DISCORD BOT Computer Science Project

Name:

Aakash Alloria

Armaan Saini

Chaitanya Sharma

Class: XII-J

School: Delhi Public School, Sector-45,Gurgaon

TITLE

1. Certificate
2.Acknowledgment
3. Python Code
4. SQL Database
5. Output

CERTIFICATE

This is to certify that Chaitanya Sharma of class XII-J has prepared this project. This report is a culmination of his efforts and endeavors and has been accepted as the final project report for the subject Computer of class XII.

____

Ms. Chanchal Chandna

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my Computer Science teacher Ms. Chanchal Chandna for her vital support, guidance and encouragement, without which this project would not have come to be.

PYTHON CODE

FILE STRUCTURE

MAIN

|____Bot.py

|____ _cogs3_

|_____animecommands.py

|_____project.py

BOT.PY

```
#IMPORTS


from __future__ import print_function

import datetime

from datetime import timezone

import youtube_search

from youtube_search import YoutubeSearch

import math

import validators

import youtube_dl

import youtube_dlc

import discord

from discord.ext import commands

import unidecode

import json

import os
import urllib.request

import re

import spotipy

from spotipy.oauth2 import SpotifyClientCredentials

import youtube

import asyncio

from discord import Spotify

import emoji

import mysql.connector

import os

from googleapiclient.discovery import build

from googleapiclient.errors import

HttpError from oauth2client.tools import

argparser import random

import time

from discord import Member

import string as s

import asyncio
from jikanpy import Jikan

from discord import

Spotify import spotipy

import requests

import re

import lyricsgenius

import youtube_dlc

import struct

from PIL import Image

import numpy as np

from discord.utils import


get #SETTING BASIC VARIABLES


q={}

q["id"] = 0

q["channel"] = 0

q["guild"]=0
```

```
timer={}

timer['minutes']=0

timer['seconds']=0

timer['time_start']=0

timer["looping"]=False

timer['timer_now']=0

timer['currently_playing']="no"

timer['play']='yes'

timer['disconnecting']=False

timer["connected"]=False

timer["ctx"]=None

timer["skip"]=False

timer["title"]=None

timer["duration"]=None

timer["errorsearching"]="No"

timer["playlist"]=False

timer["songl"]=False


#SETTING API IDS
genius = lyricsgenius.Genius("mQxOjdgYx6XMzC8ERdff8uGH_BLoSCtlYMckMCo8L9YlNo9dhizZTAmVtHXJVLdV")

sp = spotipy.Spotify("BQAXQQYwP9M9qQbCFW36Y1O5mPrt3qNsbGvwMwTew8J9vWt2hw8mrtcocny2erkOvW2-
tij8ixWXznJrI5eGfve15Y1cJnN22rRKZb5MktBlL0oCQyDTFW1g2n_PT2B8MpmrtWp2bVomHI0HRVhyExY_GDWL1OnzsX4-
_hlLNjHG2bOO0E3uF60DTAUjvl0w4JyIpPW7KvmHWAZe5RiVM0mCfeBzr8FJFCfg9gpSsm4kxwrW3PYwIzz3YOcTQ5kFHM_b4pcfnKLiuNcF
a lzVf7_j6FJTMLoTwoEM")

jikan = Jikan()



#SETTING PROPER INTENTS


intents = discord.Intents.default()

intents.members = True


#YOUTUBE_API_SETUP


API_KEY = "AIzaSyDt2OJjpuyEo4LYbgywEGUBEdasyJj4GSY"

YOUTUBE_API_SERVICE_NAME = "youtube"

YOUTUBE_API_VERSION = "v3"

DEVELOPER_KEY = "AIzaSyAmmO3EOa4WZcP1L3AKgFrkAo07bPXwNGo"
```

```
#SEARCHING_YT_LINK_USING_YT_API


def search_by_keyword(qs):

 try:

 youtube = build(

 YOUTUBE_API_SERVICE_NAME,

 YOUTUBE_API_VERSION,

 developerKey=API_KEY

 )

 search_response = youtube.search().list(

 q=qs,

 part="id" ,

 maxResults=1

 ).execute()


 videos = []
 for search_result in search_response.get("items", []):

 if search_result["id"]["kind"] == "youtube#video":

 videos.append(search_result["id"]["videoId"])  return videos

 except Exception as e:

 raise e

 timer["errorsearching"]="Yes"

 return [0]

ydl_opts = {

 'format': 'bestaudio/best','ignore_errors':'True',


 'source_address': '0.0.0.0',

 'reconnect_streamed':True, # bind to ipv4 since ipv6 addresses cause issues

sometimes  'no_warnings': 'True',

 #'outtmpl': '%(extractor)s-%(id)s-%(title)s.%(ext)s',

 'noplaylist': True,


 'postprocessors': [{

 'key': 'FFmpegExtractAudio',

 'preferredcodec': 'mp3',

 'preferredquality': '384',
 }],
 }
ytdl = youtube_dlc.YoutubeDL(ydl_opts)
print(ytdl)

ffmpeg_options = {
 'options': '-vn'
}

global dirx2

dirx2=""

#GETTING LIST OF TRACKNAMES ALONG WITH ARTIST FROM SPOTIFY

LINKS def getTracks(playlistURL):

 with open("config.json", encoding='utf-8-sig') as json_file:
 APIs = json.load(json_file)
 # Creating and authenticating our Spotify app.
 client_credentials_manager = SpotifyClientCredentials(APIs["spotify"]["client_id"],
APIs["spotify"]["client_secret"])
 spotify = spotipy.Spotify(client_credentials_manager=client_credentials_manager)

 # Getting a playlist.
 results = spotify.user_playlist_tracks(user="",playlist_id=playlistURL)

 tracks = results['items']
 while results['next']:
 results = spotify.next(results)
 tracks.extend(results['items'])
 playlistdir=playlistURL.split("/playlist/")[1]+".txt"
 playlistdir=playlistdir.replace("?","")
 playlistdirfinal=playlistURL.split("/playlist/")[1]+"url.txt"
 playlistdirfinal=playlistdirfinal.replace("?","")
 dirx=R"C:\\Users\\aakas\\Desktop\\Persona 11-12\\Persona Bot\\spotifyplaylist\\" + playlistdir
```

```
global dirx2
 dirx2=R"C:\\Users\\aakas\\Desktop\\Persona 11-12\\Persona Bot\\spotifyplaylist\\" + playlistdirfinal


 print(dirx)
 #print(tracks[:10])
 trackList = []
 for i in tracks:
 try:


 if (i["track"]["artists"].__len__() == 1):
 trackList.append(i["track"]["name"] + " - " + i["track"]["artists"][0]["name"])  else:
 nameString = ""
 for index, b in enumerate(i["track"]["artists"]):
 nameString += (b["name"])
 # If it isn't the last artist.
 if (i["track"]["artists"].__len__() - 1 != index):
 nameString += ", "
 trackList.append(i["track"]["name"] + " - " + nameString)
 except:
  print(i)
 continue
 if os.path.isfile(dirx):
 L=[]
 newsongs=[]
 with open(dirx,"r",encoding="utf-8") as f:
 x=f.readlines()
 for i in x:
 i=i.replace("\n","")
 L.append(i)
 for i in trackList:
 if i in L:
 print("h")
 else:
 #print(i)
 newsongs.append(i)
 i=i+"\n"
 with open(dirx,"a",encoding='utf-8') as f:
```

```
 f.write(i)

 else:

 #print(dirx)

 with open(dirx,"w",encoding="utf-8") as f:

 newsongs=trackList

 for i in trackList:

 i=i+"\n"

 f.write(i)


 return newsongs


#SEARCHING_YT_LINK_FROM_URL_SCRAPER


def searchYoutubeAlternative(songName):

 try:

 search_keyword=songName.replace(" ","+")

 #search_keyword=search_keyword.replace(" ","+")

 html = urllib.request.urlopen(r"https://www.youtube.com/results?search_query=" + search_keyword)
 video_ids = re.findall(r"watch\?v=(\S{11})", html.read().decode())

 return("https://www.youtube.com/watch?v=" + video_ids[0])+" "+songName

 except:

 print(songName)

 pass


#ALTERNATIVE_WAY_TO_SEARCCH_YT_LINKS


def searchYoutube(songName):

 with open("config.json", encoding='utf-8-sig') as json_file:

 APIs = json.load(json_file)


 api = youtube.API(client_id=APIs["youtube"]["client_id"],

 client_secret=APIs["youtube"]["client_secret"],

 api_key=APIs["youtube"]["api_key"])

 video = api.get('search', q=songName,part="snippet", maxResults=1, type='video', order='relevance' , )

songName=songName.replace("'","")

 songName=songName.replace(")","")

 songName=songName.replace("(","")
 songName=songName.replace('"',"")

 return("https://www.youtube.com/watch?v="+video["items"][0]["id"]["videoId"])+" " +
video["items"][0]["snippet"]["title"]


#DEFINING_CLASS_FOR_THE_MUSIC_BOT


class YTDLSource(discord.PCMVolumeTransformer):


 def __init__(self, source, *, data, volume=0.5):

 super().__init__(source, volume)


 self.data = data


 self.title = data.get('title')

 self.url = data.get('url')

 @classmethod

 async def from_url(cls, url, *, loop=None, stream=False):

 loop = loop or asyncio.get_event_loop()

 data = await loop.run_in_executor(None, lambda: ytdl.extract_info(url, download=not stream))
 if 'entries' in data:tract_info(url, download=not stream)


 if 'entries' in data:

 # take first item from a playlist

 data = data['entries'][0]


 filename = data['url'] if stream else ytdl.prepare_filename(data)

 return cls(discord.FFmpegPCMAudio(filename, **ffmpeg_options,before_options=" -reconnect 1 -
reconnect_streamed 1 -reconnect_delay_max 5"), data=data)


#CREATING_DICTIONARY_CLASS_FOR_GLOBAL_DICTIONARY


class DictLikeClass:

 def __init__(self):

 super(DictLikeClass, self).__init__()


 def __getitem__(self, key):

 return getattr(self, key)
 def __setitem__(self, key, value):

 setattr(self, key, value)


ctr= [0]


#PROCESS_TO_REMOVE_SONGS_FROM_QUEUE_AFTER_SONG_ENDS


def my_after():

 while True:

 with open ("queue.txt","r") as f:

 x=f.readline()

 Queue=int(x)

 with open ("queue.txt" , "w") as f:

 queuenew=Queue-1

 f.write(str(queuenew))

 break

 timer['play']="no"

 print("done")
#ERROR_HANDLING_FOR_INCORRECT_QUEUE


async def errorqueue():

 channel=timer["channel"]

 with open ("queue.txt","r") as f:

 x=f.readline()

 Queue=int(x)

 with open ("queue.txt" , "w") as f:

 queuenew=Queue-1

 f.write(str(queuenew))

 timer['play']="no"

 print("done")

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord', )

cursor = db.cursor()

 cursor.execute("delete from music limit 1")

 db.commit()

 db.close()
 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord', )

cursor = db.cursor()

 cursor.execute("select * from music")

 results = cursor.fetchall()

 print(f"results {results}")

 if results==[]:

 ctr[0]=0

 print()

 print("CTR = 0")

 await channel.send("> Queue Finished Please Use `.leave` to disconnect the bot or `.p [songname]`  to play
more songs")

 print()

 timer['minutes']=0

 timer['seconds']=0

 timer['time_start']=0

 timer['timer_now']=0

 with open("id.txt","r") as f:

 results=f.readline()

 results=results.split(" ")

 results=results[1:]
 strxx=""

 print(results)

 for j in range(len(results)):

 i=results[j]

 print(i)

 if j==(len(results)-1):

 strxx=strxx+i

 elif i!=" ":

 strxx=strxx+i+" "

 print(strxx,end="")

 print("hello")

 with open ("id.txt","w") as f:

 f.write("")

 with open("id.txt","w") as f:

 f.write(str(strxx))

 print(f"file written {strxx}")

 timer['currently_playing']="no"
```

```
k={}

R=[]


def thanks_check(x):

 x=str(x)

 if x.isnumeric():

 return


#BOT_PREFIX


with open ("prefix.txt","r") as p:

 m=p.read(1)

 p.close()

bot =

commands.Bot(command_prefix=m,case_insensitive=True)

bot.remove_command('help')


#BOT_START


@bot.event
async def on_ready():

 print('Logged in as')

 print(bot.user.name)

 print(bot.user.id)

 print('-------')

 await bot.change_presence(status=discord.Status.online, activity=discord.Game(name="Discord Bot"))


#BOT_PREFIX


@bot.command()

async def prefix(ctx,a: str):

 with open('prefix.txt','a') as f:

 f.truncate(0)

 f.close()

 with open('prefix.txt','a') as p:

 p.write(a)
 p.close()

 print(a)

 bot.command_prefix(a)

 print(bot.command_prefix)

