Python Project Report

13006107 Introduction to Computers and Programming

Software Engineering Program

Faculty of Engineering, KMITL

By

64011338 Anucha Cheewachanon

Project Description

This project is a python discord music bot. It is lightweight and cross-platform. There are commands and features that you may expect from the contemporary discord bots. This includes pausing, queuing, checking the playing song's name, etc. It uses discord.py library with youtube_dl to handle the music source. The objective for this bot is for everyone to use to listen to music with their friends in their discord server.

Code:

```
import discord
from discord.ext import commands
import asyncio
from async_timeout import timeout
import tkinter
from discord.player import FFmpegAudio
import youtube_dl
import json
import itertools
import datetime

ytdl_format_options = {
    'format': 'bestaudio/best',
    'outtmpl': '%(extractor)s-%(id)s-%(title)s.%(ext)s',
    'restrictfilenames': True,
    'noplaylist': True,
    'nocheckcertificate': True,
    'ignoreerrors': False,
    'logtostderr': True,
    'quiet': True,
    'no_warnings': True,
    'default_search': 'auto',
}
ffmpeg_options = {'options': '-vn'}

ytdl = youtube_dl.YoutubeDL(ytdl_format_options)

class Source(discord.PCMVolumeTransformer):
    def __init__(self,source,*,data,requester):
        super().__init__(source)
```

```
        self.data = data

        self.title = data.get('title')
        self.url = data.get('url')
        self.duration = data.get('duration')
        self.webpage_url = data.get('webpage_url')
        self.requester = requester

    def __getitem__(self, item: str): #so that it becomes subscriptable.
        return self.__getattribute__(item)

    @classmethod
    async def from_url(cls,ctx,url,*,loop=None,dl=False): #PLEASE THE
DEFAULT MDOE IS STREAM. I AM HOSTING THIS ON MY OWN PC
        loop = loop or asyncio.get_event_loop()
        data = await loop.run_in_executor(None, lambda:
ytdl.extract_info(url,download=dl))

        if 'entries' in data:
            data = data['entries'][0]

        await ctx.send(embed=discord.Embed(title='Song added to the
queue',description=f"Added [{data['title']}]({data['webpage_url']}) to
the queue.",color=discord.Color.green()))
        filename = ytdl.prepare_filename(data) if dl else data['url']
        return
cls(discord.FFmpegPCMAudio(filename,**ffmpeg_options),data=data,requester
=ctx.author)

class MusicPlayer: #a musicplayer object for tomkai to listen to wangy
wangy and lmao to listen to love supreme at the same time
    def __init__(self,ctx):
        self.bot = ctx.bot
        self.guild = ctx.guild
        self.channel = ctx.channel
        self.cog = ctx.cog

        self.queue = asyncio.Queue()
        self.next = asyncio.Event()

        self.np = None
        self.current = None

        ctx.bot.loop.create_task(self.playerloop())

    async def playerloop(self): #the main magic
        await self.bot.wait_until_ready()
```

```
        while not self.bot.is_closed():
            self.next.clear()
            try:
                async with timeout(180):
                    source = await self.queue.get()
            except asyncio.TimeoutError:
                await self.channel.send("adios")
                return self.destroy(self.guild)

            source.volume = self.volume
            self.current = source
            self.guild.voice_client.play(source, after=lambda i:
self.bot.loop.call_soon_threadsafe(self.next.set))
            self.np = await
self.channel.send(embed=discord.Embed(title='Now
playing',description=f"[{source['title']}]({source['webpage_url']})",colo
r=discord.Color.blue()))
            await self.next.wait()

            source.cleanup()
            self.current = None

    def destroy(self, guild): #tells the cog to cleanup()
        return self.bot.loop.create_task(self.cog.cleanup(guild))

class MusicCommand(commands.Cog):
    def __init__(self,bot):
        self.bot=bot
        self.players = dict()

    async def cleanup(self,guild):
        try:
            await guild.voice_client.disconnect()
        except AttributeError:
            pass
        try:
            del self.players[guild.id]
        except KeyError:
            pass

    def getPlayer(self,ctx):
        try:
            player = self.players[ctx.guild.id]
        except KeyError:
            player = MusicPlayer(ctx)
            self.players[ctx.guild.id] = player
        return player
```

```
    @commands.command(aliases=['connect', 'j', 'con'])
    async def join(self, ctx, *, channel: discord.VoiceChannel):
        """Tells the bot to join the channel you're currently in"""
        if channel:
            return await ctx.send("Someone is currently using the bot in
the other channel.")
        if ctx.voice_client is not None:
            return await ctx.voice_client.move_to(channel)

        await channel.connect()

    @commands.command(aliases=['p'])
    async def play(self,ctx,*,url):
        """Add a song to the queue."""
        async with ctx.typing():
            player = self.getPlayer(ctx)
            source = await
Source.from_url(ctx,url,loop=self.bot.loop,dl=False)
            if not isinstance(source,Source):
                return await ctx.send(embed=discord.Embed(title="Error
adding the song",description="There has been anerror adding the song to
the queue.",color=discord.Color.red()))
        await player.queue.put(source)

    @commands.command(aliases=['s'])
    async def skip(self,ctx):
        """Skip the current song."""
        if ctx.voice_client is None:
            return await ctx.send(embed=discord.Embed(title="Error
skipping the song",description="I am not even in a voice
channel!",color=discord.Color.red()))

        if ctx.voice_client.is_paused():
            pass
        elif not ctx.voice_client.is_playing():
            return await ctx.send(embed=discord.Embed(title="Error
skipping the song",description="I am not playing any
song!",color=discord.Color.red()))

        ctx.voice_client.stop()

    @commands.command(aliases=['q'])
    async def queue(self,ctx):
        """Shows the queue"""
        if ctx.voice_client is None:
            return await ctx.send(embed=discord.Embed(title="Error
displaying the queue",description="I am not playing any
song!",color=discord.Color.red()))
```

```
        player = self.getPlayer(ctx)

        qlst =
list(itertools.islice(player.queue._queue,0,int(len(player.queue._queue))
))
        actlst = '\n'.join(f"{(qlst.index(i))+1}. {i.title}
`{datetime.timedelta(seconds=i.duration)}` Requested by {i.requester}\n"
for i in qlst)
        retstr = f"\n Currently playing: {ctx.voice_client.source.title}
`{datetime.timedelta(seconds=ctx.voice_client.source.duration)}`\n\nUp
next:\n"+actlst
        await ctx.invoke(self.nowplaying)
        await
ctx.send(embed=discord.Embed(title="Queue",description=retstr,color=disco
rd.Color.blue()))

    @commands.command(aliases=['rm'])
    async def remove(self,ctx,pos:int=None):
        """Removes a song from the queue"""
        if ctx.voice_client is None:
            return await ctx.send(embed=discord.Embed(title="Error
removing the song from the queue",description="I am not in a voice
channel!",color=discord.Color.red()))

        player = self.getPlayer(ctx)
        if pos == None:
            player.queue._queue.pop()
        else:
            try:
                target = player.queue._queue[pos-1]
                del player.queue._queue[pos-1] #cannot del target as it
will just remove the pointer to target's target
                await
ctx.send(embed=discord.Embed(title="Done!",description=f"Removed
[{target['title']}]({target['webpage_url']}) from the
queue.",color=discord.Color.teal()))
            except:
                return await ctx.send(embed=discord.Embed(title="Error
removing the song from the queue",description="Song not
found.",color=discord.Color.red()))

    @commands.command(aliases=['clr'])
    async def clear(self,ctx):
        """Clear the queue"""
        if ctx.voice_client is None:
```

