```
import json
import os
from os import path
from typing import Callable

import aiofiles
import aiohttp
import ffmpeg
import requests
import wget
from PIL import Image, ImageDraw, ImageFont
from pyrogram import Client, filters
from pyrogram.types import Voice
from pyrogram.errors import UserAlreadyParticipant
from pyrogram.types import InlineKeyboardButton,
InlineKeyboardMarkup, Message
from Python_ARQ import ARQ
from youtube_search import YoutubeSearch

from config import ARQ_API_KEY
from config import BOT_NAME as bn
from config import DURATION_LIMIT
```

```
from config import UPDATES_CHANNEL as updateschannel

from config import que

from function.admins import admins as a

from helpers.admins import get_administrators

from helpers.channelmusic import get_chat_id

from helpers.errors import DurationLimitError

from helpers.decorators import errors

from helpers.decorators import authorized_users_only

from helpers.filters import command, other_filters

from helpers.gets import get_file_name

from services.callsmusic import callsmusic

from services.callsmusic.callsmusic import client as USER

from services.converter.converter import convert

from services.downloaders import youtube

from services.queues import queues


aiohttpsession = aiohttp.ClientSession()

chat_id = None

arq = ARQ("https://thearq.tech", ARQ_API_KEY, aiohttpsession)

DISABLED_GROUPS = []

useer ="NaN"

def cb_admin_check(func: Callable) -> Callable:
```

```
    async def decorator(client, cb):
        admemes = a.get(cb.message.chat.id)
        if cb.from_user.id in admemes:
            return await func(client, cb)
        else:
            await cb.answer("You ain't allowed!",
show_alert=True)
            return

    return decorator


def transcode(filename):
    ffmpeg.input(filename).output(
        "input.raw", format="s16le", acodec="pcm_s16le",
ac=2, ar="48k"
    ).overwrite_output().run()
    os.remove(filename)


# Convert seconds to mm:ss
def convert_seconds(seconds):
    seconds = seconds % (24 * 3600)
    seconds %= 3600
    minutes = seconds // 60
```

```
    seconds %= 60
    return "%02d:%02d" % (minutes, seconds)



# Convert hh:mm:ss to seconds
def time_to_seconds(time):
    stringt = str(time)
    return sum(int(x) * 60 ** i for i, x in
enumerate(reversed(stringt.split(":"))))



# Change image size
def changeImageSize(maxWidth, maxHeight, image):
    widthRatio = maxWidth / image.size[0]
    heightRatio = maxHeight / image.size[1]
    newWidth = int(widthRatio * image.size[0])
    newHeight = int(heightRatio * image.size[1])
    newImage = image.resize((newWidth, newHeight))
    return newImage



async def generate_cover(requested_by, title, views,
duration, thumbnail):
    async with aiohttp.ClientSession() as session:
        async with session.get(thumbnail) as resp:
```

```
        if resp.status == 200:
            f = await aiofiles.open("background.png", mode="wb")
            await f.write(await resp.read())
            await f.close()


    image1 = Image.open("./background.png")
    image2 = Image.open("./etc/foreground.png")
    image3 = changeImageSize(1280, 720, image1)
    image4 = changeImageSize(1280, 720, image2)
    image5 = image3.convert("RGBA")
    image6 = image4.convert("RGBA")
    Image.alpha_composite(image5, image6).save("temp.png")
    img = Image.open("temp.png")
    draw = ImageDraw.Draw(img)
    font = ImageFont.truetype("etc/font.otf", 32)
    draw.text((205, 550), f"🎙 Title: {title}", (51, 215, 255), font=font)
    draw.text((205, 590), f"⏱ Duration: {duration}", (255, 255, 255), font=font)
    draw.text((205, 630), f"👁 Views: {views}", (255, 255, 255), font=font)
    draw.text(
        (205, 670),
        f"Added By: {requested_by}",
```

```
        (255, 255, 255),
        font=font,
    )
    img.save("final.png")
    os.remove("temp.png")
    os.remove("background.png")



@Client.on_message(filters.command("playlist") &
filters.group & ~filters.edited)
async def playlist(client, message):
    global que
    if message.chat.id in DISABLED_GROUPS:
        return
    queue = que.get(message.chat.id)
    if not queue:
        await message.reply_text("Player is idle")
    temp = []
    for t in queue:
        temp.append(t)
    now_playing = temp[0][0]
    by = temp[0][1].mention(style="md")
    msg = "**Now Playing** in
{}".format(message.chat.title)
    msg += "\n- " + now_playing
```

```
    msg += "\n- Req by " + by
    temp.pop(0)
    if temp:
        msg += "\n\n"
        msg += "**Queue**"
        for song in temp:
            name = song[0]
            usr = song[1].mention(style="md")
            msg += f"\n- {name}"
            msg += f"\n- Req by {usr}\n"
    await message.reply_text(msg)



# ============================= Settings
========================================



def updated_stats(chat, queue, vol=100):
    if chat.id in callsmusic.pytgcalls.active_calls:
        # if chat.id in active_chats:
        stats = "Settings of **{}**".format(chat.title)
        if len(que) > 0:
            stats += "\n\n"
            stats += "Volume : {}%\n".format(vol)
```

```
        stats += "Songs in queue : `{}`\n".format(len(que))
        stats += "Now Playing : **{}**\n".format(queue[0][0])