 str="Bot prefix set to",a

 await ctx.send(str)


#SPOTIFY_STATUS


@bot.command()

async def spotify(ctx,*, user: discord.Member=None):

 ctrx=0

 user = user or ctx.author

 for activity in user.activities:

 if isinstance(activity, Spotify):

 print("spotify")

 ctrx=1

 x1=(activity.end-datetime.datetime.utcnow())

 x=(x1.seconds)

 minutes=(x//60)

 seconds=(x%60)
 x2=(activity.duration)

 x3=(x2.seconds)

 minutes2=x3//60

 seconds2 =x3%60

 minutes3 = abs(minutes-minutes2)

 seconds3 = abs(seconds-seconds2)

 if len(str(minutes3))==1:

 minutes3="0"+str(minutes3)

 if len(str(minutes2))==1:

 minutes2="0"+str(minutes2)

 if len(str(seconds3))==1:

 seconds3="0"+str(seconds3)

 if len(str(seconds2))==1:

 seconds2="0"+str(seconds2)

 print(minutes3)

 print(seconds3)
 z=(f"{minutes3}:{seconds3}/{minutes2}:{seconds2}")

 titless=f"{activity.title} \n {z}"

 embed=discord.Embed(title=titless,description=activity.artist,color=activity.color)

embed.set_thumbnail(url=activity.album_cover_url)

 strwow=str(user.name+" is listening to:")

 strwow=s.capwords(strwow)

 embed.set_author(name=strwow,icon_url=user.avatar_url)

 embed.set_footer(text="Requested By:"+str(ctx.author),icon_url=ctx.author.avatar_url,)  await

ctx.send(embed=embed)

 if ctrx==0:

 await ctx.send(f"{user} is not listening to any song right now")


#JOIN_THE_MUSIC_CHANNEl


@bot.command(pass_context=True)

async def join(ctx):

 timer["ctx"]=ctx

 print(ctx.channel.id)

 print("Connecting")

 timer["channel"]=ctx.channel

 global voice
  channel=ctx.message.author.voice.channel

 voice = get(bot.voice_clients,guild = ctx.guild)

 if voice and voice.is_connected():

 await voice.move_to(channel)

 else:

 voice = await channel.connect()

 voice.stop()

 timer["connected"]=True

 await ctx.send(f"Joined {channel}")


#SET_THE_VOLUME_FOR_THE_MUSIC


@bot.command()

async def volume(ctx,a):

 if a=="max" or a=="Max" or a=="mAx" or a=="maX":
```

```
 a=20
 elif a=="min" or a=="Min" or a=="MIN":
 a=1
 a=float(a)
 if a>20:
 await ctx.send("Max Volume Limit Is 20")
 a=20
 await asyncio.sleep(0.2)
 voice = get(bot.voice_clients, guild=ctx.guild)
 if voice!=None:
 voice.source.volume = a/10

#ADDING_SONGS_FROM_A_YOUTUBE_PLAYLIST


@bot.command(pass_context=True,aliases =["pl"])
async def playlist(ctx,*,url: str):
 author=ctx.author.name
 await asyncio.sleep(0.2)
 voice = get(bot.voice_clients, guild=ctx.guild)
 if voice==None:
 await ctx.send("`Bot Needs To Be Connected To A Voice Channel To Use This Commands`")   return
 timer["ctx"]=ctx
 ydl_opts = {
 'format': 'bestaudio/best','ignore_errors':'True',

 'source_address': '0.0.0.0',
 'reconnect_streamed':True, # bind to ipv4 since ipv6 addresses cause issues sometimes
'no_warnings': 'True',
 'noplaylist': True,
 'postprocessors': [{
 'key': 'FFmpegExtractAudio',
 'preferredcodec': 'mp3',
 'preferredquality': '384',
 }],
 }
 await asyncio.sleep(0.2)
 voice = get(bot.voice_clients, guild=ctx.guild)

 with open ("queue.txt","r") as f:

 x=f.readline()

  Queue=int(x)

 valid=validators.url(url)

 if valid!=True:

 await ctx.send("``Please Use A Valid Url``")

 return

 else:

 x=url

 x=x.split("list=")[1]

 x=x.split("&i")[0]

  youtube = build("youtube", "v3", developerKey=DEVELOPER_KEY)

 def get_videos_from_playlist(youtube, items, playlistID):

 response = items.list(part="snippet", playlistId=playlistID)

 while response:

 playlistitems_list_response = response.execute()

 for playlist_item in playlistitems_list_response["items"]:  title =

playlist_item["snippet"]["title"]

 video_id = playlist_item["snippet"]["resourceId"]["videoId"]  yield video_id

 yield title
 response = youtube.playlistItems().list_next(

 response, playlistitems_list_response)

 items = youtube.playlistItems()

 playlist = get_videos_from_playlist(youtube, items,x)

 L=[]

 T=[]

 ctrx=0

 for x in (playlist):

 if ctrx %2!=0:

 title=x

 title=title.replace("'","")

 title=title.replace("]","")

 title=title.replace("[","")

 title=title.replace("-","")

 title=title.replace(")"," ")

 title=title.replace("("," ")
 title=title.replace(" "," ")

 title = unidecode.unidecode(title)

 T.append(title)

 else:

 url="https://www.youtube.com/watch?v="+str(x)

 L.append(url)

 ctrx+=1

 adding= await ctx.send(f"``Adding {len(T)} songs to The Queue Please Hold On``")  with

open ("queue.txt","w") as f:

 queuenew=Queue+1

 f.write(str(queuenew))

 strmusic="insert into music Values"

 for i in range(len(L)):

 if i==len(L)-1:

 strmusic=strmusic +f"('{str(T[i])[:120]}','{str(L[i])}','{author}')"  else:

 strmusic=strmusic +f"('{str(T[i])[:120]}','{str(L[i])}','{author}'),"


 if i ==1:

 with open ("queue.txt","w") as f:

 queuenew=Queue+1
 f.write(str(queuenew))

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord', )

cursor = db.cursor()

 print(strmusic)

 cursor.execute(strmusic)

 db.commit()

 string=f"``Added {len(T)} songs to The Queue``"

 with open ("queue.txt","w") as f:

 queuenew=Queue+len(T)-1

 f.write(str(queuenew))

 await adding.edit(content=string)


#SHUFFLE_THE_SONGS_IN_THE_QUEUE


@bot.command()

async def shuffle(ctx):
 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord', )
cursor = db.cursor()
 cursor.execute("select * from music")
 results = cursor.fetchall()


 x = results.pop(0)
 print(results)
 random.shuffle(results)


 results.insert(0,x)
 print(results)
 strx="insert into music values"
 for i in results:
 strx=strx+str(i)+","
 strx=strx.rstrip(",")



 cursor = db.cursor()
 cursor.execute("delete from music")
 db.commit()
 cursor = db.cursor()
 cursor.execute(strx)
 db.commit()


 await ctx.send("``Queue Shuffled Succesfully``")


#ADDS_THE_SONG_TO_QUEUE_OR_PLAYS_IT_DIRECTLY_IF_NO_SONG_IN_QUEUE


@bot.command(pass_context=True,aliases =["p"])
async def play(ctx,*,url: str):
 await asyncio.sleep(0.2)
 voice = get(bot.voice_clients, guild=ctx.guild)
 author=ctx.author.name
 if voice==None:
 await ctx.send("``Bot needs to be connected to an audio channel to play music (Join using
'.join')``")
 return
 timer["ctx"]=ctx

 if url.lower()=="ramranch" or url=="ram ranch":

 await ctx.send("YOU ARE GAY NOT ME")

 return


 ydl_opts = {

 'format': 'bestaudio/best','ignore_errors':'True',


 'source_address': '0.0.0.0',

 'reconnect_streamed':True, # bind to ipv4 since ipv6 addresses cause issues sometimes

'no_warnings': 'True',

 'noplaylist': True,

 'postprocessors': [{

 'key': 'FFmpegExtractAudio',

 'preferredcodec': 'mp3',

 'preferredquality': '384',

 }],

 }


 await asyncio.sleep(0.2)

 voice = get(bot.voice_clients, guild=ctx.guild)
 print(ctx.guild)

 with open ("queue.txt","r") as f:

 x=f.readline()

 Queue=int(x)

 #print(Queue)

 valid=validators.url(url)

 #print(valid)

 if valid==True and "open.spotify.com/track/" in url:

 print("spotify song")

 with open("config.json", encoding='utf-8-sig') as json_file:

 APIs = json.load(json_file)

 client_credentials_manager = SpotifyClientCredentials(APIs["spotify"]["client_id"],
APIs["spotify"]["client_secret"])

 sp = spotipy.Spotify(client_credentials_manager=client_credentials_manager)  artist =

sp.track(url)
```

```
 url=f"{artist['name']}-{artist['artists'][0]['name']}"
 valid=False
 if valid!=True:
 try:
  x=search_by_keyword(url)[0]
 except IndexError:
 await ctx.send("Sorry Song Not Found Please Use Youtube Link If This Persists")  return
 if timer["errorsearching"]=="Yes":
 timer["errorsearching"]=="No"
 await ctx.send("``API ERROR Please Contact @GamyingOnline#6312 ``")  return


 url="https://www.youtube.com/watch?v="+str(x)


 with open ("queue.txt","w") as f:
 queuenew=Queue+1
 f.write(str(queuenew))
 if ctr[0]!=0:
 Queue+=1
 ctr[0]=ctr[0]+1
 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord', )
cursor = db.cursor()
 db.commit()
 db.close()


 voicestr=str(voice.is_playing())
 print(f"Queue {Queue} ctr {ctr[0]} voice.is_playing {voice.is_playing()}")
ydl_opts = {
 'format': 'bestaudio/best','ignore_errors':'True',


 'source_address': '0.0.0.0',
 'reconnect_streamed':True, # bind to ipv4 since ipv6 addresses cause issues sometimes
'no_warnings': 'True',
 'noplaylist': True,
 'postprocessors': [{
 'key': 'FFmpegExtractAudio',
 'preferredcodec': 'mp3',
 'preferredquality': '320',


 }],
 }
 with youtube_dlc.YoutubeDL(ydl_opts) as ydl:
 print("Streaming audio now\n")
 meta = ydl.extract_info(url, download=False)
 duration=meta['duration']
 title=meta['title']
 title=title.replace("'","")
 title=title.replace("]","")
 title=title.replace("[","")
 title=title.replace("-","")
 title=title.replace(")"," ")
 title=title.replace("("," ")
 title=title.replace(" "," ")
 timer["title"]=title
 idx=meta['id']
 duration=meta['duration']


 if Queue==0 and ctr[0]==0 and voice.is_playing()==False:

 ctr[0]=1
 timer['play']="yes"
 print("working2")
 ydl_opts = {
 'format': 'bestaudio/best','ignore_errors':'True',


 'source_address': '0.0.0.0',
 'reconnect_streamed':True, # bind to ipv4 since ipv6 addresses cause issues sometimes
'no_warnings': 'True',
 'noplaylist': True,
 'postprocessors': [{
 'key': 'FFmpegExtractAudio',
 'preferredcodec': 'mp3',
 'preferredquality': '320',
 }],
 }
 edit=await ctx.send("Getting everything ready now <a:loading:715841171540279306>")  #with
youtube_dlc.YoutubeDL(ydl_opts) as ydl:
 edit2 = await edit.edit(content="Streaming Song Now <a:loading:715841171540279306>")


 with open("id.txt","a")as f:
 strxx=f"{idx} "
  f.write(strxx)


 ytdl = youtube_dlc.YoutubeDL(ydl_opts)
 async with ctx.typing():
 player = await YTDLSource.from_url(url, loop=bot.loop,stream=True)
 ctx.voice_client.play(player, after=lambda e: print('Player error: %s' % e) if e else None)  my_after()


 voice.source.volume = 0.7
 timer['currently_playing']="yes"


 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord',  )
 cursor = db.cursor()
 try:

 cursor.execute(f"insert into music values('{title}','{url}','{author}')")  except:
 cursor.execute(f"insert into music
values('{unidecode.unidecode(title)}','{url}','{unidecode.unidecode(author)}')")
 #cursor.execute(f"insert into queuename values('{title}')")
 db.commit()
 strx=(f"Playing: {title}")
 embed=discord.Embed(title=strx,color=16711680)
 await ctx.send(embed=embed)
 await bot.change_presence(status=discord.Status.online, activity=discord.Game(name=f"Playing:  {title}"))
 await edit.delete()
 Start_time=datetime.datetime.utcnow()
 timer['time_start']=Start_time
 minutes=str(duration//60)
 seconds=str(duration%60)
 if len(str(minutes))==1:
 minutes="0"+minutes
```

```
if len(str(seconds))==1:

seconds="0"+seconds

timer['minutes']=minutes

timer['seconds']=seconds

print("playing\n")


elif Queue==0 and ctr[0]==0 and voice.is_playing()==True:

return


else:


db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord',  )