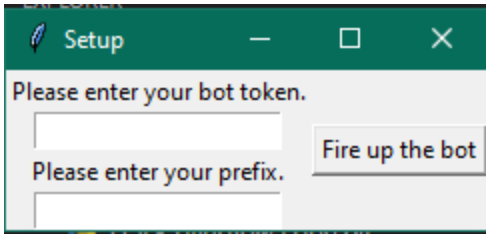
```
            return await ctx.send(embed=discord.Embed(title="Error
clearing the queue",description="I am not in a voice
channel!",color=discord.Color.red()))

        self.getPlayer(ctx).queue._queue.clear()
        await ctx.send("Queue cleared!")

    @commands.command()
    async def pause(self,ctx):
        """Pause the music. Use this command again to unpause"""
        if ctx.voice_client is None:
            return await ctx.send(embed=discord.Embed(title="Error
pausing the music",description="I am not in a voice
channel!",color=discord.Color.red()))
        elif not ctx.voice_client.is_playing():
            return await ctx.send(embed=discord.Embed(title="Error
pausing the music",description="I am not playing
anything",color=discord.Color.red()))
        if not ctx.voice_client.is_paused():
            ctx.voice_client.pause()
            await ctx.send("Paused.")
        else:
            ctx.voice_client.resume()
            await ctx.send("Resume playing.")

    @commands.command(aliases=['np'])
    async def nowplaying(self,ctx):
        """See what's playing"""
        saus = ctx.voice_client.source
        if ctx.voice_client is None:
            return await ctx.send(embed=discord.Embed(title="Error
displaying music info",description="I am not in a voice
channel!",color=discord.Color.red()))
        elif not ctx.voice_client.is_playing():
            return await ctx.send(embed=discord.Embed(title="Error
displaying music info",description="I am not playing
anything",color=discord.Color.red()))

        time = str(datetime.timedelta(seconds=saus.duration))
        await ctx.send(embed=discord.Embed(title="Now
playing!",description=f"[{saus.title}]({saus.url}) `{time}` Requested by
{saus.requester}",color=discord.Color.teal()))

    @commands.command()
    async def stop(self, ctx):
        """Stops and disconnects the bot from voice. Also destroys the
player"""
        if ctx.voice_client is None:
```

```
            return await ctx.send(embed=discord.Embed(title="Error
leaving the channel",description="I am not in a voice
channel!",color=discord.Color.red()))
        await ctx.send("adios")
        await self.cleanup(ctx.guild)

    @play.before_invoke
    async def ensure_voice(self,ctx):
        if ctx.voice_client is None:
            if ctx.author.voice:
                await ctx.author.voice.channel.connect()
            else:
                await ctx.send("You are not connected to a voice
channel.")
                raise commands.CommandError("Author not connected to a
voice channel.")

class Tokenwindow: #use NewSetup instead
    def __init__(self):
        self.window = tkinter.Tk()
        self.window.title("Enter your token")
        self.tokenVar = tkinter.StringVar()
        tkinter.Label(self.window,text="Please enter your bot
token.").pack()
        tkinter.Entry(self.window,textvariable=self.tokenVar).pack()
        tkinter.Button(self.window,text="Ok",command=self.savetoken).pack
()
        self.window.mainloop()

    def savetoken(self):
        global token
        token=self.tokenVar.get()
        self.window.destroy()

class Prefixwindow: #use NewSetup instead
    def __init__(self):
        self.window = tkinter.Tk()
        self.window.title("Enter your prefix")
        self.tokenVar = tkinter.StringVar()
        tkinter.Label(self.window,text="Please enter your
prefix.").pack()
        tkinter.Entry(self.window,textvariable=self.tokenVar).pack()
        tkinter.Button(self.window,text="Ok",command=self.savetoken).pack
()
        self.window.mainloop()

    def savetoken(self):
        global prefix
```

```
        prefix=self.tokenVar.get()
        self.window.destroy()

class NewSetup():
    def __init__(self,token="",pref=""):
        self.window = tkinter.Tk()
        self.window.title("Setup")
        self.prefixVar = tkinter.StringVar()
        self.tokenVar = tkinter.StringVar()
        tkinter.Label(self.window,text="Please enter your bot
token.").grid(row=1,column=1)
        tkinter.Entry(self.window,textvariable=self.tokenVar,show="*").gr
id(row=2,column=1)
        tkinter.Label(self.window,text="Please enter your
prefix.").grid(row=3,column=1)
        tkinter.Entry(self.window,textvariable=self.prefixVar).grid(row=4
,column=1)
        tkinter.Button(self.window,text="Fire up the
bot",command=self.savetoken).grid(row=1,column=2,rowspan=4)
        self.window.mainloop()

    def savetoken(self):
        global prefix
        global token
        token=self.tokenVar.get()
        prefix=self.prefixVar.get()
        self.window.destroy()

try:
    settings = open("settings.json","r")
except FileNotFoundError:
    NewSetup()
    newcfgdata = {"token":token,"prefix":prefix}
    with open("settings.json", "w") as outfile:
        json.dump(newcfgdata, outfile)
finally:
    with open("settings.json","r") as json_file:
        cfg = json.load(json_file)
try:
    prefix = cfg.get('prefix')
except KeyError:
    Prefixwindow()

try:
    token = cfg.get('token')
except KeyError:
    Tokenwindow()
```

```
bot=commands.Bot(command_prefix=commands.when_mentioned_or(prefix),descri
ption="A simple music bot. What do you expect?")

@bot.event
async def on_ready():
    print(f"\nI'm in as {bot.user} {bot.user.id}\n")
    await bot.change_presence(status=discord.Status.online,
activity=discord.Game("Miss Kobayashi's Dragon Maid Burst Forth!! Choro-
gon☆Breath"))
bot.add_cog(MusicCommand(bot))
bot.run(token)
```