        stats += "Requested by : {}".format(queue[0][1].mention)
    else:
        stats = None
    return stats



def r_ply(type_):
    if type_ == "play":
        pass
    else:
        pass
    mar = InlineKeyboardMarkup(
        [
            [
                InlineKeyboardButton("⏹", "leave"),
                InlineKeyboardButton("⏸", "puse"),
                InlineKeyboardButton("▶️", "resume"),
                InlineKeyboardButton("⏭", "skip"),
            ],
            [
```

```
            InlineKeyboardButton("Playlist 📖",
"playlist"),
        ],
        [InlineKeyboardButton("❌ Close", "cls")],
    ]
)
return mar
```

```
@Client.on_message(filters.command("current") &
filters.group & ~filters.edited)
async def ee(client, message):
    if message.chat.id in DISABLED_GROUPS:
        return
    queue = que.get(message.chat.id)
    stats = updated_stats(message.chat, queue)
    if stats:
        await message.reply(stats)
    else:
        await message.reply("No VC instances running in
this chat")
```

```
@Client.on_message(filters.command("player") &
filters.group & ~filters.edited)
@authorized_users_only
```

```
async def settings(client, message):
    if message.chat.id in DISABLED_GROUPS:
        await message.reply("Music Player is Disabled")
        return
    playing = None
    chat_id = get_chat_id(message.chat)
    if chat_id in callsmusic.pytgcalls.active_calls:
        playing = True
    queue = que.get(chat_id)
    stats = updated_stats(message.chat, queue)
    if stats:
        if playing:
            await message.reply(stats,
reply_markup=r_ply("pause"))


        else:
            await message.reply(stats,
reply_markup=r_ply("play"))
    else:
        await message.reply("No VC instances running in
this chat")



@Client.on_message(
    filters.command("musicplayer") & ~filters.edited &
~filters.bot & ~filters.private
```

```
)
@authorized_users_only
async def hfmm(_, message):
    global DISABLED_GROUPS
    try:
        user_id = message.from_user.id
    except:
        return
    if len(message.command) != 2:
        await message.reply_text(
            "I only recognize `/musicplayer on` and /musicplayer `off only`"
        )
        return
    status = message.text.split(None, 1)[1]
    message.chat.id
    if status == "ON" or status == "on" or status == "On":
        lel = await message.reply("`Processing...`")
        if not message.chat.id in DISABLED_GROUPS:
            await lel.edit("Music Player Already Activated In This Chat")
            return
        DISABLED_GROUPS.remove(message.chat.id)
        await lel.edit(
```

```
            f"Music Player Successfully Enabled For Users
In The Chat {message.chat.id}"
        )


    elif status == "OFF" or status == "off" or status ==
"Off":
        lel = await message.reply("`Processing...`")


        if message.chat.id in DISABLED_GROUPS:
            await lel.edit("Music Player Already turned off
In This Chat")
            return
        DISABLED_GROUPS.append(message.chat.id)
        await lel.edit(
            f"Music Player Successfully Deactivated For
Users In The Chat {message.chat.id}"
        )
    else:
        await message.reply_text(
            "I only recognize `/musicplayer on` and
/musicplayer `off only`"
        )



@Client.on_callback_query(filters.regex(pattern=r"^(p
laylist)$"))
```

```
async def p_cb(b, cb):
    global que
    que.get(cb.message.chat.id)
    type_ = cb.matches[0].group(1)
    cb.message.chat.id
    cb.message.chat
    cb.message.reply_markup.inline_keyboard[1]
[0].callback_data
    if type_ == "playlist":
        queue = que.get(cb.message.chat.id)
        if not queue:
            await cb.message.edit("Player is idle")
        temp = []
        for t in queue:
            temp.append(t)
        now_playing = temp[0][0]
        by = temp[0][1].mention(style="md")
        msg = "**Now Playing** in
{}".format(cb.message.chat.title)
        msg += "\n- " + now_playing
        msg += "\n- Req by " + by
        temp.pop(0)
        if temp:
            msg += "\n\n"
            msg += "**Queue**"
```

```
            for song in temp:
                name = song[0]
                usr = song[1].mention(style="md")
                msg += f"\n- {name}"
                msg += f"\n- Req by {usr}\n"
        await cb.message.edit(msg)



@Client.on_callback_query(

filters.regex(pattern=r"^(play|pause|skip|leave|puse
|resume|menu|cls)$")
)
@cb_admin_check
async def m_cb(b, cb):
    global que
    if (
        cb.message.chat.title.startswith("Channel Music:
")
        and chat.title[14:].isnumeric()
    ):
        chet_id = int(chat.title[13:])
    else:
        chet_id = cb.message.chat.id
    qeue = que.get(chet_id)
```

```
    type_ = cb.matches[0].group(1)
    cb.message.chat.id
    m_chat = cb.message.chat


    the_data =
cb.message.reply_markup.inline_keyboard[1]
[0].callback_data
    if type_ == "pause":
        if (chet_id not in callsmusic.pytgcalls.active_calls)
or (
            callsmusic.pytgcalls.active_calls[chet_id] ==
"paused"
        ):
            await cb.answer("Chat is not connected!",
show_alert=True)
        else:
            callsmusic.pytgcalls.pause_stream(chet_id)


            await cb.answer("Music Paused!")