cursor = db.cursor()

cursor.execute("select count(*) from music")

results = cursor.fetchall()

#print(results)

results=(results[0][0])

#print(results)

if results==0:
return

await ctx.send(f"{title} added to queue at {results}")

db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord',  )

cursor = db.cursor()

title=title.replace("'","")

title=title.replace("]","")

title=title.replace("[","")

title=title.replace("-","")

title=title.replace(")"," ")

title=title.replace("("," ")

title=title.replace(" "," ")

try:


cursor.execute(f"insert into music values('{title}','{url}','{author}')")  except:

cursor.execute(f"insert into music
values('{unidecode.unidecode(title)}','{url}','{unidecode.unidecode(author)}')")


db.commit()
```

```
 if Queue==0 and ctr[0]==0 and voice.is_playing()==True:

 return


#DISPLAYS_A_SCROLLABLE_LIST_OF_ALL_THE_SONGS_IN_QUEUE


@bot.command(aliases=["q"])

async def Queue(ctx,a: int=1):


 if q["id"]!=0 and q["channel"]==ctx.channel and q["guild"]==ctx.guild.id:

 print(id)

 xid=q["id"]

 msg = await ctx.fetch_message(xid)

 await msg.delete()

 q["id"]=0

 elif q["id"]!=0 and q["channel"]!=ctx.channel and q["guild"]!= ctx.guild.id:

 q["channel"]=ctx.channel

 q["guild"]=ctx.guild.id

 m=0
 z=0

 while m!=12412:

 z+=1

 number=10*(a-1)+1


 L=[]

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord', )


 cursor = db.cursor()

 cursor.execute("select count(*) from music")

 results = cursor.fetchall()

 total=(results[0][0])

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord', )

cursor = db.cursor()

 cursor.execute(f"select * from music limit 0,1")

 rem = total%10

 if rem!=0:

 rem = (total)//10+1

 else:
```

```
 rem = total//10

 pagenumber=('Page {}/{} '.format(a,rem))


 results = cursor.fetchall()

 for i in results:

 L.append(i)

 print(L)

 if L==[]:

 await ctx.send("** No Songs In Queue Right Now**")

 return

 first=f"[{(L[0][0])}]({L[0][1]})"


 first = "**"+first+"**" + f" ``Requested by:{L[0][2]}``"

 queuelist=""

 print(L[0])

 if timer["looping"]==True:

 title="Now Playing (CURRENTLY LOOPING)"

 else:
 title="Now Playing"

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord', )

cursor = db.cursor()

 cursor.execute(f"select * from music limit {number},10")

 results = cursor.fetchall()

 for i in results:

  L.append(i)

 print(L)


 for i in range(1,len(L)):

 print(i)

 queuelist = queuelist + f"**{str(number+i-1)}. [{L[i][0]}]({L[i][1]})** ``Requested
By:{L[i][2]}`` \n"

 if queuelist!="":

 first=first+"\n \n"+"**IN QUEUE**"+"\n \n"+queuelist


 embed = discord.Embed(title=title,description=first,color=3800852)


 embed.set_footer(text="Requested By:"+str(ctx.author)+" "+pagenumber,icon_url=ctx.author.avatar_url,)
```

```
#await ctx.send(embed=embed)

 if z==1:

 msg = await ctx.send(embed=embed)

 message=ctx.message

 channel=ctx.message

 await message.delete()

 q["id"]= msg.id

 q["channel"]=msg.channel

 q["guild"]=msg.guild.id


 else:

 if q["id"]!=0 and q["guild"]==ctx.guild.id:

 msg2 = await msg.edit(embed=embed)


 if z==1:

 await msg.add_reaction('<a:left_arrow:712339584796852324>')

 await msg.add_reaction("<a:right_arrow:712339647333793903>")

 def check(reaction, user):
 return user == ctx.author and str(reaction.emoji) == ' ',

 try:

 reaction, user = await bot.wait_for('reaction_add', timeout=40.0, check=check)  except

asyncio.TimeoutError:

 if q["id"]!=0 and q["guild"]==ctx.guild.id:

 #message=ctx.message

 q["channel"]=ctx.channel

 q["guild"]=msg.guild.id

 await msg.delete()

 #await message.delete()

 #await message.delete()

 q["id"]=0

 break


 else:

 if bot.user!=user and user==ctx.author:

 if str(reaction) == "<a:right_arrow:712339647333793903>":  print(rem,"rem")

 if a==rem:

 a=0
 if a<=rem-1:

 a=a+1

 await msg.remove_reaction("<a:right_arrow:712339647333793903>",ctx.author)  elif str(reaction)==

'<a:left_arrow:712339584796852324>':  if a==1:

 a=rem+1

 if a>=2:

 a=a-1

 await msg.remove_reaction('<a:left_arrow:712339584796852324>' ,ctx.author)


#ADDS_SONGS_FROM_SPOTIFY_PLAYLISTS_AND_ALSO_CACHES_IT



@bot.command()

async def sp(ctx,*,url):

 author=ctx.author.name

 tracks = getTracks(str(url))
 #print(tracks)

 print("Searching songs...")

 cont=await ctx.send(f"``Caching Song Approx Time Required:{math.trunc((len(tracks)*1.10+4))} seconds``")

songs = []

 for i in tracks:

 i = i[:120]

 try:

 songs.append(searchYoutubeAlternative(i))

 except:

 i = unidecode.unidecode(i)

 songs.append(searchYoutubeAlternative(i))

 print("Search finished!")

 await cont.edit(content="``Caching Finished Thank You For Your Patience``")



 for i in songs:

 print(i)

 if os.path.isfile(dirx2):

 mode="a"

 else:

 mode="w"
 with open(dirx2,mode,encoding="utf-8") as f:

 for i in songs:

 if i!=None:

 i=i+"\n"

 f.write(i)

 #print("DOne")


 with open(dirx2,"r",encoding="utf-8") as f:

 x=f.readlines()

 L=[]

 for i in x:

 i=i.replace("\n","")

 z=i.split(" ",1 )

 L.append(z)

 #print(L)

 strx="insert into music values"
 for i in L:

 i[1]=i[1].replace("'","")

 i[1]=i[1].replace(")","")

 i[1]=i[1].replace("(","")

 i[1]=i[1].replace('"',"")

 strx=strx+f"('{i[1]}','{i[0]}','{author}'),"

 strx=strx.rstrip(",")

 #print(strx)

 await asyncio.sleep (0.1)

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord', )

cursor = db.cursor()

 #print(strx)

 cursor.execute(strx)

 db.commit()

 await asyncio.sleep (0.05)

 with open ("queue.txt","r") as f:

 x=f.readline()

 Queue=int(x)

 print("Done Adding Songs To The List")

 with open ("queue.txt","w") as f:

 queuenew=Queue+len(L)
 f.write(str(queuenew))

 await ctx.send(f"``{len(L)} songs added to the Queue``")


#DISPLAYS_THE_SONG_CURRENTLY_PLAYING_ALONG_WITH_DURATION_OF_SONG_PLAYED_AND_NEXT_SONG_IN_QUEUE_IF_ANY


@bot.command(aliases=["np"])

async def NowPlaying(ctx):

 if timer['time_start']==0:

 await ctx.send("** No Songs Is Playing Right Now Song Maybe Loading**")

return

 time_now=datetime.datetime.utcnow()

 timer['timer_now']=time_now

 timer_delta=timer['timer_now']-timer['time_start']

 print(timer_delta)

 temp_time=timer_delta.seconds

 minutex=str(temp_time//60)
 secondsx=str(temp_time%60)

 if len(minutex)==1:

 minutex="0"+minutex

 if len(secondsx)==1:

 secondsx="0"+secondsx

 if len(str(timer['seconds']))==0:

 timer['seconds']="00"

 if len(str(timer['minutes']))==0:

 timer['minutes']="00"

 current_time_playing=f" {minutex}:{secondsx} / {timer['minutes']}:{timer['seconds']}"

L=[]

 L2=[]

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord', )

cursor = db.cursor()

 cursor.execute("select * from music limit 1")

 results = cursor.fetchall()


 for i in results: L.append(i)

 print(L)

 if L==[]:

 await ctx.send("** No Songs In Queue Right Now**")
 return
 first=f"**[{L[0][0]}]({L[0][1]})** ``Requested By: {L[0][2]}``"
 if timer["looping"]==True:
 txt="Now Playing (CURRENTLY LOOPING)"
 else:
 txt="Now Playing"
 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord', )
cursor = db.cursor()
 cursor.execute("select * from music limit 1,1")
 results2 = cursor.fetchall()
 for i in results2: L2.append(i)
 if L2==[]:
 str2=""
 else:
 str2=f"**[{L2[0][0]}]({L2[0][1]})** ``Requested By: {L2[0][2]}``"
 embed = discord.Embed(title=txt+current_time_playing,description=first,color=3800852)
 if str2!="":
 embed.add_field(name="Next Song",value=str2)
 embed.set_footer(text="Requested By:"+str(ctx.author),icon_url=ctx.author.avatar_url,)
await ctx.send(embed=embed)


#PAUSES_THE_CURRENTLY_PLAYING_SONG_IF_ANY


@bot.command(pass_context=True, aliases=['pa'])
async def pause(ctx):
 await asyncio.sleep(0.2)
 voice = get(bot.voice_clients, guild=ctx.guild)
 if voice.is_playing():
 voice.pause()
 await ctx.send("Music Paused Use Resume Command To Resume")
 return
 if voice.is_paused():
 await ctx.send("Music Is Already Paused")
 return


#RESUMES_THE_SONG_IF_ANY_SONG_IS_PAUSED
```

```
@bot.command(pass_context=True, aliases=['r'])

async def resume(ctx):

 await asyncio.sleep(0.2)

 voice = get(bot.voice_clients, guild=ctx.guild)

 if voice.is_paused():

 voice.resume()

 await ctx.send("Music Resumed")

 return

 if voice.is_playing():

 await ctx.send("Music Is Already Playing")

 return


#SKIPS_THE_CURRENTLY_PLAYING_SONG


@bot.command(pass_context=True,aliases=["s"])

async def skip(ctx):
 channel=ctx.message.author.voice.channel

 voice = get(bot.voice_clients,guild = ctx.guild)

 if voice==None:

 await ctx.send("Bot Needs To Be Connected To A Channel And Be Playing Something to Stop It!")

return

 if voice.is_playing()!=True:

 await ctx.send("You need to be playing something to stop it!")

 return

 voice.stop()

 timer["skip"]=True

 await ctx.send("Song Skipped")


#LOOPS_THE_CURRENTLY_PLAYING_SONG


@bot.command()

async def loop(ctx):

 voice = get(bot.voice_clients,guild = ctx.guild)

 if voice==None:

 await ctx.send("Bot Needs To Be Connected To A Channel And Be Playing Something to Stop It!")
```

```
return
 if timer["looping"]==True:
 timer["looping"]=False
 await ctx.send("Loop Disabled")
 else:
 timer["looping"]=True
 await ctx.send("Loop Enabled")

#STOPS_THE_SONG_AND_CLEARS_THE_QUEUE

@bot.command(pass_context=True)
async def stop(ctx):
 channel=ctx.message.author.voice.channel
 voice = get(bot.voice_clients,guild = ctx.guild)
 if voice==None:
 await ctx.send("Bot Needs To Be Connected To A Channel And Be Playing Something to Stop It!")
return
 if voice.is_playing()!=True:
 await ctx.send("You need to be playing something to stop it!")
 return
 voice.stop()
 with open ("queue.txt","w") as f:
 f.write("0")
 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord', )
cursor = db.cursor()
 cursor.execute("delete from music ")
 db.commit()
 db.close()
 ctr[0]=0
 timer['minutes']=0
 timer['seconds']=0
 timer['time_start']=0
 timer['timer_now']=0
 for file in os.listdir("./"):
 if file.endswith(".mp3"):
 os.remove(file)
 with open ("id.txt","w") as f:
```

```
 f.write("")

 timer["currently_playing"]="no"

 timer["looping"]=False

 await ctx.send("Song Stopped And Queue Is Cleared")


#LEAVES_THE_VOICE_CHANNEL_AND_CLEARS_THE_QUEUE


@bot.command(pass_context=True)

async def leave(ctx):

 channel=ctx.message.author.voice.channel

 guild=ctx.guild

 voice = get(bot.voice_clients,guild = ctx.guild)

 if voice==None:

 await ctx.send("``Bot needs to be connected to an audio channel to play music (Join using
 '.join')``")

 return

 voice.stop()

 with open ("queue.txt","w") as f:

 f.write("0")
 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord', )

cursor = db.cursor()

 cursor.execute("delete from music ")

 db.commit()

 db.close()

 ctr[0]=0

 timer["looping"]=False

 timer['minutes']=0

 timer['seconds']=0

 timer['time_start']=0

