database and store more information about the Tracks
- Implement Playlists
- Add option to search both tracks or album
- Connect music GUI to Download feature
- Add secret features
- Autoplay, Themes, Loops

We plan to add these in the upcoming release.

5.4 : References

https://www.youtube.com/watch?v=djDcVWbEYoE&list=PLXLYwvNGGPoUzjKiGvuXm8qeiOYMnFria
-Used for GUI of the project and as a reference for how to play music using pygame.

https://anzeljg.github.io/rin2/book2/2405/docs/tkinter/index.html
- used for General tutorials on tkinter and resolving many queries in the project.

https://developers.deezer.com/
- API used in a project.

https://deezer-python.readthedocs.io/en/stable/
-Deezer API Wrapper

https://coolors.co/
Color palette

https://www.flaticon.com/free-icons/mp3-player
Used for icons in GUI

https://icons8.com/
Also used for icons in GUI.

5.5 : Contributors

- Adeeb S Rahman
- Nazia Niha Kaesh
- Subham Sharma

The full Github repository can be found in the following link:

https://github.com/Arctican4Real/Melodia

Github Repository