            await cb.message.edit(
                updated_stats(m_chat, qeue),
reply_markup=r_ply("play")
            )


    elif type_ == "play":
        if (chet_id not in callsmusic.pytgcalls.active_calls)
or (
```

```
            callsmusic.pytgcalls.active_calls[chet_id] ==
"playing"
        ):
            await cb.answer("Chat is not connected!",
show_alert=True)
        else:
            callsmusic.pytgcalls.resume_stream(chet_id)
            await cb.answer("Music Resumed!")
            await cb.message.edit(
                updated_stats(m_chat, qeue),
reply_markup=r_ply("pause")
            )


    elif type_ == "playlist":
        queue = que.get(cb.message.chat.id)
        if not queue:
            await cb.message.edit("Player is idle")
        temp = []
        for t in queue:
            temp.append(t)
        now_playing = temp[0][0]
        by = temp[0][1].mention(style="md")
        msg = "**Now Playing** in
{}".format(cb.message.chat.title)
        msg += "\n- " + now_playing
        msg += "\n- Req by " + by
```

```
        temp.pop(0)
        if temp:
            msg += "\n\n"
            msg += "**Queue**"
            for song in temp:
                name = song[0]
                usr = song[1].mention(style="md")
                msg += f"\n- {name}"
                msg += f"\n- Req by {usr}\n"
        await cb.message.edit(msg)


    elif type_ == "resume":
        if (chet_id not in callsmusic.pytgcalls.active_calls)
or (
            callsmusic.pytgcalls.active_calls[chet_id] ==
"playing"
        ):
            await cb.answer("Chat is not connected or
already playng", show_alert=True)
        else:
            callsmusic.pytgcalls.resume_stream(chet_id)
            await cb.answer("Music Resumed!")
    elif type_ == "puse":
        if (chet_id not in callsmusic.pytgcalls.active_calls)
or (
```

```
            callsmusic.pytgcalls.active_calls[chet_id] ==
"paused"
        ):
            await cb.answer("Chat is not connected or
already paused", show_alert=True)
        else:
            callsmusic.pytgcalls.pause_stream(chet_id)


            await cb.answer("Music Paused!")
    elif type_ == "cls":
        await cb.answer("Closed menu")
        await cb.message.delete()


    elif type_ == "menu":
        stats = updated_stats(cb.message.chat, qeue)
        await cb.answer("Menu opened")
        marr = InlineKeyboardMarkup(
            [
                [
                    InlineKeyboardButton("⏹", "leave"),
                    InlineKeyboardButton("⏸", "puse"),
                    InlineKeyboardButton("▶️", "resume"),
                    InlineKeyboardButton("⏭", "skip"),
                ],
                [
```

```
                InlineKeyboardButton("Playlist 📖",
"playlist"),
            ],
            [InlineKeyboardButton("❌ Close", "cls")],
        ]
    )
    await cb.message.edit(stats, reply_markup=marr)
  elif type_ == "skip":
    if qeue:
        qeue.pop(0)
    if chet_id not in callsmusic.pytgcalls.active_calls:
        await cb.answer("Chat is not connected!",
show_alert=True)
    else:
        callsmusic.queues.task_done(chet_id)

        if callsmusic.queues.is_empty(chet_id):

callsmusic.pytgcalls.leave_group_call(chet_id)

            await cb.message.edit("- No More Playlist..\n-
Leaving VC!")
        else:
            callsmusic.pytgcalls.change_stream(
                chet_id, callsmusic.queues.get(chet_id)
["file"]
```

```
            )
            await cb.answer("Skipped")
            await cb.message.edit((m_chat, qeue),
reply_markup=r_ply(the_data))
            await cb.message.reply_text(
                f"- Skipped track\n- Now Playing **
{qeue[0][0]}**"
            )


    else:
        if chet_id in callsmusic.pytgcalls.active_calls:
            try:
                callsmusic.queues.clear(chet_id)
            except QueueEmpty:
                pass


            callsmusic.pytgcalls.leave_group_call(chet_id)
            await cb.message.edit("Successfully Left the
Chat!")
        else:
            await cb.answer("Chat is not connected!",
show_alert=True)



@Client.on_message(command("play") & other_filters)
async def play(_, message: Message):
```

```
    global que
    global useer
    if message.chat.id in DISABLED_GROUPS:
        return
    lel = await message.reply("🔄 **Processing**")
    administrators = await
get_administrators(message.chat)
    chid = message.chat.id


    try:
        user = await USER.get_me()
    except:
        user.first_name = "helper"
    usar = user
    wew = usar.id
    try:
        # chatdetails = await USER.get_chat(chid)
        await _.get_chat_member(chid, wew)
    except:
        for administrator in administrators:
            if administrator == message.from_user.id:
                if message.chat.title.startswith("Channel
Music: "):
                    await lel.edit(
```

```
                "<b>Remember to add helper to your
channel</b>",
            )
            pass
        try:
            invitelink = await
_.export_chat_invite_link(chid)
        except:
            await lel.edit(
                "<b>Add me as admin of yor group
first</b>",
            )
            return

        try:
            await USER.join_chat(invitelink)
            await USER.send_message(
                message.chat.id, "I joined this group for
playing music in VC"
            )
            await lel.edit(
                "<b>helper userbot joined your
chat</b>",
            )

        except UserAlreadyParticipant:
```

```
                pass
            except Exception:
                # print(e)
                await lel.edit(
                    f"<b>🔴 Flood Wait Error 🔴 \nUser {user.first_name} couldn't join your group due to heavy requests for userbot! Make sure user is not banned in group."