 timer['timer_now']=0

 for file in os.listdir("./"):

 if file.endswith(".mp3"):

 os.remove(file)

 with open ("id.txt","w") as f:

 f.write("")

 timer["currently_playing"]="no"
 await asyncio.sleep(0.1)

 timer["connected"]=False

 try:

 await voice.disconnect()

 channel=ctx.message.author.voice.channel

 time.sleep(1)

 print(voice)

 print(voice.is_connected())

 await ctx.send(f"Bot has left {channel}")

 await bot.change_presence(status=discord.Status.online, activity=discord.Game(name="Discord Bot"))

except AttributeError:

 pass


#PROCESS_TO_CHECK_IF_THE_CURRENT_SONG_PLAYING_HAS_ENDED


async def cplay():

 await bot.wait_until_ready()

 while True :

 if timer['play']=="yes":
 #print("ok")

 await asyncio.sleep(4)

 if timer["connected"]==False:

 await asyncio.sleep(0.5)


 voice = get(bot.voice_clients)

 #print(ctr[0])


 if voice!=None and timer['currently_playing']=="yes" and voice.is_playing()==False and ctr[0]!=0 and
voice.is_paused()==False:

 timer["songl"]=True

 channel=timer["channel"]

 if timer["skip"]==True:

 print("ok")

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031',
database='discord', )

 cursor = db.cursor()

 cursor.execute("delete from music limit 1")

 db.commit()
 db.close()


 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031',
database='discord', )

 cursor = db.cursor()

 cursor.execute("select * from music")

 results = cursor.fetchall()

 #print(f"results {results}")


 if results==[]:

 ctr[0]=0

 print()

 print("CTR = 0")

 await channel.send("> Queue Finished Please Use `.leave` to disconnect the bot or `.p  [songname]` to play
more songs")

 print()

 timer['minutes']=0

 timer['seconds']=0

 timer['time_start']=0
 timer['timer_now']=0

 with open("id.txt","r") as f:

 results=f.readline()

 results=results.split(" ")

 results=results[1:]

 strxx=""

 print(results)

 for j in range(len(results)):

 i=results[j]

 print(i)

 if j==(len(results)-1):

 strxx=strxx+i

 elif i!=" ":

  strxx=strxx+i+" "

 print(strxx,end="")

 print("hello")

 with open ("id.txt","w") as f:

 f.write("")
 with open("id.txt","w") as f:

 f.write(str(strxx))

  print(f"file written {strxx}")

 timer["skip"]=False


 elif timer["looping"]==True:

 timer['minutes']=0

 timer['seconds']=0

 timer['time_start']=0

 timer['timer_now']=0


 else:

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031',
database='discord', )

 cursor = db.cursor()

 cursor.execute("delete from music limit 1")

 db.commit()


 db.close()

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031',
database='discord', )

 cursor = db.cursor()

 cursor.execute("select * from music")

 results = cursor.fetchall()

 print(f"results {results}")


 if results==[]:

 ctr[0]=0

 print()

 print("CTR = 0")

 await channel.send("> Queue Finished Please Use `.leave` to disconnect the bot or `.p  [songname]` to play
more songs")

 print()

 timer['minutes']=0

 timer['seconds']=0

 timer['time_start']=0

 timer['timer_now']=0
 with open("id.txt","r") as f:

 results=f.readline()

 results=results.split(" ")

 results=results[1:]

 strxx=""

 print(results)

 for j in range(len(results)):

 i=results[j]

 print(i)

 if j==(len(results)-1):

 strxx=strxx+i

 elif i!=" ":

 strxx=strxx+i+" "

 print(strxx,end="")

 print("hello")

 with open ("id.txt","w") as f:

 f.write("")

 with open("id.txt","w") as f:

 f.write(str(strxx))

 print(f"file written {strxx}")
 timer['currently_playing']="no"

 timer["songl"]=False


 await asyncio.sleep(0.2)


#REPLACES_THE_POSITION_OF_SONG
 '''

@bot.command()

async def move(ctx,a: int,b: int):

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord', )

cursor = db.cursor()

 cursor.execute(f"select * from music limit {a},1")

 results1 = cursor.fetchall()

 if results1==[]:

 await ctx.send(f"``There is no song at queueposition {a}``")

 return

 #print(results)
 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord', )

cursor = db.cursor()

 cursor.execute(f"select * from music limit {b},1")

 results2 = cursor.fetchall()

 if results2==[]:

 await ctx.send(f"``There is no song at queueposition {b}``")

 return


 print()

 print()

 print()

 print(results1[0])

 print(results2[0])



 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord', )

cursor = db.cursor()

 cursor.execute(f"update music set Song_name='temproryqueuename' where Song_name = (select Song_name from
(select Song_name from music limit {b},1) as t)")

 db.commit()

 db.close()
 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord', )

cursor = db.cursor()

 cursor.execute(f"update music set Song_name='{results2[0][0]}', link = '{results2[0][1]}' ,Requested =
'{results2[0][2]}' where Song_name = (select Song_name from (select Song_name from music limit {a},1) as
t)")

 db.commit()

 db.close()



 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord', )

cursor = db.cursor()

 cursor.execute(f"update music set Song_name='{results1[0][0]}', link = '{results1[0][1]}', Requested =
'{results1[0][2]}' where Song_name = 'temproryqueuename'")

 db.commit()

 db.close()
 await ctx.send(f"``Song Position of:{results1[0][0]} replaced with {results2[0][0]}``"
'''


#REMOVES_THE_SONG_AT_A_SPECIFIED_POSTION


@bot.command()

async def remove(ctx,n :int):

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord', )

cursor = db.cursor()

 cursor.execute(f"delete from music where Song_Name = (select Song_Name from (select Song_Name from music
limit {n},1) as t)")

 db.commit()

 db.close()

 with open ("queue.txt","r") as f:

 x=f.readline()

 Queue=int(x)

 with open ("queue.txt" , "w") as f:

 queuenew=Queue-1

 f.write(str(queuenew))

 await ctx.send(f"`Song Id {n} removed from Queue`")
#PROCESS_TO_PLAY_THE_NEXT_SONG_IN_QUEUE


async def queue2():

 await bot.wait_until_ready()

 await asyncio.sleep(3)


 try:

 while True :

 if timer["songl"]==True:

 await asyncio.sleep(1.5)

 continue


 #if timer['leaving']==True:

 # continue

 if timer['play']=="yes":

 #print("proccessing")

 await asyncio.sleep(6)
 if timer["connected"]==False:

 await asyncio.sleep(1.5)

 continue

 try:

 voice = get(bot.voice_clients)

 except AttributeError:

 pass

 if voice==None:

 continue

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord',  )


 cursor = db.cursor()

 cursor.execute("select count(*) from music")

 results = cursor.fetchall()

 results=(results[0][0])

 #print(results)

 if voice != None:

 if timer["playlist"]==True:
 print("Adding Playlist")

 await asyncio.sleep(0.2)

 continue

 x=timer['currently_playing']

 #print(f"{voice.is_connected()} {voice.is_playing()} timer {x}")

 if results!=0 and voice.is_connected() and voice.is_playing()==False and
voice.is_paused()==False :

 channel=timer["channel"]


 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031',
database='discord', )

 cursor = db.cursor()

 cursor.execute("select * from music")

 results = cursor.fetchall()

 #print(results)

 url=(results[0][1])

 #print(url)

 print("working1")

 song_there = os.path.isfile("song.mp3")
 try:

 if song_there:

 os.remove("song.mp3")

 print("Removed old song file")

 except PermissionError:

 print("Trying to delete song file, but it's being played")  #await

 ctx.send("ERROR: Music playing")

 return

 edit=await channel.send("Getting everything ready now <a:loading:715841171540279306>")  voice =
get(bot.voice_clients,)


 ydl_opts = {

 'format': 'bestaudio/best','ignore_errors':'True',


 'source_address': '0.0.0.0',

 'reconnect_streamed':True, # bind to ipv4 since ipv6 addresses cause issues sometimes  'no_warnings':

 'True',

 'noplaylist': True,

 'postprocessors': [{
 'key': 'FFmpegExtractAudio',

 'preferredcodec': 'mp3',

 'preferredquality': '320',


 }]

 }

 with youtube_dlc.YoutubeDL(ydl_opts) as ydl:

 print("Streaming audio now\n")

 edit2 = await edit.edit(content="Streaming Song now <a:loading:715841171540279306>")  try:


 meta = ydl.extract_info(url, download=False)

 except Exception :

 await edit.delete()

 await channel.send("Error Extracting Music: Check if the youtube video is  accessible")

 await errorqueue()


 continue
 title=meta['title']
 timer["title"]=title
 idx=meta['id']
 duration=meta['duration']
 timer["duration"]=duration
 #ydl.download([url])

 ytdl = youtube_dlc.YoutubeDL(ydl_opts)
 async with channel.typing():
 print(voice.is_playing())
  try:
 player = await YTDLSource.from_url(url, loop=bot.loop,stream=True)  except
youtube_dlc.utils.DownloadError :
 try:
 player = await YTDLSource.from_url(url, loop=bot.loop,stream=True)  except
youtube_dlc.utils.DownloadError:
 await channel.send("Error Extracting Video Please Try Later")  await edit.delete()
 my_after()
 continue

 try:
 ctx = timer["ctx"]
 ctx.voice_client.play(player, after=lambda e: print('Player error: %s' % e) if e  else None)
 my_after()
 voice.source.volume = 0.8

 except Exception as e:
 await edit.delete()
 print(e)
 continue

 timer['currently_playing']="yes"
 ctr[0]=1
 strx=(f"Playing: {title}")
 await edit.delete()
 embed=discord.Embed(title=strx,color=16711680)

 await channel.send(embed=embed)

 await bot.change_presence(status=discord.Status.online,
activity=discord.Game(name=f"Playing: {title}"))

 Start_time=datetime.datetime.utcnow()

 timer['time_start']=Start_time

 minutes=str(duration//60)

 seconds=str(duration%60)

 if len(minutes)==1:

 minutes="0"+minutes

 if len(seconds)==1:

 seconds="0"+seconds

 timer['minutes']=minutes

 timer['seconds']=seconds

 print("playing\n")

 await asyncio.sleep(2.6)

 except Exception as E:

 raise(E)

 pass
#RELOAD_COG_FOR_DEBUGGING_PROCESS


@bot.command()

@commands.is_owner()

async def reload(ctx, cog):

 try:

 bot.reload_extension(f"_cogs3_.{cog}")

 await ctx.send(f"{cog} got reloaded")

 except Exception as E:

 raise E




@reload.error

async def reload_error(ctx,error):

 if isinstance(error,commands.errors.NotOwner):

 await ctx.send("You don't have the permission to execute this command",delete_after=3)

raise error
```

```
#DISPLAYS_THE_LATENCY

#PING

@bot.command(pass_context=True,case_insensitive=True)
async def ping (ctx):
 before = time.monotonic()
 message = await ctx.send("Pong!")
 ping = (time.monotonic() - before) * 1000
 await message.edit(content=f"Pong! `{int(ping)}ms`")
 print(f'Ping {int(ping)}ms')

#LOADING_THE_COGS

for cog in os.listdir(".//_cogs3_"):
 if cog.endswith(".py"):

 try:
 print(cog)
 cog= f"_cogs3_.{cog.replace('.py', '')}"
 print(cog)
 time.sleep(2)
 bot.load_extension(cog)



 except Exception as E:
 print(f"{cog} cannot be loaded")
 raise E
#INITIALIZING_THE_BACKGROUND_TASKS

bot.loop.create_task(queue2())
bot.loop.create_task(cplay())

#RUNNING_THE_BOT

bot.run('NzEzODE4MzgzNzAzMDgxMDEx.XslpLA.PAdqMrTXC8-BNDnXpE4gU56Arhg')
```

