# Annotation for user

SoundIsland is a small music streaming service with huge discography. Using this app, you can find and listen to any song by It's name or artist(**Doesn't support songs that has Cyrillic and French symbols in names**).

Most common problems:

1. If SoundIsland doesn’t download songs check stability of your internet connection.
2. After you make your query, please give some time for app to download all the songs found by your query.
3. If the application does not download songs and the previous points did not help, please email me.
4. If you entered name song that has Cyrillic and French symbols in name, you need to rerun app.

# For programmers

## 1. C# part of project:

Note: This part is responsible for the body of app, design, connection with API and all the features app has. Also, bin directory is necessary for this project, because it contains downloadSong.py, which we use to download songs and all the links needed for this file to work.

**1. Libraries used in C# part**: Newtonsoft, WMPLib.(All the libraries of this project are stored in bin\debug\net6.0-windows, with .dll extension)

**2. API used in project:** Deezer API.

Note: Due to the fact that I use WinForms for this app some of the functions are connected to buttons and labels through properties.

## 3. Menu screen

## Variables:

1. isClickedOnSearchingBar - Bool variable that indicates that user clicked on searching bar.

2. oldSize - Variable used to remember previous size of the window.

3. Path - Variable that stores path to python.exe.

## Functions:

a. public void CheckPythonEnable – Function used to find python.exe path in register, if it exists. It gets the nothing and returns nothing.
b. public string FindVersion - Function used to find python version in register. It gets the keyDirectory(Key directory in windows register) and returns string version(Version of python downloaded on user's computer), if it exists.
c. private async void Search_Click - Function connected to button search on Menu frame. It gets the obj sender and EventArg e and in result of working sends text from searching bar to next frame and opens SearchingResultPage.
d. private void SearchingBar_Click - Function connected to searching bar and used to change style of searching bar when

user clicks on it. It gets the obj sender and EventArg e and returns nothing.

e. private void Menu_Click - Function connected to frame and used to change style of searching bar and delete text from searching bar if user doesn't use it. It gets the obj sender and EventArg e and returns nothing.
f. private void Menu_FormClosing - Function conected to frame and used to kill the proces of app and clean cache after app is closed. It gets the obj sender and EventArg e and returns nothing.
g. private void ResizeAll - Function used to resize all objects and buttons on window. It gets the 2 args control(used to resize objects) and newSize(used to calculate new sizes of objects).
h. private void SearchingBar_TextChanged - Function connected to searching bar and used to change param Enabled on text. It gets the obj sender and EventArg e and returns nothing.

## 4. SearchingResultPage

## Variables:

1. pauseButton - Variable that stores pause button image.

2. continueButton - Variable that stores continue button image.

3. buttons - List that stores buttons that used as player background.

4. controlButtons - List that stores control buttons.

5. player – Variable that stores music player.

6. tracks - List that stores links of songs.

7. MatchID - Dictionary that stores as key buttons and as value song links.

8. matchIDControls - Dictionary that stores as key control buttons and as value song links.

9. numberOfPlayers - Constant variable that stores max number of players.

## Functions:

i. public async void MakeRequest - Function used to make request to Deezer API, get the data and store it. It gets the text from searching bar on previous frame and in result makes call to another function.

j. private void getSong - Function used to rewrite all the links to Song link.txt file that stores all the song links. It gets the nothing and in result calls another function run_cmd and checkSongEnable.

k. private void run_cmd - Function used to run Python file that used to download all the tracks using their links. It gets the param cmd(Name of the Python file) and returns nothing.

l. private void checkSongEnable - Function used to check if song mp3 file is downloaded. We compare name of directory(As names we use tracks ID) with track ID to check

attendance of song. It gets the nothing and in result makes call to function ManagePlayer.

m. public async void ManagePlayer – Function used to set design and functional of player. It gets the 3 params track(list that stores song ID's) button(list that stores buttons) and controlButton(list that stores control buttons).
n. private void changeVolume - Function connected to volume bar and used to change volume of track. It gets the obj sender and EventArg e and returns nothing.
o. public async void CreateNewPlayer - Function used to generate all buttons used to make background of player and control buttons and set all the buttons sizes. It gets the nothing and returns nothing.
p. private void SerchingResultPage_SizeChanged - Function connected to frame and used to remember old size of window and in result method makes call to another method ResizeAll. It gets the obj sender and EventArg e and returns nothing.(Function isn’t finished).
q. private void ResizeAll - Function used to resize all objects and buttons on window. It gets the 2 args control(used to resize objects) and newSize(used to calculate new sizes of objects).(Function isn’t finished).
r. private void SerchingResultPage_FormClosin - Function connected to frame and used to kill the app's process when form is closing, stop player and clean all its data and clean all the cache. It gets the obj sender and EventArg e and returns nothing.

s. private void BackToMenu_Click - Function connected back to menu button and used to stop player and clean all its data and clean all the cache. and return to menu frame. It gets the obj sender and EventArg e and returns nothing.

## Classes and their methods:

## DeezerTrack class:

**Aim of this class** – This class used to store all the data we get from Deezer API.

### Properties of this class:

a. public string Title – propertie used to store song title.
b. public string ArtistName – propertie used to store the name of artist.
c. public string AlbumName – propertie used to store the name of album.
d. public string Cover – propertie used to store link with album cover(smallest size).
e. public string CoverSmall – propertie used to store link with album cover(small size).
f. public string CoverMid – propertie used to store link with album cover(medium size).
g. public string CoverXL – propertie used to store link with album cover(The biggest size).
h. public string Duration – propertie used to store duration of song.

i. public string PreviewUrl – propertie used to store preview link of the song(30 sec. snippet).
j. public string DeezerUrl – propertie used to store song link that we use for download.
k. public string ToString – Method used to convert all the data to string.