                    "\n\nOr manually add assistant to your Group and try again</b>",
                )
    try:
        await USER.get_chat(chid)
        # lmoa = await client.get_chat_member(chid,wew)
    except:
        await lel.edit(
            f"<i> {user.first_name} Userbot not in this chat, Ask admin to send /play command for first time or add {user.first_name} manually</i>"
        )
        return
    text_links=None
    await lel.edit("🔎 **Finding**")
    if message.reply_to_message:
        entities = []
```

```
        toxt = message.reply_to_message.text or
message.reply_to_message.caption
        if message.reply_to_message.entities:
            entities = message.reply_to_message.entities +
entities
        elif message.reply_to_message.caption_entities:
            entities = message.reply_to_message.entities +
entities
        urls = [entity for entity in entities if entity.type ==
'url']
        text_links = [
            entity for entity in entities if entity.type ==
'text_link'
        ]
    else:
        urls=None
    if text_links:
        urls = True
    user_id = message.from_user.id
    user_name = message.from_user.first_name
    rpk = "[" + user_name + "](tg://user?id=" +
str(user_id) + ")"
    audio = (
        (message.reply_to_message.audio or
message.reply_to_message.voice)
        if message.reply_to_message
        else None
```

```
    )
    if audio:
        if round(audio.duration / 60) > DURATION_LIMIT:
            raise DurationLimitError(
                f"❌ Videos longer than {DURATION_LIMIT} minute(s) aren't allowed to play!"
            )
        keyboard = InlineKeyboardMarkup(
            [
                [
                    InlineKeyboardButton("📖 Playlist", callback_data="playlist"),
                    InlineKeyboardButton("Menu ⏯ ", callback_data="menu"),
                ],
                [InlineKeyboardButton(text="❌ Close", callback_data="cls")],
            ]
        )
        file_name = get_file_name(audio)
        title = file_name
        thumb_name = "https://telegra.ph/file/f6086f8909fbfeb0844f2.png"
        thumbnail = thumb_name
        duration = round(audio.duration / 60)
        views = "Locally added"
```

```
        requested_by = message.from_user.first_name
        await generate_cover(requested_by, title, views, duration, thumbnail)
        file_path = await convert(
            (await message.reply_to_message.download(file_name))
            if not path.isfile(path.join("downloads", file_name))
            else file_name
        )
    elif urls:
        query = toxt
        await lel.edit("🎵 **Processing**")
        ydl_opts = {"format": "bestaudio[ext=m4a]"}
        try:
            results = YoutubeSearch(query, max_results=1).to_dict()
            url = f"https://youtube.com{results[0]['url_suffix']}"
            # print(results)
            title = results[0]["title"][:40]
            thumbnail = results[0]["thumbnails"][0]
            thumb_name = f"thumb{title}.jpg"
            thumb = requests.get(thumbnail, allow_redirects=True)
            open(thumb_name, "wb").write(thumb.content)
```

```
        duration = results[0]["duration"]
        results[0]["url_suffix"]
        views = results[0]["views"]

    except Exception as e:
        await lel.edit(
            "Song not found.Try another song or maybe spell it properly."
        )
        print(str(e))
        return
    dlurl=url
    dlurl=dlurl.replace("youtube","youtubepp")
    keyboard = InlineKeyboardMarkup(
        [
            [
                InlineKeyboardButton(text="🎬 YouTube", url=f"{url}"),
                InlineKeyboardButton(text="Download 📥", url=f"{dlurl}"),
            ],
            [InlineKeyboardButton(text="❌ Close", callback_data="cls")],
        ]
    )
    requested_by = message.from_user.first_name
```

```
        await generate_cover(requested_by, title, views,
duration, thumbnail)

        file_path = await
convert(youtube.download(url))

    else:

        query = ""

        for i in message.command[1:]:

            query += " " + str(i)

        print(query)

        await lel.edit("🎵 **Processing**")

        ydl_opts = {"format": "bestaudio[ext=m4a]"}


        try:

          results = YoutubeSearch(query,
max_results=5).to_dict()

        except:

          await lel.edit("Give me something to play")

        # Looks like hell. Aren't it?? FUCK OFF

        try:

            toxxt = "**Select the song you want to
play**\n\n"

            j = 0

            useer=user_name

            emojilist = ["1⃣","2⃣","3⃣","4⃣","5⃣",]


            while j < 5:
```

```
            toxxt += f"{emojilist[j]} **🎙️ Title - [{results[j]['title']}](https://youtube.com{results[j]['url_suffix']})**\n"

            toxxt += f"╚**⏱️ Duration** - {results[j]['duration']}\n"

            toxxt += f"╚**👁‍🗨 Views** - {results[j]['views']}\n"

            toxxt += f"╚**🔔 Channel** - {results[j]['channel']}\n\n"


            j += 1

        koyboard = InlineKeyboardMarkup(

            [

                [

                    InlineKeyboardButton("1️⃣", callback_data=f'plll 0|{query}|{user_id}'),

                    InlineKeyboardButton("2️⃣", callback_data=f'plll 1|{query}|{user_id}'),

                    InlineKeyboardButton("3️⃣", callback_data=f'plll 2|{query}|{user_id}'),

                ],

                [

                    InlineKeyboardButton("4️⃣", callback_data=f'plll 3|{query}|{user_id}'),

                    InlineKeyboardButton("5️⃣", callback_data=f'plll 4|{query}|{user_id}'),

                ],
```

```
                [InlineKeyboardButton(text="Close 🛑", callback_data="cls")],
            ]
        )
        await lel.edit(toxxt,reply_markup=koyboard,disable_web_page_preview=True)
        # WHY PEOPLE ALWAYS LOVE PORN ?? (A point to think)
        return
        # Returning to pornhub
    except:
        await lel.edit("No Enough results to choose.. Starting direct play..")