ANIMECOMMANDS.PY

```
#BASIC_IMPORTS


from __future__ import
print_function import discord
from discord.ext import commands
import asyncio
from discord import Spotify
import emoji
import mysql.connector
from omdb import OMDBClient
import os
import random
import time
from discord import Member
import string as s
import asyncio
import requests
from jikanpy import Jikan
import binascii
import struct
from PIL import Image
import numpy as np
import scipy
import scipy.misc
import scipy.cluster
imdbx =
OMDBClient(apikey="13fa8e20") jikan
= Jikan()


#DEFINING_DICTIONARY_CLASS_AND_INITIALIZE
class DictLikeClass:
  def __init__(self):
 super(DictLikeClass, self).__init__()
 def __getitem__(self, key):

 return getattr(self, key)


 def __setitem__(self, key, value):

 setattr(self, key, value)

d = DictLikeClass()

d["id"] = 0

d["channel"] = 0

d["guild"]=0


d2 = DictLikeClass()

d2["id"] = 0


d2["channel"] = 0


d2["guild"]=0


#BEGIN_OF_COG


class animecommands(commands.Cog):

 def __init__(self,bot):

 self.bot = bot


 #HELP_COMMAND_TO_DM_LIST_OF_ALL_THE_COMMANDS


 @commands.command()

 async def help(self,ctx):


 embed = discord.Embed(title="Help",description="Help For Persona Bot",color=3800852)

embed.add_field(name="1. .join" , value="> ``Joins the vc user is currently in``",inline=False)

embed.add_field(name="2. .leave" , value="> ``Leaves the vc an clears the queue``",inline=False)

 embed.add_field(name="3. .play(.p) [url/song_name]", value="> ``Plays the given song using
youtube``",inline=False)
 embed.add_field(name="4. .pl [yt_playlist_url]" , value="> ``Gets the playlist using youtube
url``",inline=False)

 embed.add_field(name="5. .sp [spotify_playlist_url]" , value="> ``Gets the playlist using spotify
url``",inline=False)
```

```
 embed.add_field(name="6. .pause" , value="> ``Pauses the song currently playing``",inline=False)

embed.add_field(name="7. .resume" , value="> ``Resumes the song currently playing``",inline=False)

embed.add_field(name="8. .skip" , value="> ``Skips the song currently playing``",inline=False)

 embed.add_field(name="9. .move [song_position_1 song_position_2" , value="> ``Swaps the position of  the
two songs given ``",inline=False)

 embed.add_field(name="10. .shuffle" , value="> ``Shuffles the playlist``",inline=False)

embed.add_field(name="11. .loop" , value="> ``Toggles the song loop``",inline=False)

 embed.add_field(name="12. .nowplaying (.np)" , value="> ``Shows the status of the song currently
playing``",inline=False)

 embed.add_field(name="13. .queue(.q) [queue_page_number (writing nothing shows first page)]" ,
value="> ``Shows the list of songs in queue`",inline=False)

 embed.add_field(name="14. .stop" , value="> ``Stops the current song playing and removes all the  songs
from the queue``",inline=False)


 embed.set_author(name="Persona Bot",icon_url=self.bot.user.avatar_url)

 embed.set_footer(text="Requested By:"+str(ctx.author),icon_url=ctx.author.avatar_url,)

await ctx.author.send(embed=embed)

 await ctx.send("Help Commands Sent In DM ")


 #GETTING_DATA_OF_MOVIE/SERIES_FROM_IMDB

 @commands.command()

 async def imdb(self,ctx,*,strx:str):

 print(strx)

 movie=imdbx.get(title=strx)

 #await ctx.send(movie['title'])

 #await ctx.send(movie['imdb_rating'])

 #await ctx.send(movie['plot'])

 #await ctx.send(movie['year'])

 #await ctx.send(movie['poster'])

 typex=movie['type'].title()


 embed=discord.Embed(title=f"{movie['title']}({typex})",color=16711708)

 embed.set_thumbnail(url=movie['poster'])

 embed.add_field(name="Rating", value=movie['imdb_rating'], inline=True)

 embed.add_field(name="Release Date", value=movie['released'], inline=True)

embed.add_field(name="Length", value=movie['runtime'], inline=True)

 embed.add_field(name="Director(s)", value=movie['director'], inline=False)
embed.add_field(name="Actors", value=movie['actors'] , inline=False)
 embed.add_field(name="Plot", value=movie['plot'], inline=False)
 await ctx.send(embed=embed)




 #ADDING_ALIAS_TO_WALLPAPER_FOLDERS
 @commands.command()
 async def addalias(self,ctx,*,rep):
 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord')
cursor = db.cursor()
 cursor.execute("select name,alias from list")
 results = cursor.fetchall()
 Z=[]
 for i in range(len(results)):
 #print(results[i][0],results[i][1])
 if rep.lower() in str(results[i][0]).lower() or rep in str(results[i][1]).lower():
Z.append(results[i])
 print(Z)
 if len(Z)>1:
 msg=await ctx.send("Please Be More Specific")
 await asyncio.sleep(4)
 await msg.delete()
 return
 strx="update list set alias= "
 print(Z)
 if Z[0][1]==None:
 #al=input("Enter Alias")
 await ctx.send(f"Enter the alias for: {Z[0][0]} or type exit to exit")
print("entering alias")
 def check(m):
 return m.author == ctx.message.author and m.channel == ctx.message.channel  msg = await
self.bot.wait_for('message',check=check,timeout=30)
 al=str(msg.content)
 if al.lower=="exit":
 await ctx.send("Exited Succesfully")
 return
 print(al)
 strx+=f"'{al}' where name='{Z[0][0]}'"
 else:
 #al=input("Enter Alias")
 await ctx.send(f"Enter the alias for: {Z[0][0]} or type exit to exit")  def
check(m):
 return m.author== ctx.message.author and m.channel == ctx.message.channel  msg = await
self.bot.wait_for('message',check=check,timeout=30)
 al=str(msg.content)

 if al.lower=="exit":
 await ctx.send("Exited Succesfully")
 return
 print("message received")
 print(al)
 al=al+" "+Z[0][1]
 strx+=f"'{al}' where name='{Z[0][0]}'"
 print(strx)
 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord')
cursor = db.cursor()
 cursor.execute(strx)
 await ctx.send(f"Alias {al} added to {Z[0][0]} ")
 db.commit()
 db.close()

 #WALLPAPER_COMMAND

 @commands.command(aliases=['a'])
 async def anime(self,ctx,*,rep=None):

 if rep!=None:
 if len(rep)<3:
 await ctx.send("You just wanna break my bot don't you")
 return
 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord')
 cursor = db.cursor()
 cursor.execute("select name,alias from list")

 results = cursor.fetchall()

 L=[]

 Repeat=[]


  try:

 if rep==None:

 await ctx.send("`The Correct Syntax Is $anime [animename]`")  return



 fetching=await ctx.send("**Fetching The Wallpaper** <a:loading:715841171540279306> ")  if

rep=="random" or rep=="Random":

 lenl=len(results)

 rand=random.randint(0,lenl)

 rep=results[rand-1][0]

 print(rep,"rep")

 for i in range(len(results)):

 if str(results[i][1])!="None":

 x=str(results[i][0])+"__"+(results[i][1])

 y=x.lower()

 z=str(results[i][0]).lower()

 else:

  x=str(results[i][0])

 y=x.lower()

 z=y



 if str(rep).lower() in y :

 print(x,str(rep))

 rep=x.split("__")[0]

 record=i

 print(record,"record")

 print(rep)

 z=z.split("__")[0]

 Repeat.append(z)
 #print("Hello")

 print(Repeat)
 if len(Repeat)>1:

 repstr=""


 for i in range(len(Repeat)):

 zenn=Repeat[i]

 zenn=s.capwords(zenn)

 repstr=repstr+" ** "+ str(i+1)+". "+zenn+" **\n \n"

 embed = discord.Embed(title=" There Are 2 Or More Anime With The Same Word In It \nPlease  Give the Number
Along the Anime Name which you wanted to specify: \n",description=repstr, color=16711708)

 embed.set_author(name=self.bot.user.name,icon_url=self.bot.user.avatar_url)


 msg2=await ctx.send(embed=embed)

 #await ctx.send(repstr)

 channel = ctx.channel

 print(channel)

 def check(m):

 return m.content.isnumeric() and m.channel == channel and ctx.author == m.author  msg = await

self.bot.wait_for('message',check=check,)


 x=(msg.content)

 print(x)

 recordx=Repeat[int(x)-1]

 print(recordx)

 #recordx=str(L[record])

 dirx=r"C:\\Users\\aakas\\Desktop\\Discord Wallpaper"+"\\" + str(recordx)   onlyfiles =

next(os.walk(dirx))[2]

 x=(len(onlyfiles))

 num=random.randint(1,x)

 name=str(num)+".jpg"

 dirx=dirx+"/"+str(num)+".jpg"


 colour = "07c4e1"

 msg3=(f"{s.capwords(recordx)}")

 file = discord.File(dirx, filename=name)
 embed = discord.Embed(title=msg3,color=int(colour,16))

 embed.set_footer(text="Requested By:"+str(ctx.author),icon_url=ctx.author.avatar_url,)

embed.set_image(url=f"attachment://{name}")
```

```
 await ctx.send(file=file, embed=embed)

 await fetching.delete()

 await msg2.delete()

 print(rep)



 elif len(Repeat)==1:

 print(x)

 print(L)

 print(record)

 rep=rep.split("__")[0]

 print("Rep:",rep)


 #recordx=str(L[record])

 dirx=r"C:\\Users\\aakas\\Desktop\\Discord Wallpaper"+"\\" + rep  print(dirx)

 onlyfiles = next(os.walk(dirx))[2]

 x=(len(onlyfiles))

 num=random.randint(1,x)

 name=str(num)+".jpg"

 dirx=dirx+"/"+str(num)+".jpg"

 print(dirx)

 try:


 NUM_CLUSTERS = 3

 im = Image.open(dirx)

 im = im.resize((350, 350))

 ar = np.asarray(im)

 shape = ar.shape

 ar = ar.reshape(scipy.product(shape[:2]), shape[2]).astype(float)  codes, dist =

scipy.cluster.vq.kmeans(ar, NUM_CLUSTERS)

 vecs, dist = scipy.cluster.vq.vq(ar, codes)

 counts, bins = scipy.histogram(vecs, len(codes))
 index_max = scipy.argmax(counts)

 peak = codes[index_max]