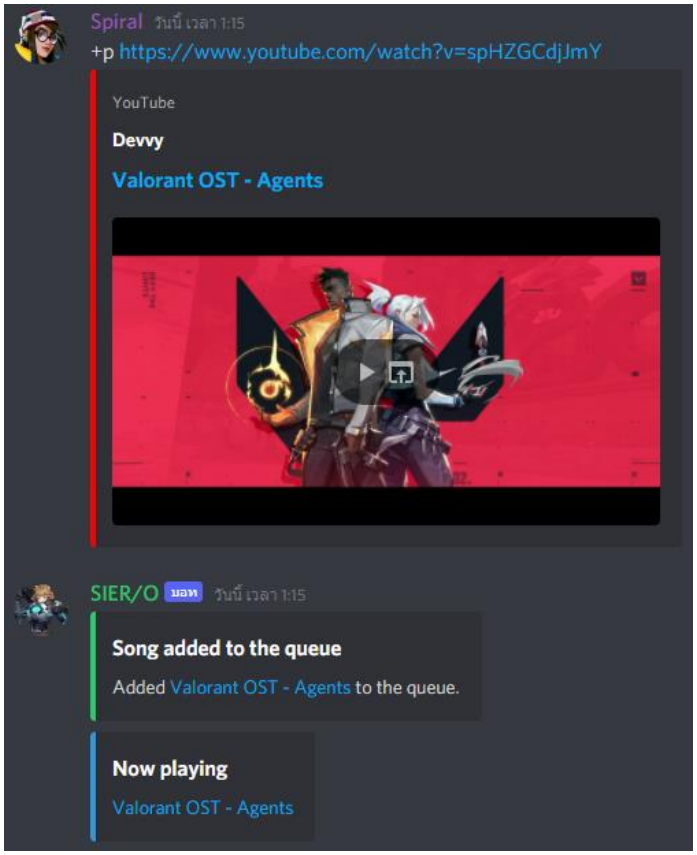
Preview:



The Setup window

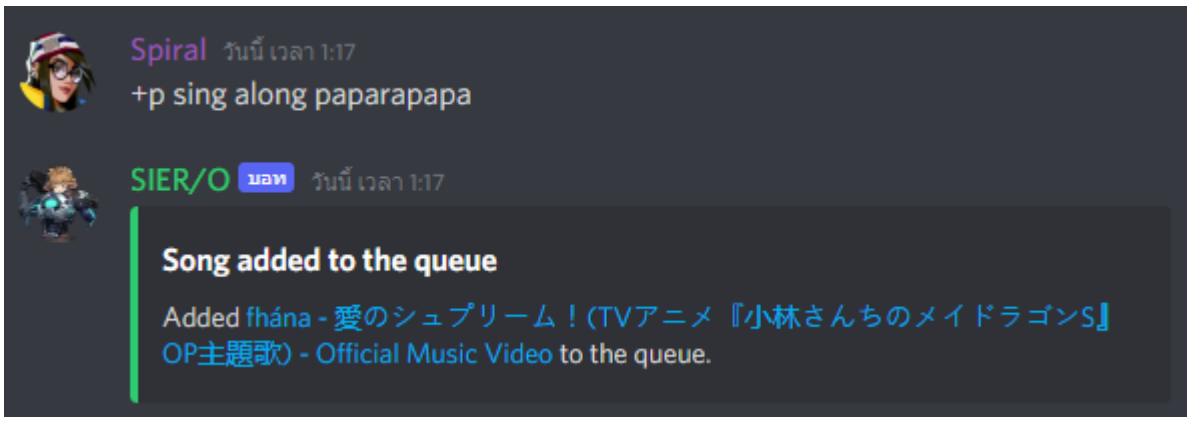
```
I'm in as Sierokarte Gaming#1159 697714696911847475
```