        # print(results)
        try:
            url = f"https://youtube.com{results[0]['url_suffix']}"
            title = results[0]["title"][:40]
            thumbnail = results[0]["thumbnails"][0]
            thumb_name = f"thumb{title}.jpg"
            thumb = requests.get(thumbnail, allow_redirects=True)
            open(thumb_name, "wb").write(thumb.content)
            duration = results[0]["duration"]
```

```
            results[0]["url_suffix"]
            views = results[0]["views"]


        except Exception as e:
            await lel.edit(
                "Song not found.Try another song or maybe spell it properly."
            )
            print(str(e))
            return
        dlurl=url
        dlurl=dlurl.replace("youtube","youtubepp")
        keyboard = InlineKeyboardMarkup(
            [
                [
                    InlineKeyboardButton("📖 Playlist", callback_data="playlist"),
                    InlineKeyboardButton("Menu ⏯ ", callback_data="menu"),
                ],
                [
                    InlineKeyboardButton(text="🎬 YouTube", url=f"{url}"),
                    InlineKeyboardButton(text="Download 📥 ", url=f"{dlurl}"),
                ],
```

```
                [InlineKeyboardButton(text="❌ Close", callback_data="cls")],
            ]
        )
        requested_by = message.from_user.first_name
        await generate_cover(requested_by, title, views, duration, thumbnail)
        file_path = await convert(youtube.download(url))
    chat_id = get_chat_id(message.chat)
    if chat_id in callsmusic.pytgcalls.active_calls:
        position = await queues.put(chat_id, file=file_path)
        qeue = que.get(chat_id)
        s_name = title
        r_by = message.from_user
        loc = file_path
        appendable = [s_name, r_by, loc]
        qeue.append(appendable)
        await message.reply_photo(
            photo="final.png",
            caption=f"#⃣ Your requested song **queued** at position {position}!",
            reply_markup=keyboard,
        )
        os.remove("final.png")
```

```
        return await lel.delete()
    else:
        chat_id = get_chat_id(message.chat)
        que[chat_id] = []
        qeue = que.get(chat_id)
        s_name = title
        r_by = message.from_user
        loc = file_path
        appendable = [s_name, r_by, loc]
        qeue.append(appendable)
        try:
            callsmusic.pytgcalls.join_group_call(chat_id, file_path)
        except:
            message.reply("Group Call is not connected or I can't join it")
            return
        await message.reply_photo(
            photo="final.png",
            reply_markup=keyboard,
            caption="▶️ **Playing** here the song requested by {} via Youtube Music 😜".format(
                message.from_user.mention()
            ),
        )
```

```
        os.remove("final.png")
        return await lel.delete()



@Client.on_message(filters.command("ytplay") &
filters.group & ~filters.edited)
async def ytplay(_, message: Message):
    global que
    if message.chat.id in DISABLED_GROUPS:
        return
    lel = await message.reply("🔄 **Processing**")
    administrators = await
get_administrators(message.chat)
    chid = message.chat.id


    try:
        user = await USER.get_me()
    except:
        user.first_name = "helper"
    usar = user
    wew = usar.id
    try:
        # chatdetails = await USER.get_chat(chid)
        await _.get_chat_member(chid, wew)
    except:
```

```
        for administrator in administrators:
            if administrator == message.from_user.id:
                if message.chat.title.startswith("Channel
Music: "):
                    await lel.edit(
                        "<b>Remember to add helper to your
channel</b>",
                    )
                    pass
                try:
                    invitelink = await
_.export_chat_invite_link(chid)
                except:
                    await lel.edit(
                        "<b>Add me as admin of yor group
first</b>",
                    )
                    return

                try:
                    await USER.join_chat(invitelink)
                    await USER.send_message(
                        message.chat.id, "I joined this group for
playing music in VC"
                    )
                    await lel.edit(
```

```
                "<b>helper userbot joined your
chat</b>",
            )


        except UserAlreadyParticipant:
            pass
        except Exception:
            # print(e)
            await lel.edit(
                f"<b>🔴 Flood Wait Error 🔴 \nUser
{user.first_name} couldn't join your group due to
heavy requests for userbot! Make sure user is not
banned in group."