 colour = binascii.hexlify(bytearray(int(c) for c in peak)).decode('ascii')  if

int(colour,16)>16777215:
 colour = "07c4e1"

 except:

 colour="07c4e1"

 msg3=(f"{s.capwords(rep)}")

 file = discord.File(dirx, filename=name,)

  embed = discord.Embed(title=msg3,color=int(colour,16))

 embed.set_footer(text="Requested By:"+str(ctx.author),icon_url=ctx.author.avatar_url,)

embed.set_image(url=f"attachment://{name}")

 await ctx.send(file=file, embed=embed)

 await fetching.delete()

 print(rep)

 elif rep==None:

 ctx.send("The Correct Syntax Is $anime [animename]")

 else:

 print("Hello")

 print(rep)

 request= str(ctx.guild)+" : #"+str(ctx.channel)+" : "+str(ctx.author)+" --> "+str(rep)

print(request)

 with open('animerequest.txt','a',encoding="utf-8") as p:

 p.write(request)

 p.write("\n")

 p.close()

 await fetching.delete()

 await ctx.send("Anime Doesn't Exist \nThis Anime Will Be Added Soon...")



 except Exception as E:

 raise E


 #SYNC_THE_DATABASE_TO_ALL_THE_FOLDERS_FOR_WALLPAPER


 @commands.command()
 async def sync(self,ctx):

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord')  L=[]

 lis=[]

 cursor = db.cursor()
 cursor.execute("select name from list")

 results = cursor.fetchall()

 start=len(results)

 for i in results:

 L.append(i[0])

 strx="insert into list(sno,name) values "

 print(L)


 for i,j,y in os.walk(r"C:\\Users\\aakas\\Desktop\\Discord Wallpaper\\"):

 print(j)

 if j!=[]:

 lis=j

 print("Hello")

 print(lis)

 for i in lis:

  if i not in L:

 print(i)

 start+=1

 strx+=f"({start},'{i}'),"

 print(strx.rstrip(","))

 if strx!="insert into list(sno,name) values ":

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord')  cursor =

db.cursor()

 cursor.execute(strx.rstrip(","))

 db.commit()


 #LIST_OF_ALL_THE_ANIME_IN_THE_DATABASE_IN_A_SCROLLABLE_FORMAT

 @commands.command(aliases=['al'])

 async def animelist(self,ctx,a :int=1):

 print("starting")

 if d["id"]!=0 and d["channel"]==ctx.channel and d["guild"]==ctx.guild.id:

 print(id)
 xid=d["id"]

 msg = await ctx.fetch_message(xid)

 await msg.delete()

 d["id"]=0
 elif d["id"]!=0 and d["channel"]!=ctx.channel and d["guild"]!= ctx.guild.id:

d["channel"]=ctx.channel

 d["guild"]=ctx.guild.id

 z=0

 m=0

 while m!=123456:

 z+=1

 print(ctx.channel)

 page = (10*(a-1))+1

 b=a

 NO=[]

 NAME=[]

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031', database='discord',  )

 cursor = db.cursor()

 cursor.execute("select * from list order by name asc")

 results = cursor.fetchall()

 k=1

 for x in results:

 print(x)

 NO.append(k)

 NAME.append(x[1])

 k+=1


 num = len(NO)

 print(num)

 print(len(NAME))




 strxx=""

 rem = len(NO)%10

 if rem!=0:

 rem = len(NO)//10+1

 else:
 rem = len(NO)//10

 pagenumber=(' Page {}/{}'.format(a,rem))

 print(page+9)

 print(len(NO)+1)
 if (page+9) >= (len(NO)):


 for i in range(page-1,(len(NO))):


 repx=NAME[i]


 dirx=r"C:\\Users\\aakas\\Desktop\\Discord Wallpaper\\" + str(repx)

 onlyfiles = next(os.walk(dirx))[2]


 x=(len(onlyfiles))

 strxx=strxx+"\n \n"+str(NO[i])+"--->"+str(NAME[i]+"["+str(x)+"]")  else:

 for i in range(page-1,(page+9)):

 repx=NAME[i]


 dirx=r"C:\\Users\\aakas\\Desktop\\Discord Wallpaper\\" + str(repx) +"/"


 onlyfiles = next(os.walk(dirx))[2]

 #print(onlyfiles)


 x=(len(onlyfiles))

 strxx=strxx+"\n \n"+str(NO[i])+"--->"+str(NAME[i]+"["+str(x)+"]")  embed =

discord.Embed(title="Anime List",description=strxx, color=16711708)

embed.set_author(name=self.bot.user.name,icon_url=self.bot.user.avatar_url)

embed.set_footer(text=pagenumber,icon_url=ctx.author.avatar_url,)


 if z==1:

 msg = await ctx.send(embed=embed)

 message=ctx.message

 channel=ctx.message

 await message.delete()

 d["id"]= msg.id
 d["channel"]=msg.channel

 d["guild"]=msg.guild.id


 else:

 if d["id"]!=0 and d["guild"]==ctx.guild.id:

 msg2 = await msg.edit(embed=embed)
 if z==1:
 await msg.add_reaction('<a:left_arrow:712339584796852324>')
 await msg.add_reaction("<a:right_arrow:712339647333793903>")
 def check(reaction, user):
 return user == ctx.author and str(reaction.emoji) == ' ',
 try:
 reaction, user = await self.bot.wait_for('reaction_add', timeout=20.0, check=check)  except
asyncio.TimeoutError:
 if d["id"]!=0 and d["guild"]==ctx.guild.id:
 #message=ctx.message
 d["channel"]=ctx.channel
 d["guild"]=msg.guild.id
 await msg.delete()
 #await message.delete()
 #await message.delete()
 d["id"]=0
 break

 else:
 if self.bot.user!=user and user==ctx.author:
 if str(reaction) == "<a:right_arrow:712339647333793903>":
 print(rem,"rem")
 if a==rem:
 a=0
 if a<=rem-1:
 a=a+1
 await msg.remove_reaction("<a:right_arrow:712339647333793903>",ctx.author)  elif str(reaction)==
'<a:left_arrow:712339584796852324>':  if a==1:
 a=rem+1
 if a>=2:
 a=a-1
 await msg.remove_reaction('<a:left_arrow:712339584796852324>' ,ctx.author)

#GETS_INFO_ABOUT_AN_ANIME_WITH_SCROLLABLE_INFO_ABOUT_SEQUAL_AND_PREQUEL
```

```
 @commands.command()

 async def manime(self,ctx,*,rep):


 print("it works")

 if d2["id"]!=0 and d2["channel"]==ctx.channel and d2["guild"]==ctx.guild.id:

xid=d2["id"]

 msg = await ctx.fetch_message(xid)

 await msg.delete()

 d2["id"]=0

 elif d2["id"]!=0 and d2["channel"]!=ctx.channel and d2["guild"]!= ctx.guild.id:

d2["channel"]=ctx.channel

 d2["guild"]=ctx.guild.id

 msg=await ctx.send("**Loading** <a:loading:715841171540279306>")

 z=0

 L1=(jikan.search('anime',rep))

 id=(L1["results"][0]["mal_id"])

 id1=id

 id2=id

 while True:

 z+=1

 L2=(jikan.anime(id))

 rating=L2['rating']

 title=(L2['title'])

 status=L2['status']

 genres2=L2['genres']

 if 'Prequel' in dict.keys(L2['related']):

 prequel=L2['related']['Prequel'][0]['mal_id']

 Prequel=L2['related']['Prequel'][0]['name']

 else:

 prequel=None
 if 'Sequel' in dict.keys(L2['related']):

 sequel=L2['related']['Sequel'][0]['mal_id']

 Sequel=L2['related']['Sequel'][0]['name']

 else:

 sequel=None

 genresx=""
 for i in genres2:
 cat=(i['name'])
 if i['name']!=genres2[len(genres2)-1]['name']:
 genresx += cat + " **|** "
 else:
 genresx += cat
 synopsis=L2['synopsis'][:346]+"... Read More On The MAL SITE"

 if L2['airing']==True:
 airing="airing"
 else:
 airing="not airing"
 url=L2['url']
 episodes=L2['episodes']
 type=L2['type']
 image_url=L2['image_url']
 duration=L2["duration"]
 score=L2['score']
 rank=L2['rank']
 trailer_url=L2['trailer_url']
 strx="[{}]({})".format("Watch The trailer",trailer_url)  embed =
discord.Embed(title=title,url=url,color=0xeee657)
embed.set_thumbnail(url=image_url)
 embed.add_field(name="Type", value=type, inline=True)
embed.add_field(name="Episodes", value=episodes, inline=True)
embed.add_field(name="Status", value=status, inline=True)
embed.add_field(name="Episodes Duration", value=duration, inline=True)
embed.add_field(name="Score", value=score, inline=True)
embed.add_field(name="Rank", value=rank, inline=True)   embed.add_field(name="Age
Rating", value=rating, inline=False)
 embed.add_field(name="Genres", value=genresx, inline=True)
 embed.add_field(name="Description", value=synopsis, inline=False)
 if prequel:
 embed.add_field(name="Prequel", value=Prequel, inline=True)
 if sequel:
```

```
 embed.add_field(name="Sequel", value=Sequel, inline=True)

 embed.add_field(name="Trailer", value=strx, inline=False)

 embed.set_author(name=self.bot.user.name,icon_url=self.bot.user.avatar_url)

embed.set_footer(text="Requested By:"+str(ctx.author),icon_url=ctx.author.avatar_url,)


 if z==1:

 msg4 = await ctx.send(embed=embed)

 embed2=embed

 #print(msg4)

 await msg.delete()

 message=ctx.message

 await message.delete()

 d2["id"]= msg4.id

 d2["channel"]=msg4.channel

 d2["guild"]=msg4.guild.id

 else:

 if editx=="yes":

 print(prequel)

 if d2["id"]!=0 and d2["guild"]==ctx.guild.id:

 msgxx=await msg4.edit(embed=embed)


 if z==1:

 editx="no"

 await msg4.add_reaction('<a:left_arrow:712339584796852324>')

 await msg4.add_reaction("<a:right_arrow:712339647333793903>")

 def check(reaction, user):

 return user == ctx.author and (str(reaction.emoji) == '<a:left_arrow:712339584796852324>' or
str(reaction.emoji)== '<a:right_arrow:712339647333793903>')

 reaction, user = await self.bot.wait_for('reaction_add',check=check)

 if self.bot.user!=user and user==ctx.author:

 if str(reaction) == '<a:right_arrow:712339647333793903>':

 if sequel!=None:

 if id!= sequel:
 id = sequel

 editx="yes"

 else:

 editx="no"
 await msg4.remove_reaction('<a:right_arrow:712339647333793903>',ctx.author)

 elif str(reaction)== '<a:left_arrow:712339584796852324>':

 if prequel!=None:
 if id!=prequel:
 id = prequel
 editx="yes"
 else:
  editx="no"

 await msg4.remove_reaction('<a:left_arrow:712339584796852324>' ,ctx.author)

 @animelist.error
 async def animelist_error(self, ctx, error):
 if isinstance(error, commands.BadArgument):
 await ctx.send("You need to enter a integer after `$animelist`")

#DEFINING_THE_COG

def setup(bot):
 bot.add_cog(animecommands(bot))
```