# 2. Python part of project:

Note: Python part only responsible for download of song by link. Path of python file that we use to download songs - bin\debug\net6.0-windows\downloadSong.py

## 1. Libraries:

Python: pydeezer. (If app doesn't work, check all the links from pydeezer library using downloadSong.py file).

## 2. Python functions(downloadSong.py):

a. def get_track_id – function used get track ID from link we get from Deezer API. It gets list_of_url(list that stores all the links) and returns string list id_ list (list with track ID's).
b. def download – function used to download all the songs in Song cache directory using track ID's we get from previous function. It gets id_list(list with track ID's) and returns nothing.

# 3. Other important files and directories:

**All those files and directories are in bin\debug\net6.0-windows directory. If you'll compile files in other directory, copy those files too.**

a. ARL.txt - file that stores constant ARL key used to log in to Deezer through pydeezer library(Do not change this file).
b. Song list.txt - file that stores links we get from Deezer API and then used by python file for download of songs.
c. Song Cache – Directory that stores .mp3 files that in result used by players(all files are cleared after switching back to the menu or closing the app)
d. Pause Button.png and Continue Button.png - images used by C# part of project as images for control buttons.
e. Also, this directory contains necessary libs(It can be directory or .py file) used by python part of project, for example: aiohttp, pydeezer, aiosignal … (If you see files with .py extension, because they are really important for this project)

## 4. Analogs of this app, API and libraries:

1. Deezer API – This API has analogs such as Spotify API and iTunes API, but they have way less abilities and they can only be used to store metadata(making playlists or statistical analysis) due to very strict politics of these streaming services.

2. Newtonsoft – This library has analogs such as System.Text.Json, Utf8Json and Jil, but Newtonsoft is very easy to use.

3. Pydeezer – This library has analogs such as Spotipy, but It’s used with another streaming service and as I said they have very strict politics towards projects that using theirs's data

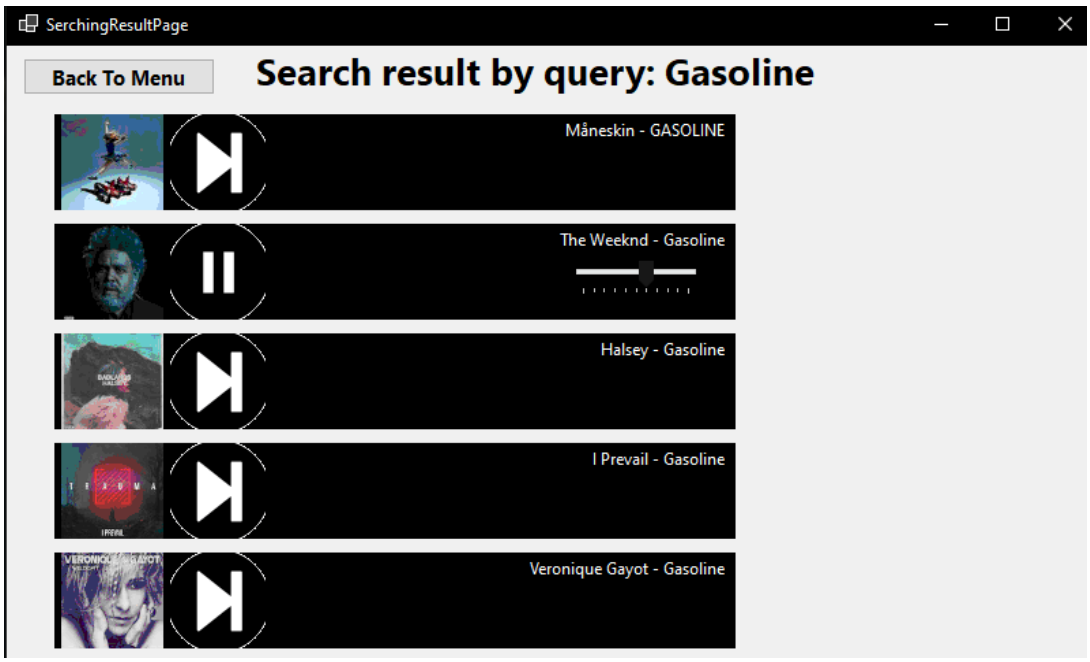
4. Off course my app has analogs that have bigger functional, but my app is easy to use, it has wide discography and It’s completely free.

## 5. Representation of received by App data

SoundIsland searches for songs using their names or names of artists, but not by album names or their lyrics. Also, this app doesn't support song that has Cyrillic and French symbols in names due to pydeezer limits.

## 6. Representation of output data by app

If you entered data correctly, wait until app download all the songs and then you can listen to your track using control buttons.



Pict. 1. Example of songs found by query

## 7. Unfinished parts

Unfortunately, it didn't succeed everything I expected in the very beginning.

1. Rewind of songs – right now this app can only turn on the song and user can't rewind the song during listening, and I want to add this feature in later versions of this program.
2. Better design – right now this app has very minimalistic and pretty decent design, which I want to improve in later versions and add more feature connected to customization of this app.
3. Support of Cyrillic languages – currently this app doesn't support this group of languages due to the limits of pydeezer library. In later versions I want to find a way to get around that limitation or find better analog of this library.

## 8. Conclusion

This bot has all the most necessary stuff for app of this type, such as wide discography, ability to find the song by the name or artist and ability to listen to full version of track in good quality. I personally think that I did a great job developing this app - I learned how to work and extract data from API, ways to run python project using C# and threading for better optimization.

## Few query examples for tests:

The Weeknd

Dani California

Ausländer