                "\n\nOr manually add assistant to your
Group and try again</b>",
            )
    try:
        await USER.get_chat(chid)
        # lmoa = await
client.get_chat_member(chid,wew)
    except:
        await lel.edit(
            f"<i> {user.first_name} Userbot not in this chat,
Ask admin to send /play command for first time or
add {user.first_name} manually</i>"
        )
        return
```

```
    await lel.edit("🔎 **Finding**")
    user_id = message.from_user.id
    user_name = message.from_user.first_name



    query = ""
    for i in message.command[1:]:
        query += " " + str(i)
    print(query)
    await lel.edit("🎵 **Processing**")
    ydl_opts = {"format": "bestaudio[ext=m4a]"}
    try:
        results = YoutubeSearch(query,
max_results=1).to_dict()
        url = f"https://youtube.com{results[0]
['url_suffix']}"
        # print(results)
        title = results[0]["title"][:40]
        thumbnail = results[0]["thumbnails"][0]
        thumb_name = f"thumb{title}.jpg"
        thumb = requests.get(thumbnail,
allow_redirects=True)
        open(thumb_name, "wb").write(thumb.content)
        duration = results[0]["duration"]
        results[0]["url_suffix"]
```

```
        views = results[0]["views"]


    except Exception as e:
        await lel.edit(
            "Song not found.Try another song or maybe spell it properly."
        )
        print(str(e))
        return
    dlurl=url
    dlurl=dlurl.replace("youtube","youtubepp")
    keyboard = InlineKeyboardMarkup(
            [
                [
                 InlineKeyboardButton(text="🎬 YouTube", url=f"{url}"),
                 InlineKeyboardButton(text="Download 📥", url=f"{dlurl}"),
                ],
                [InlineKeyboardButton(text="❌ Close", callback_data="cls")],
            ]
    )
    requested_by = message.from_user.first_name
    await generate_cover(requested_by, title, views, duration, thumbnail)
```

```
    file_path = await convert(youtube.download(url))
    chat_id = get_chat_id(message.chat)
    if chat_id in callsmusic.pytgcalls.active_calls:
        position = await queues.put(chat_id,
file=file_path)
        qeue = que.get(chat_id)
        s_name = title
        r_by = message.from_user
        loc = file_path
        appendable = [s_name, r_by, loc]
        qeue.append(appendable)
        await message.reply_photo(
            photo="final.png",
            caption=f"#⃣ Your requested song **queued**
at position {position}!",
            reply_markup=keyboard,
        )
        os.remove("final.png")
        return await lel.delete()
    else:
        chat_id = get_chat_id(message.chat)
        que[chat_id] = []
        qeue = que.get(chat_id)
        s_name = title
        r_by = message.from_user
```

```
        loc = file_path
        appendable = [s_name, r_by, loc]
        qeue.append(appendable)
        try:
            callsmusic.pytgcalls.join_group_call(chat_id, file_path)
        except:
            message.reply("Group Call is not connected or I can't join it")
            return
        await message.reply_photo(
            photo="final.png",
            reply_markup=keyboard,
            caption="▶️ **Playing** here the song requested by {} via Youtube Music 😜".format(
                message.from_user.mention()
            ),
        )
        os.remove("final.png")
        return await lel.delete()


@Client.on_message(filters.command("dplay") & filters.group & ~filters.edited)
async def deezer(client: Client, message_: Message):
    if message_.chat.id in DISABLED_GROUPS:
        return
```

```
    global que
    lel = await message_.reply("🔄 **Processing**")
    administrators = await
get_administrators(message_.chat)
    chid = message_.chat.id
    try:
        user = await USER.get_me()
    except:
        user.first_name = "DaisyMusic"
    usar = user
    wew = usar.id
    try:
        # chatdetails = await USER.get_chat(chid)
        await client.get_chat_member(chid, wew)
    except:
        for administrator in administrators:
            if administrator == message_.from_user.id:
                if message_.chat.title.startswith("Channel
Music: "):
                    await lel.edit(
                        "<b>Remember to add helper to your
channel</b>",
                    )
                    pass
                try:
```

```
            invitelink = await
client.export_chat_invite_link(chid)
        except:
            await lel.edit(
                "<b>Add me as admin of yor group
first</b>",
            )
            return

        try:
            await USER.join_chat(invitelink)
            await USER.send_message(
                message_.chat.id, "I joined this group for
playing music in VC"
            )
            await lel.edit(
                "<b>helper userbot joined your
chat</b>",
            )

        except UserAlreadyParticipant:
            pass
        except Exception:
            # print(e)
            await lel.edit(
```

```
                    f"<b>🔴 Flood Wait Error 🔴 \nUser {user.first_name} couldn't join your group due to heavy requests for userbot! Make sure user is not banned in group."