PROJECT.PY

```
#BASIC_IMPORTS
import discord
from discord.ext import commands
import asyncio
import emoji
import mysql.connector
import random
import time
```

```
from discord import Member

import string as s

import asyncio

from jikanpy import Jikan

from wallhaven import Wallhaven

import os


prejectx={}

prejectx['channel']=None

class project(commands.Cog):

 def __init__(self,bot):

 self.bot = bot




 @commands.command()

 async def project(self,ctx):

                  channel=ctx.message.channel

prejectx['channel']=channel

prejectx['author']=ctx.message.author


 #MENU_FOR_THE_PROJECT


 async def menu():

 channel=prejectx['channel']
 c="y"

 while (c=="Y" or c=="y"):

 ch="n"

 menustr='''``` MENU\n========================\n1. Access Level Records\n2.  Enter New Records\n3.
Update Records\n4. Delete Records\n5. Export Records\n6. Exit```'''

 msg1=await channel.send(menustr)

 await msg1.add_reaction("1️⃣")

 await msg1.add_reaction("2️⃣")

 await msg1.add_reaction("3️⃣")

 await msg1.add_reaction("4️⃣")

 await msg1.add_reaction("5️⃣")

 await msg1.add_reaction("6️⃣")

 emoji=['1️⃣','2️⃣','3️⃣','4️⃣','5️⃣','6️⃣']
```

```
 def check(reaction, user):
 return user == ctx.author and (str(reaction.emoji) in emoji)  try:
 reaction, user = await self.bot.wait_for('reaction_add', timeout=20.0,check=check)  except
asyncio.TimeoutError:
 await channel.send("You took too long to respond")
 return
 if self.bot.user!=user and user==ctx.author:
 if str(reaction)==emoji[0]:
 print(1)
 await msg1.delete()
 await submenu()
 # await msg1.delete()

 await menustring.delete()
 await smenu.delete()



 elif str(reaction)==emoji[1]:
 await adddata()
 #await msg1.delete()
  await msg1.delete()
 await menustring.delete()

 print(2)
 elif str(reaction)==emoji[2]:
 await updatedata()
 await msg1.delete()
 await menustring.delete()

 print(3)
 elif str(reaction)==emoji[3]:
 await deldata()
 await msg1.delete()
 await menustring.delete()

 print(4)
 elif str(reaction)==emoji[4]:
```

```
 await exportdata()

 await msg1.delete()

 await menustring.delete()


 print(5)

 elif str(reaction)==emoji[5]:

 print(6)

 exiting=await channel.send("`Exiting.... Thank For Using The Program`")


 await msg1.delete()

 await menustring.delete()

 await exiting.delete()

 break


 return

 ch="n"


 #DEFINING_DELETE_DATA_PROCESS


 async def deldata():

 channel=prejectx['channel']

 author=prejectx['author']

 def check(m):
 return m.author==author and m.channel == channel

 while True:

 await channel.send("`Enter the PID of the record you want to delete:`")  msg = await

self.bot.wait_for('message', check=check)

 pid=msg.content

 #pid=int(input("Enter the PID of the record you want to delete:"))  import

mysql.connector

 try:

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031',
database='CriminalRecords', )

 cursor = db.cursor()

 cursor.execute("SELECT * FROM Information where PID="+str(pid))  results =

cursor.fetchall()

 if results!=[]:
```

```
 await channel.send(results[0])


 except Exception as e:

 if str(e)=="1146 (42S02): Table 'criminalrecords.information' doesn't exist":  import

mysql.connector

 try:

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031',  database='CriminalRecords',)

 cursor = db.cursor()

 cursor.execute("create table information (PID integer primary key,Name  varchar(20),EntryDate
date,ExitDate date,PLevel integer,Contact varchar(20))" )

 await channel.send("`Table Information Does Not Exist.....`")  time.sleep(.75)

  await channel.send("`Creating Table Information.....`")  time.sleep(.75)

 await channel.send("`Table Created`")

 db.commit()

 break

 except Exception as e:

 print(e)

 break

 else:

 await channel.send ("PID was incorrect")

 break
 if results ==[]:

 await channel.send("`No Record With Such PID Exists`")

 break

 await channel.send("`Are you sure you want to delete the record (Y/N):`")  msg = await

self.bot.wait_for('message', check=check)

 ans=msg.content

 #ans=input("Are you sure you want to delete the record (Y/N):")  if ans=="Y" or

ans == "y":

 import mysql.connector

 try:

 db=mysql.connector.connect(host='localhost',user='root',password='aak20f031',
database='CriminalRecords', )

 cursor = db.cursor()

 abc="Delete from information where PID="+str(pid)

 cursor.execute(abc)

 db.commit()
```

```
strx="`Record with PID "+str(pid)+" deleted succesfully`"  await channel.send(strx)

break

except Exception as e:

raise e


#DEFINING_EXPORT_DATA_PROCESS


async def exportdata():

channel=prejectx['channel']

author=prejectx['author']

def check(m):

return m.author==author and m.channel == channel

no=0

with open("Record.txt", "a") as f:

f.truncate(0)

f.close()

abc=""

import datetime

import mysql.connector

try:
db=mysql.connector.connect(host="localhost",user="root",password='aak20f031',
database='CriminalRecords',)

cursor = db.cursor()

cursor.execute("SELECT * FROM Information " )

results = cursor.fetchall()

for x in results:

for i in x:

abc=abc+str(i)+"/"

no+=1

with open("Record.txt", "a") as f:

f.write(abc)

f.write("\n")

f.close()

abc=""

strx=f"`{no} Records Exported Succesfully`"


await channel.send(strx)
```

```
 dirx=r"C:\Users\aakas\Desktop\Persona 11-12\Persona Bot\Record.txt"  await
ctx.send(file=discord.File(dirx))
 except Exception as e:
 if str(e)=="1146 (42S02): Table 'criminalrecords.information' doesn't exist":  import
mysql.connector
 try:
 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031',
database='CriminalRecords',)
 cursor = db.cursor()
 cursor.execute("create table information (PID integer primary key,Name  varchar(20),EntryDate
date,ExitDate date,PLevel integer,Contact varchar(20))" )
 await channel.send("Table Information Does Not Exist.....")  time.sleep(.75)
 await channel.send("Creating Table Information.....")  time.sleep(.75)
 await channel.send("Table Created")
 db.commit()
 except Exception as e:
 print(e)
 else:
 await channel.send("Error: unable to fetch data")
 #DEFINING_UPDATE_DATA_PROCESS


 async def updatedata():
 channel=prejectx['channel']
 author=prejectx['author']
 def check(m):
 return m.author==author and m.channel == channel
 ch=await channel.send("Enter The Id Of The Record You Want To Change:")  msg =
await self.bot.wait_for('message', check=check)
 pid=msg.content
 import mysql.connector
 try:
 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031',
database='CriminalRecords', )
 cursor = db.cursor()
 cursor.execute("SELECT * FROM information where PID="+str(pid ))
 results = cursor.fetchall()
 Data=[]
 for x in results:
```

```
 Data.append(x)

 if Data==[]:

 await channel.send("No Records Available")

 await ch.delete()

 return


 else:

 await channel.send(Data)

 except Exception as e:

 if str(e)=="1146 (42S02): Table 'criminalrecords.information' doesn't exist":   import
mysql.connector

 try:

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031',  database='CriminalRecords',)

 cursor = db.cursor()

 cursor.execute("create table information (PID integer primary key,Name  varchar(20),EntryDate
date,ExitDate date,PLevel integer,Contact varchar(20))" )

 await channel.send("Table Information Does Not Exist.....")
  time.sleep(.75)

 await channel.send("Creating Table Information.....")  time.sleep(.75)

 await channel.send("Table Created")

 db.commit()

 except Exception as a:

 print(a)


 else:

 print(e)

 strx='''```Which Field Would You Like To Change:\n1. Name\n2. Entry Date:\n3. Exit Date:\n4.  PLevel\n5.
Contact\n6. Exit```'''

 msgfield=await channel.send(strx)

 await msgfield.add_reaction("1️⃣")

 await msgfield.add_reaction("2️⃣")

 await msgfield.add_reaction("3️⃣")

 await msgfield.add_reaction("4️⃣")

 await msgfield.add_reaction("5️⃣")

 await msgfield.add_reaction("6️⃣")

 emoji=['1️⃣','2️⃣','3️⃣','4️⃣','5️⃣','6️⃣']

 def check(reaction, user):
```

```
 return user == ctx.author and (str(reaction.emoji) in emoji)
 try:
 reaction, user = await self.bot.wait_for('reaction_add', timeout=20.0,check=check)  except
asyncio.TimeoutError:
 await channel.send("You took too long to respond")
 return
 if self.bot.user!=user and user==ctx.author:
 if str(reaction)==emoji[0]:
 choice=1
 elif str(reaction)==emoji[1]:
 choice=2
 elif str(reaction)==emoji[2]:
 choice=3
 elif str(reaction)==emoji[3]:
 choice=4
 elif str(reaction)==emoji[4]:
 choice=5
 elif str(reaction)==emoji[5]:
 choice=6
 choice=int(choice)
 print(choice)
 def check2(m):
 return m.author == ctx.message.author and m.channel == ctx.message.channel  if choice==1:
 await channel.send("Enter The New Name For The Field:")  msg = await