The message that returns when the bot has started



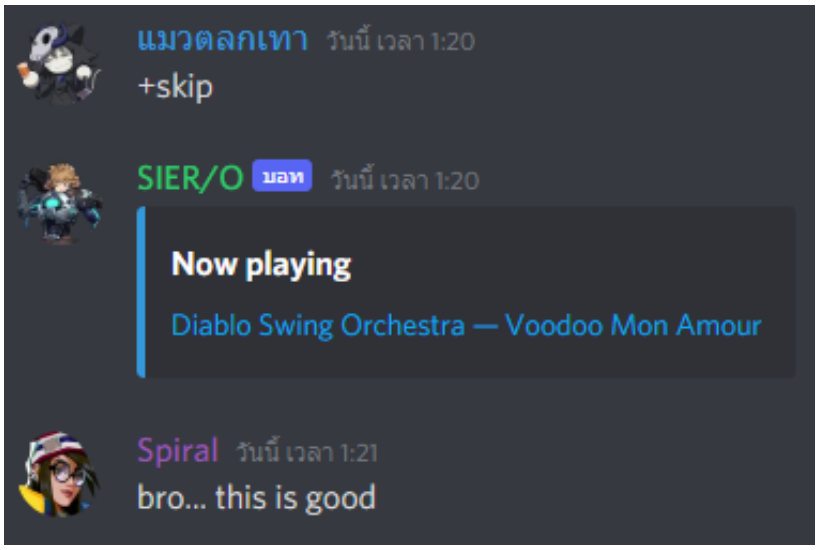
Spiral tells the bot to play Valorant music through a direct link

SIER/O (The bot) connects and play the music

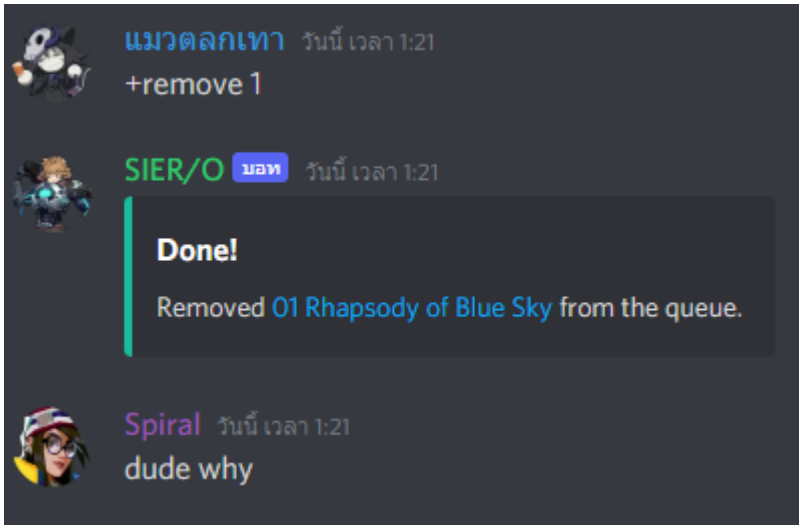


Spiral shows that you can just type the search term to play music too

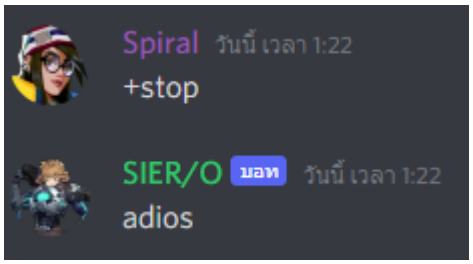
Funny grey cat demonstrates the queue function



Funny grey cat skips Spiral's music



Funny grey cat removes Spiral's music from the queue



Spiral tells the bot to disconnect