                    "\n\nOr manually add assistant to your Group and try again</b>",
                )
    try:
        await USER.get_chat(chid)
        # lmoa = await client.get_chat_member(chid,wew)
    except:
        await lel.edit(
            f"<i> {user.first_name} Userbot not in this chat, Ask admin to send /play command for first time or add {user.first_name} manually</i>"
        )
        return
    requested_by = message_.from_user.first_name


    text = message_.text.split(" ", 1)
    queryy = text[1]
    query = queryy
    res = lel
    await res.edit(f"Searching 👀👀👀👀👀 for `{queryy}` on deezer")
    try:
```

```
        songs = await arq.deezer(query,1)
        if not songs.ok:
            await message_.reply_text(songs.result)
            return
        title = songs.result[0].title
        url = songs.result[0].url
        artist = songs.result[0].artist
        duration = songs.result[0].duration
        thumbnail =
"https://telegra.ph/file/f6086f8909fbfeb0844f2.png"


    except:
        await res.edit("Found Literally Nothing, You
Should Work On Your English!")
        return
    try:
        duuration= round(duration / 60)
        if duuration > DURATION_LIMIT:
            await cb.message.edit(f"Music longer than
{DURATION_LIMIT}min are not allowed to play")
            return
    except:
        pass


    keyboard = InlineKeyboardMarkup(
```

```
        [
            [
                InlineKeyboardButton("📖 Playlist", callback_data="playlist"),
                InlineKeyboardButton("Menu ⏯ ", callback_data="menu"),
            ],
            [InlineKeyboardButton(text="Listen On Deezer 🎬", url=f"{url}")],
            [InlineKeyboardButton(text="❌ Close", callback_data="cls")],
        ]
    )
    file_path = await convert(wget.download(url))
    await res.edit("Generating Thumbnail")
    await generate_cover(requested_by, title, artist, duration, thumbnail)
    chat_id = get_chat_id(message_.chat)
    if chat_id in callsmusic.pytgcalls.active_calls:
        await res.edit("adding in queue")
        position = await queues.put(chat_id, file=file_path)
        qeue = que.get(chat_id)
        s_name = title
        r_by = message_.from_user
        loc = file_path
```

```
        appendable = [s_name, r_by, loc]
        qeue.append(appendable)
        await res.edit_text(f"✯{bn}✯= #️⃣ Queued at
position {position}")
    else:
        await res.edit_text(f"✯{bn}✯=▶️ Playing.....")


        que[chat_id] = []
        qeue = que.get(chat_id)
        s_name = title
        r_by = message_.from_user
        loc = file_path
        appendable = [s_name, r_by, loc]
        qeue.append(appendable)
        try:
            callsmusic.pytgcalls.join_group_call(chat_id,
file_path)
        except:
            res.edit("Group call is not connected of I can't
join it")
            return


    await res.delete()


    m = await client.send_photo(
```

```
        chat_id=message_.chat.id,
        reply_markup=keyboard,
        photo="final.png",
        caption=f"Playing [{title}]({url}) Via Deezer",
    )
    os.remove("final.png")



@Client.on_message(filters.command("splay") &
filters.group & ~filters.edited)
async def jiosaavn(client: Client, message_: Message):
    global que
    if message_.chat.id in DISABLED_GROUPS:
        return
    lel = await message_.reply("🔄 **Processing**")
    administrators = await
get_administrators(message_.chat)
    chid = message_.chat.id
    try:
        user = await USER.get_me()
    except:
        user.first_name = "DaisyMusic"
    usar = user
    wew = usar.id
    try:
```

```
        # chatdetails = await USER.get_chat(chid)
        await client.get_chat_member(chid, wew)
    except:
        for administrator in administrators:
            if administrator == message_.from_user.id:
                if message_.chat.title.startswith("Channel
Music: "):
                    await lel.edit(
                        "<b>Remember to add helper to your
channel</b>",
                    )
                    pass
                try:
                    invitelink = await
client.export_chat_invite_link(chid)
                except:
                    await lel.edit(
                        "<b>Add me as admin of yor group
first</b>",
                    )
                    return

                try:
                    await USER.join_chat(invitelink)
                    await USER.send_message(
```

```
                    message_.chat.id, "I joined this group for playing music in VC"
                )
                await lel.edit(
                    "<b>helper userbot joined your chat</b>",
                )


            except UserAlreadyParticipant:
                pass
            except Exception:
                # print(e)
                await lel.edit(
                    f"<b>🔴 Flood Wait Error 🔴 \nUser {user.first_name} couldn't join your group due to heavy requests for userbot! Make sure user is not banned in group."
                    "\n\nOr manually add @InnexiaMusic to your Group and try again</b>",
                )
    try:
        await USER.get_chat(chid)
        # lmoa = await client.get_chat_member(chid,wew)
    except:
        await lel.edit(
```

```
            "<i> helper Userbot not in this chat, Ask admin to send /play command for first time or add assistant manually</i>"
        )
        return
    requested_by = message_.from_user.first_name
    chat_id = message_.chat.id
    text = message_.text.split(" ", 1)
    query = text[1]
    res = lel
    await res.edit(f"Searching 👀👀👀👀👀 for `{query}` on jio saavn")
    try:
        songs = await arq.saavn(query)
        if not songs.ok:
            await message_.reply_text(songs.result)
            return
        sname = songs.result[0].song
        slink = songs.result[0].media_url
        ssingers = songs.result[0].singers
        sthumb = songs.result[0].image
        sduration = int(songs.result[0].duration)
    except Exception as e:
        await res.edit("Found Literally Nothing!, You Should Work On Your English.")