self.bot.wait_for('message', check=check2)  new=str(msg.content)
 string='UPDATE INFORMATION SET NAME="'+new+'" where PID='+str(pid)  elif
choice==2:
 await channel.send("Enter The New Entry Date For The Field (yyyy-mm-dd) :")  msg = await
self.bot.wait_for('message', check=check2)  new=str(msg.content)
 string='UPDATE INFORMATION SET EntryDate="'+new+'" where PID='+str(pid)  elif
choice==3:
 await channel.send("Enter The New Exit Date For The Field (yyyy-mm-dd) :")  msg = await
self.bot.wait_for('message', check=check2)  new=str(msg.content)
 string='UPDATE INFORMATION SET ExitDate="'+new+'" where PID='+str(pid)  elif
 choice==4:
 await channel.send("Enter The New PLevel For The Field:")  msg = await
```

```
self.bot.wait_for('message', check=check2)  new=str(msg.content)

 string='UPDATE INFORMATION SET PLevel="'+str(new)+'" where PID='+str(pid)  elif

choice==5:

 await channel.send("Enter The New Contact For The Field:")  msg = await

self.bot.wait_for('message', check=check2)  new=str(msg.content)

 string='UPDATE INFORMATION SET Contact="'+new+'" where PID='+str(pid)  elif

choice==6:

 msgfield.delete()

 return

 print("hello")

 print(string)
 import mysql.connector

 try:

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031',
database='CriminalRecords',)

 cursor = db.cursor()

 cursor.execute(string)

 db.commit()

 await channel.send("Record Updated Succesfully")

 except Exception as e:

 print(e)


 #DEFINING_ADD_DATA_PROCESS


 async def adddata():

 channel=prejectx['channel']

 author=prejectx['author']

 def check(m):

 return m.author==author and m.channel == channel  abc="INSERT INTO

information Values ("

 await channel.send("`Please Type The PID`")

 msg = await self.bot.wait_for('message', check=check)

 PID=msg.content

 abc=abc+str(PID)+","

  await channel.send("`Please Type The Name`")

 msg = await self.bot.wait_for('message', check=check)

 Name=msg.content
 abc=abc+"'"+Name+"'"+","

 await channel.send("`Enter The Date of Entry (yyyy-mm-dd)`")  msg =

await self.bot.wait_for('message', check=check)

 EDate=msg.content

 abc=abc+"'"+EDate+"'"+","

 await channel.send("`Enter The Date of Exit (yyyy-mm-dd)`")  msg =

await self.bot.wait_for('message', check=check)

 ExDate=msg.content

 abc=abc+"'"+ExDate+"'"+","

 await channel.send("`Enter The PLevel`")

 msg = await self.bot.wait_for('message', check=check)

 PLvl=msg.content
 abc=abc+str(PLvl)+","

 await channel.send("`Enter The Contact Information`")

 msg = await self.bot.wait_for('message', check=check)

 Contact=msg.content

 abc=abc+"'"+Contact+"'"+")"

 import mysql.connector

 try:

 db=mysql.connector.connect(host='localhost',user='root',password='aak20f031',
database='CriminalRecords' , )

 cursor = db.cursor()

 cursor.execute(abc)

 db.commit()

 await channel.send("Record Added Succesfully")

 except Exception as e:

 if str(e)=="1146 (42S02): Table 'criminalrecords.information' doesn't exist":   import

mysql.connector

 try:


db=mysql.connector.connect(host="localhost",user="root",password='aak20f031',

database='CriminalRecords')  cursor = db.cursor()

                                     cursor.execute("create table information (PID integer primary key,Name
varchar(20),EntryDate date,ExitDate date,PLevel integer,Contact varchar(20))" )

          await channel.send("Table Information Does Not Exist.....") time.sleep(.75)

                                    await channel.send("Creating Table Information.....")

 time.sleep(.75)

                                    await channel.send("Table Created")
                                    db.commit()

 except Exception as a:

 print(a)


 else:

 print(e)


 #DEFINING_FETCH_SPECIFIC_DATA_PROCESS


 async def fetchspecificdata():

 channel=prejectx['channel']
 author=prejectx['author']

 def check(m):

 return m.author==author and m.channel == channel

 acc=await channel.send("`Please Type The PID Of The Record You Want To Access`")  msg =

await self.bot.wait_for('message', check=check)

 pid=int(msg.content)

 try:

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031',
database='CriminalRecords', )

 cursor = db.cursor()

 abcd="SELECT PLevel FROM information where PID="+str(pid)

 cursor.execute(abcd)

 results = cursor.fetchall()

 Data=[]

 if results==[]:

 await channel.send("`No Records Available For This PID`")

 await acc.delete()

 return

 for x in results:

 if x==1:

 cursor = db.cursor()

 abcd="SELECT * FROM information where PID="+str(pid)

 cursor.execute(abcd)

 results = cursor.fetchall()

 Data=[]

 for x in results:
```

```
 await channel.send(x)

 await acc.delete()

 else:

 await channel.send("`Please Enter The Password To Enter This High Level Record`")  msg2 = await
self.bot.wait_for('message', check=check)  pas=msg2.content

 if pas=="Admin":

 cursor = db.cursor()

 abcd="SELECT * FROM information where PID="+str(pid)  cursor.execute(abcd)

 results = cursor.fetchall()

 Data=[]
 for x in results:

 await channel.send(x)

 await acc.delete()

 else:

         await channel.send("Password Is Incorrect")  await acc.delete()

 except Exception as e:

 if str(e)=="1146 (42S02): Table 'criminalrecords.information' doesn't exist":

 try:

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031',
database='CriminalRecords', )

 cursor = db.cursor()

 cursor.execute("create table information (PID integer primary key,Name  varchar(20),EntryDate
date,ExitDate date,PLevel integer,Contact varchar(20))" )

 await channel.send("`Table Information Does Not Exist.....`")  time.sleep(.75)

 await channel.send("`Creating Table Information.....`")  time.sleep(.75)

 await channel.send("`Table Created`")

 db.commit()

 except Exception as a:

 raise a

 else:

 raise e


 #DEFINING_FETCH_GENERAL_DATA_PROCESS


 async def fetchdata():

 print("fetching data ")

 channel=prejectx['channel']
 author=prejectx['author']

 #import mysql.connector

 try:

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031',
database='CriminalRecords',)

 cursor = db.cursor()

 cursor.execute("SELECT * FROM information where PLevel=1" )
 results = cursor.fetchall()

 Data=[]

 for x in results:

 await channel.send(x)

 Data.append(x)

 if Data==[]:

 await channel.send("`No Records Available`")

 except Exception as e:

 if str(e)=="1146 (42S02): Table 'criminalrecords.information' doesn't exist":   #import

mysql.connector

 try:

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031',
database='CriminalRecords')

 cursor = db.cursor()

 cursor.execute("create table information (PID integer primary key,Name  varchar(20),EntryDate
date,ExitDate date,PLevel integer,Contact varchar(20))" )

 await channel.send("`Table Information Does Not Exist.....`")  time.sleep(.75)

 await channel.send("`Creating Table Information.....`")  time.sleep(.75)

 await channel.send("`Table Created`")

 db.commit()

 except Exception as a:

 print(a)

 else:

 print(e)


 #DEFINING_FETCH_DATA_USING_PID_PROCESS


 async def fetchdata2():

 channel=prejectx['channel']

 author=prejectx['author']

 def check(m):
```

```
 return m.author==author and m.channel == channel

 await channel.send("`Please Enter The Password To Enter This High Level Record`")  msg2 =

await self.bot.wait_for('message', check=check)

 pas=msg2.content

 if pas=="Admin":
 try:

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031',
database='CriminalRecords', )

 cursor = db.cursor()

 cursor.execute("SELECT * FROM information where PLevel=2 or PLevel=3" )  results =

cursor.fetchall()

 Data=[]

 for x in results:

 await channel.send(x)

 Data.append(x)

 if Data==[]:

 await channel.send("`No Records Available`")

 except Exception as e:

 if str(e)=="1146 (42S02): Table 'criminalrecords.information' doesn't exist":   try:

 db=mysql.connector.connect(host="localhost",user="root",password='aak20f031',
database='CriminalRecords',)

 cursor = db.cursor()

 cursor.execute("create table information (PID integer primary key,Name  varchar(20),EntryDate
date,ExitDate date,PLevel integer,Contact varchar(20))" )

 await channel.send("`Table Information Does Not Exist.....`")  time.sleep(.75)

 print("`Creating Table Information.....`")

 time.sleep(.75)

 await channel.send("`Table Created`")

 db.commit()

 except Exception as a:

 print(a)

 else:

 print(e)

 else:

 await channel.send("`Passkey was incorrect`")

 #SUBMENU_TO_ACCESS_DATA


 async def submenu():
```

```
 channel=prejectx['channel']

 author=prejectx['author']

 strx='''``` ACCESS RECORDS MENU\n===============================\n1. Access Specific  Records\n2. Access
Level 1 Records\n3. Access Level 2 and 3 Records```'''
```