        print(str(e))
```

```
        return
    try:
        duuration= round(sduration / 60)
        if duuration > DURATION_LIMIT:
            await cb.message.edit(f"Music longer than {DURATION_LIMIT}min are not allowed to play")
            return
    except:
        pass
    keyboard = InlineKeyboardMarkup(
        [
            [
                InlineKeyboardButton("📖 Playlist", callback_data="playlist"),
                InlineKeyboardButton("Menu ⏯ ", callback_data="menu"),
            ],
            [
                InlineKeyboardButton(
                    text="Join Updates Channel", url=f"https://t.me/{updateschannel}"
                )
            ],
            [InlineKeyboardButton(text="❌ Close", callback_data="cls")],
        ]
```

```
    )
    file_path = await convert(wget.download(slink))
    chat_id = get_chat_id(message_.chat)
    if chat_id in callsmusic.pytgcalls.active_calls:
        position = await queues.put(chat_id,
file=file_path)
        qeue = que.get(chat_id)
        s_name = sname
        r_by = message_.from_user
        loc = file_path
        appendable = [s_name, r_by, loc]
        qeue.append(appendable)
        await res.delete()
        m = await client.send_photo(
            chat_id=message_.chat.id,
            reply_markup=keyboard,
            photo="final.png",
            caption=f"★{bn}★=#⃣ Queued at position
{position}",
        )


    else:
        await res.edit_text(f"{bn}=▶️ Playing.....")
        que[chat_id] = []
        qeue = que.get(chat_id)
```

```
        s_name = sname
        r_by = message_.from_user
        loc = file_path
        appendable = [s_name, r_by, loc]
        qeue.append(appendable)
        try:
            callsmusic.pytgcalls.join_group_call(chat_id,
file_path)
        except:
            res.edit("Group call is not connected of I can't
join it")
            return
    await res.edit("Generating Thumbnail.")
    await generate_cover(requested_by, sname,
ssingers, sduration, sthumb)
    await res.delete()
    m = await client.send_photo(
        chat_id=message_.chat.id,
        reply_markup=keyboard,
        photo="final.png",
        caption=f"Playing {sname} Via Jiosaavn",
    )
    os.remove("final.png")
```

```
@Client.on_callback_query(filters.regex(pattern=r"plll
"))
async def lol_cb(b, cb):
    global que


    cbd = cb.data.strip()
    chat_id = cb.message.chat.id
    typed_=cbd.split(None, 1)[1]
    #useer_id =
cb.message.reply_to_message.from_user.id
    try:
        x,query,useer_id = typed_.split("|")
    except:
        await cb.message.edit("Song Not Found")
        return
    useer_id = int(useer_id)
    if cb.from_user.id != useer_id:
        await cb.answer("You ain't the person who
requested to play the song!", show_alert=True)
        return
    await cb.message.edit("Hang On... Player Starting")
    x=int(x)
    try:
        useer_name =
cb.message.reply_to_message.from_user.first_name
    except:
```

```
        useer_name = cb.message.from_user.first_name


    results = YoutubeSearch(query,
max_results=5).to_dict()
    resultss=results[x]["url_suffix"]
    title=results[x]["title"][:40]
    thumbnail=results[x]["thumbnails"][0]
    duration=results[x]["duration"]
    views=results[x]["views"]
    url = f"https://youtube.com{resultss}"


    try:
        duuration= round(duration / 60)
        if duuration > DURATION_LIMIT:
            await cb.message.edit(f"Music longer than
{DURATION_LIMIT}min are not allowed to play")
            return
    except:
        pass
    try:
        thumb_name = f"thumb{title}.jpg"
        thumb = requests.get(thumbnail,
allow_redirects=True)
        open(thumb_name, "wb").write(thumb.content)
    except Exception as e:
```

```
        print(e)
        return
    dlurl=url
    dlurl=dlurl.replace("youtube","youtubepp")
    keyboard = InlineKeyboardMarkup(
            [
                [
                 InlineKeyboardButton(text="🎬 YouTube",
url=f"{url}"),
                 InlineKeyboardButton(text="Download 📥",
url=f"{dlurl}"),
                ],
                [InlineKeyboardButton(text="❌ Close",
callback_data="cls")],
            ]
    )
    requested_by = useer_name
    await generate_cover(requested_by, title, views,
duration, thumbnail)
    file_path = await convert(youtube.download(url))
    if chat_id in callsmusic.pytgcalls.active_calls:
        position = await queues.put(chat_id,
file=file_path)
        qeue = que.get(chat_id)
        s_name = title
        try:
```

```
            r_by = cb.message.reply_to_message.from_user
        except:
            r_by = cb.message.from_user
        loc = file_path
        appendable = [s_name, r_by, loc]
        qeue.append(appendable)
        await cb.message.delete()
        await b.send_photo(chat_id,
            photo="final.png",
            caption=f"#□ Song requested by {r_by.mention} **queued** at position {position}!",
            reply_markup=keyboard,
        )
        os.remove("final.png")


    else:
        que[chat_id] = []
        qeue = que.get(chat_id)
        s_name = title
        try:
            r_by = cb.message.reply_to_message.from_user
        except:
            r_by = cb.message.from_user
        loc = file_path
        appendable = [s_name, r_by, loc]
```

```
        qeue.append(appendable)


        callsmusic.pytgcalls.join_group_call(chat_id,
file_path)
        await cb.message.delete()
        await b.send_photo(chat_id,
            photo="final.png",
            reply_markup=keyboard,
            caption=f"▶️ **Playing** here the song
requested by {r_by.mention} \n** Duration {duration}
\n ",
        )


        os.remove("final.png")


# Have u read all. If read RESPECT :-)
```